

Peregrine

AssetCenter

Référence de
programmation



© Copyright 2001 Peregrine Systems, Inc.

Tous droits réservés.

Les informations contenues dans ce document sont la propriété de Peregrine Systems, Incorporated, et ne peuvent être utilisées ou communiquées qu'avec l'autorisation écrite préalable de Peregrine Systems, Inc. La reproduction de tout ou partie de ce manuel est soumise à l'accord écrit préalable de Peregrine Systems, Inc. Cette documentation désigne de nombreux produits par leur marque. La plupart de ces citations sont des marques déposées de leurs propriétaires respectifs.

Peregrine Systems® et AssetCenter® sont des marques déposées de Peregrine Systems, Inc.

Les logiciels décrits dans ce manuel sont fournis avec un contrat de licence entre Peregrine Systems, Inc., et l'utilisateur final ; ils doivent être utilisés suivant les termes de ce contrat. Les informations contenues dans ce document sont susceptibles d'être modifiées sans préavis et sont fournies sans engagement aucun de la part de Peregrine Systems, Inc. Contactez le support client de Peregrine Systems, Inc. pour contrôler la date de la dernière version de ce document.

Les noms de personnes et de sociétés cités dans le manuel, dans la base d'exemple ou dans les visites guidées sont fictifs et sont destinés à illustrer l'utilisation des logiciels. Toute ressemblance avec des sociétés ou personnes existantes ou ayant existé n'est qu'une pure coïncidence.

Cette édition s'applique à la version 4.0.0 du programme sous contrat de licence
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



Table des matières

I. Introduction	53
Chapitre 1. Classification des fonctions	55
Familles de fonctions	55
Champs d'application des fonctions	56
Domaines fonctionnels	57
Chapitre 2. Conventions	59
Conventions d'écriture	59
Format des constantes de type Date+Heure dans les scripts	60
A propos des dates	61
Format des constantes de type Durée	61
Chapitre 3. Définitions	63
Définition d'une fonction	63
Définition du lien virtuel CurrentUser	64
Définition	64
Equivalences	64
Définition d'un handle	65
Définition d'un code d'erreur	65

A partir d'outils externes	65
En interne	66
Chapitre 4. Typage des fonctions et des paramètres de fonctions	69
Liste des types	69
Type d'une fonction	70
Type d'un paramètre	70
II. Utilisation des API	73
Chapitre 5. Préambule	75
Avertissement	76
Installation	76
Chapitre 6. Méthodologie	79
Chapitre 7. Concepts et exemples	81
Concepts	81
Manipuler les dates	82
Premier exemple	83
Second exemple	84
III. Référence alphabétique	87
Chapitre 8. Référence alphabétique	89
Abs()	89
Syntaxe BASIC interne	89
Champ d'application	89
Entrée	90
Sortie	90
Exemple	90
AmActionDde()	90
Syntaxe API	90
Syntaxe BASIC interne	91
Champ d'application	91
Entrée	91
Sortie	92
AmActionExec()	92
Syntaxe API	92

Syntaxe BASIC interne	93
Champ d'application	93
Entrée	93
Sortie	93
Exemple	94
AmActionMail()	94
Syntaxe API	94
Syntaxe BASIC interne	94
Champ d'application	95
Entrée	95
Sortie	96
AmActionPrint()	96
Syntaxe API	96
Syntaxe BASIC interne	96
Champ d'application	96
Entrée	97
Sortie	97
AmActionPrintPreview()	97
Syntaxe BASIC interne	97
Champ d'application	97
Entrée	98
Sortie	98
AmActionPrintTo()	98
Syntaxe API	98
Syntaxe BASIC interne	98
Champ d'application	99
Entrée	99
Sortie	99
AmAddAllPOLinesToInv()	99
Syntaxe API	99
Syntaxe BASIC interne	100
Champ d'application	100
Entrée	100
Sortie	100
AmAddCatRefAndCompositionToPOrder()	100
Syntaxe API	101
Syntaxe BASIC interne	101
Champ d'application	101
Entrée	101
Sortie	102
Remarques	102
AmAddCatRefToPOrder()	102
Syntaxe API	102
Syntaxe BASIC interne	102

Champ d'application	102
Sortie	103
AmAddEstimLinesToPO()	103
Syntaxe API	103
Syntaxe BASIC interne	103
Champ d'application	103
Entrée	104
Sortie	104
AmAddEstimLineToPO()	104
Syntaxe API	104
Syntaxe BASIC interne	105
Champ d'application	105
Entrée	105
Sortie	105
AmAddPOLineToInv()	106
Syntaxe API	106
Syntaxe BASIC interne	106
Champ d'application	106
Entrée	106
Sortie	107
AmAddPOOrderLineToReceipt()	107
Syntaxe API	107
Syntaxe BASIC interne	107
Champ d'application	107
Entrée	108
Sortie	108
AmAddReceiptLineToInvoice()	108
Syntaxe API	109
Syntaxe BASIC interne	109
Champ d'application	109
Entrée	109
Sortie	110
AmAddReqLinesToEstim()	110
Syntaxe API	110
Syntaxe BASIC interne	110
Champ d'application	110
Entrée	111
Sortie	111
AmAddReqLinesToPO()	111
Syntaxe API	111
Syntaxe BASIC interne	111
Champ d'application	111
Entrée	112
Sortie	112

AmAddReqLineToEstim()	112
Syntaxe API	112
Syntaxe BASIC interne	113
Champ d'application	113
Entrée	113
Sortie	113
AmAddReqLineToPO()	113
Syntaxe API	114
Syntaxe BASIC interne	114
Champ d'application	114
Entrée	114
Sortie	114
AmAddRequestLineToPOOrder()	115
Syntaxe API	115
Syntaxe BASIC interne	115
Champ d'application	115
Entrée	115
Sortie	116
AmAddTemplateLineToPOOrder()	116
Syntaxe API	116
Syntaxe BASIC interne	116
Champ d'application	116
Sortie	117
AmAddTemplateToPOOrder()	117
Syntaxe API	117
Syntaxe BASIC interne	117
Champ d'application	117
Entrée	118
Sortie	118
AmAddTemplateToRequest()	118
Syntaxe API	118
Syntaxe BASIC interne	119
Champ d'application	119
Entrée	119
Sortie	119
AmBusinessSecondsInDay()	120
Syntaxe API	120
Syntaxe BASIC interne	120
Champ d'application	120
Entrée	120
Sortie	120
AmCalcConsolidatedFeature()	121
Syntaxe API	121
Syntaxe BASIC interne	121

Champ d'application	121
Entrée	122
Sortie	122
AmCalcDepr()	122
Syntaxe API	122
Syntaxe BASIC interne	122
Champ d'application	122
Entrée	123
Sortie	123
AmCbkReplayEvent()	124
Syntaxe API	124
Syntaxe BASIC interne	124
Champ d'application	124
Entrée	124
Sortie	125
AmCheckTraceDone()	125
Syntaxe API	125
Syntaxe BASIC interne	125
Champ d'application	125
Entrée	126
Sortie	126
AmCleanup()	126
Syntaxe API	126
Champ d'application	126
AmClearLastError()	127
Syntaxe API	127
Syntaxe BASIC interne	127
Champ d'application	127
Sortie	127
AmCloseAllChildren()	128
Syntaxe API	128
Syntaxe BASIC interne	128
Champ d'application	128
Sortie	128
AmCloseConnection()	128
Syntaxe API	129
Champ d'application	129
Sortie	129
AmCommit()	129
Syntaxe API	129
Syntaxe BASIC interne	129
Champ d'application	129
Sortie	130
AmComputeAllLicAndInstallCounts()	130

Syntaxe API	130
Syntaxe BASIC interne	130
Champ d'application	130
Sortie	131
AmComputeLicAndInstallCounts()	131
Syntaxe API	131
Syntaxe BASIC interne	131
Champ d'application	131
Entrée	132
Sortie	132
AmConnectTrace()	132
Syntaxe API	132
Syntaxe BASIC interne	132
Champ d'application	133
Entrée	133
Sortie	134
AmConvertCurrency()	134
Syntaxe API	134
Syntaxe BASIC interne	134
Champ d'application	134
Entrée	135
Sortie	135
Remarques	135
Exemple	136
AmConvertDateBasicToUnix()	136
Syntaxe API	136
Syntaxe BASIC interne	136
Champ d'application	136
Entrée	137
Sortie	137
AmConvertDateIntlToUnix()	137
Syntaxe API	137
Syntaxe BASIC interne	137
Champ d'application	138
Entrée	138
Sortie	138
AmConvertDateStringToUnix()	138
Syntaxe API	138
Syntaxe BASIC interne	139
Champ d'application	139
Entrée	139
Sortie	139
AmConvertDateUnixToBasic()	139
Syntaxe API	140

Syntaxe BASIC interne	140
Champ d'application	140
Entrée	140
Sortie	140
AmConvertDateUnixToIntl()	141
Syntaxe API	141
Syntaxe BASIC interne	141
Champ d'application	141
Entrée	141
Sortie	141
AmConvertDateUnixToString()	142
Syntaxe API	142
Syntaxe BASIC interne	142
Champ d'application	142
Entrée	142
Sortie	143
AmConvertDoubleToString()	143
Syntaxe API	143
Syntaxe BASIC interne	143
Champ d'application	143
Entrée	144
Sortie	144
AmConvertMonetaryToString()	144
Syntaxe API	144
Syntaxe BASIC interne	144
Champ d'application	145
Entrée	145
Sortie	145
AmConvertStringToDouble()	145
Syntaxe API	145
Syntaxe BASIC interne	146
Champ d'application	146
Entrée	146
Sortie	146
AmConvertStringToMonetary()	146
Syntaxe API	147
Syntaxe BASIC interne	147
Champ d'application	147
Entrée	147
Sortie	147
AmCounter()	148
Syntaxe BASIC interne	148
Champ d'application	148
Entrée	148

Sortie	148
Exemple	149
AmCreateAssetPort()	149
Syntaxe API	149
Syntaxe BASIC interne	150
Champ d'application	150
Entrée	150
Sortie	150
AmCreateAssetsAwaitingDelivery()	151
Syntaxe API	151
Syntaxe BASIC interne	151
Champ d'application	151
Entrée	152
Sortie	152
AmCreateCable()	152
Syntaxe API	152
Syntaxe BASIC interne	152
Champ d'application	153
Entrée	153
Sortie	153
AmCreateCableBundle()	154
Syntaxe API	154
Syntaxe BASIC interne	154
Champ d'application	154
Entrée	155
Sortie	155
AmCreateCableLink()	155
Syntaxe API	156
Syntaxe BASIC interne	156
Champ d'application	156
Entrée	157
Sortie	157
AmCreateDelivFromPO()	157
Syntaxe API	157
Syntaxe BASIC interne	158
Champ d'application	158
Entrée	158
Sortie	158
AmCreateDevice()	159
Syntaxe API	159
Syntaxe BASIC interne	159
Champ d'application	159
Entrée	160
Sortie	160

AmCreateDeviceLink()	161
Syntaxe API	161
Syntaxe BASIC interne	161
Champ d'application	161
Entrée	162
Sortie	162
AmCreateEstimFromReq()	163
Syntaxe API	163
Syntaxe BASIC interne	163
Champ d'application	163
Entrée	163
Sortie	163
AmCreateEstimsFromAllReqLines()	164
Syntaxe API	164
Syntaxe BASIC interne	164
Champ d'application	164
Entrée	165
Sortie	165
AmCreateInvFromPO()	165
Syntaxe API	165
Syntaxe BASIC interne	165
Champ d'application	165
Entrée	166
Sortie	166
AmCreateLink()	166
Syntaxe API	166
Syntaxe BASIC interne	167
Champ d'application	167
Entrée	167
Sortie	167
AmCreatePOFromEstim()	167
Syntaxe API	168
Syntaxe BASIC interne	168
Champ d'application	168
Entrée	168
Sortie	168
AmCreatePOFromReq()	169
Syntaxe API	169
Syntaxe BASIC interne	169
Champ d'application	169
Entrée	169
Sortie	169
AmCreatePOOrderFromRequest()	170
Syntaxe API	170

Syntaxe BASIC interne	170
Champ d'application	170
Entrée	171
Sortie	171
AmCreatePOrdersFromRequest()	171
Syntaxe API	171
Syntaxe BASIC interne	171
Champ d'application	171
Entrée	172
Sortie	172
AmCreatePOsFromAllReqLines()	172
Syntaxe API	172
Syntaxe BASIC interne	172
Champ d'application	172
Entrée	173
Sortie	173
AmCreateProjectCable()	173
Syntaxe API	174
Syntaxe BASIC interne	174
Champ d'application	174
Entrée	174
Sortie	174
AmCreateProjectDevice()	175
Syntaxe API	175
Syntaxe BASIC interne	175
Champ d'application	175
Entrée	176
Sortie	176
AmCreateProjectTrace()	176
Syntaxe API	177
Syntaxe BASIC interne	177
Champ d'application	177
Entrée	177
Sortie	178
AmCreateReceiptFromPOrder()	178
Syntaxe API	178
Syntaxe BASIC interne	178
Champ d'application	179
Entrée	179
Sortie	179
AmCreateRecord()	179
Syntaxe API	179
Syntaxe BASIC interne	180
Champ d'application	180

Entrée	180
Exemple	180
AmCreateRequestToInvoice()	180
Syntaxe API	181
Syntaxe BASIC interne	181
Champ d'application	181
Entrée	181
Sortie	182
Remarques	182
AmCreateRequestToPOrder()	182
Syntaxe API	182
Syntaxe BASIC interne	183
Champ d'application	183
Entrée	183
Sortie	184
AmCreateRequestToReceipt()	184
Syntaxe API	184
Syntaxe BASIC interne	184
Champ d'application	184
Entrée	185
Sortie	185
Remarques	186
AmCreateReturnFromReceipt()	186
Syntaxe API	186
Syntaxe BASIC interne	186
Champ d'application	186
Entrée	186
Sortie	187
AmCreateTraceHist()	187
Syntaxe API	187
Syntaxe BASIC interne	187
Champ d'application	187
Entrée	188
Sortie	188
AmCryptPassword()	188
Syntaxe API	189
Syntaxe BASIC interne	189
Champ d'application	189
Entrée	189
Sortie	189
AmCurrentDate()	190
Syntaxe API	190
Syntaxe BASIC interne	190
Champ d'application	190

Sortie	190
AmCurrentIsoLang()	191
Syntaxe API	191
Syntaxe BASIC interne	191
Champ d'application	191
Sortie	191
AmCurrentLanguage()	192
Syntaxe API	192
Syntaxe BASIC interne	192
Champ d'application	192
Sortie	192
AmCurrentServerDate()	193
Syntaxe API	193
Syntaxe BASIC interne	193
Champ d'application	193
Sortie	193
AmDateAdd()	194
Syntaxe API	194
Syntaxe BASIC interne	194
Champ d'application	194
Entrée	194
Sortie	194
AmDateAddLogical()	195
Syntaxe API	195
Syntaxe BASIC interne	195
Champ d'application	195
Entrée	196
Sortie	196
AmDateDiff()	196
Syntaxe API	196
Syntaxe BASIC interne	196
Champ d'application	196
Entrée	197
Sortie	197
Exemple	197
AmDbGetDate()	197
Syntaxe API	198
Syntaxe BASIC interne	198
Champ d'application	198
Entrée	198
Sortie	198
AmDbGetDouble()	199
Syntaxe API	199
Syntaxe BASIC interne	199

Champ d'application	199
Entrée	199
Sortie	199
AmDbGetList()	200
Syntaxe API	200
Syntaxe BASIC interne	200
Champ d'application	200
Entrée	201
Sortie	201
AmDbGetListEx()	201
Syntaxe API	201
Syntaxe BASIC interne	202
Champ d'application	202
Entrée	202
Sortie	202
AmDbGetLong()	203
Syntaxe API	203
Syntaxe BASIC interne	203
Champ d'application	203
Entrée	203
Sortie	204
Exemple	204
AmDbGetPk()	204
Syntaxe API	204
Syntaxe BASIC interne	204
Champ d'application	205
Entrée	205
Sortie	205
AmDbGetString()	205
Syntaxe API	206
Syntaxe BASIC interne	206
Champ d'application	206
Entrée	206
Sortie	207
Remarques	207
Exemple	207
AmDbGetStringEx()	207
Syntaxe API	208
Syntaxe BASIC interne	208
Champ d'application	208
Entrée	208
Sortie	209
AmDeadLine()	209
Syntaxe API	209

Syntaxe BASIC interne	209
Champ d'application	209
Entrée	210
Sortie	210
Exemple	210
AmDecrementLogLevel()	210
Syntaxe BASIC interne	211
Champ d'application	211
Sortie	211
AmDefAssignee()	211
Syntaxe API	211
Syntaxe BASIC interne	212
Champ d'application	212
Entrée	212
Sortie	212
Exemple	212
AmDefaultCurrency()	213
Syntaxe API	213
Syntaxe BASIC interne	213
Champ d'application	213
Sortie	213
AmDefEscalationScheme()	214
Syntaxe API	214
Syntaxe BASIC interne	214
Champ d'application	214
Entrée	214
Sortie	215
Exemple	215
AmDefGroup()	215
Syntaxe API	215
Syntaxe BASIC interne	216
Champ d'application	216
Entrée	216
Sortie	216
Exemple	217
AmDeleteLink()	217
Syntaxe API	217
Syntaxe BASIC interne	217
Champ d'application	217
Entrée	218
Sortie	218
AmDeleteRecord()	218
Syntaxe API	218
Syntaxe BASIC interne	218

Champ d'application	218
Entrée	219
Sortie	219
AmDisconnectTrace()	219
Syntaxe API	219
Syntaxe BASIC interne	219
Champ d'application	219
Entrée	220
Sortie	220
AmDuplicateRecord()	220
Syntaxe API	220
Syntaxe BASIC interne	221
Champ d'application	221
Entrée	221
Sortie	221
AmEndOfNthBusinessDay()	221
Syntaxe API	222
Syntaxe BASIC interne	222
Champ d'application	222
Entrée	222
Sortie	222
AmEvalScript()	223
Syntaxe BASIC interne	223
Champ d'application	223
Entrée	223
Sortie	224
AmExecTransition()	224
Syntaxe BASIC interne	224
Champ d'application	224
Entrée	225
Sortie	225
AmExecuteActionById()	225
Syntaxe API	225
Syntaxe BASIC interne	225
Champ d'application	226
Entrée	226
Sortie	226
AmExecuteActionByName()	226
Syntaxe API	227
Syntaxe BASIC interne	227
Champ d'application	227
Entrée	227
Sortie	228
AmExportDocument()	228

Syntaxe API	228
Syntaxe BASIC interne	228
Champ d'application	228
Entrée	228
Sortie	229
AmFindCable()	229
Syntaxe API	229
Syntaxe BASIC interne	229
Champ d'application	229
Entrée	230
Sortie	230
AmFindDevice()	230
Syntaxe API	231
Syntaxe BASIC interne	231
Champ d'application	231
Entrée	231
Sortie	231
AmFindRootLink()	232
Syntaxe API	232
Syntaxe BASIC interne	232
Champ d'application	232
Sortie	232
AmFindTermDevice()	233
Syntaxe API	233
Syntaxe BASIC interne	233
Champ d'application	234
Entrée	234
Sortie	234
AmFindTermField()	235
Syntaxe API	235
Syntaxe BASIC interne	235
Champ d'application	235
Entrée	236
Sortie	236
AmGenSqlName()	236
Syntaxe API	236
Syntaxe BASIC interne	236
Champ d'application	237
Entrée	237
Sortie	237
Exemple	237
AmGetComputeString()	238
Syntaxe API	238
Syntaxe BASIC interne	238

Champ d'application	238
Entrée	238
Sortie	239
Exemple	239
AmGetCurrentNTDomain()	239
Syntaxe API	239
Syntaxe BASIC interne	239
Champ d'application	239
Sortie	240
AmGetCurrentNTUser()	240
Syntaxe API	240
Syntaxe BASIC interne	240
Champ d'application	240
Sortie	241
AmGetFeat()	241
Syntaxe API	241
Syntaxe BASIC interne	241
Champ d'application	241
Entrée	242
AmGetFeatCount()	242
Syntaxe API	242
Syntaxe BASIC interne	242
Champ d'application	242
Entrée	243
Sortie	243
AmGetField()	243
Syntaxe API	243
Syntaxe BASIC interne	243
Champ d'application	243
Entrée	244
AmGetFieldCount()	244
Syntaxe API	244
Syntaxe BASIC interne	244
Champ d'application	244
Entrée	245
Sortie	245
AmGetFieldDateValue()	245
Syntaxe API	245
Syntaxe BASIC interne	245
Champ d'application	246
Entrée	246
Sortie	246
AmGetFieldDescription()	246
Syntaxe API	247

Syntaxe BASIC interne	247
Champ d'application	247
Entrée	247
Sortie	247
AmGetFieldDoubleValue()	248
Syntaxe API	248
Syntaxe BASIC interne	248
Champ d'application	248
Entrée	248
Sortie	248
AmGetFieldFormat()	249
Syntaxe API	250
Syntaxe BASIC interne	250
Champ d'application	250
Entrée	251
Sortie	251
AmGetFieldFormatFromName()	251
Syntaxe API	251
Syntaxe BASIC interne	251
Champ d'application	251
Entrée	252
Sortie	252
AmGetFieldFromName()	252
Syntaxe API	252
Syntaxe BASIC interne	253
Champ d'application	253
Entrée	253
AmGetFieldLabel()	253
Syntaxe API	253
Syntaxe BASIC interne	253
Champ d'application	254
Entrée	254
Sortie	254
AmGetFieldLabelFromName()	254
Syntaxe API	254
Syntaxe BASIC interne	255
Champ d'application	255
Entrée	255
Sortie	255
AmGetFieldLongValue()	256
Syntaxe API	256
Syntaxe BASIC interne	256
Champ d'application	256
Entrée	256

Sortie	256
Remarques	257
AmGetFieldName()	257
Syntaxe API	257
Syntaxe BASIC interne	257
Champ d'application	257
Entrée	258
Sortie	258
AmGetFieldRights()	258
Syntaxe API	259
Syntaxe BASIC interne	259
Champ d'application	259
Entrée	259
Sortie	259
AmGetFieldSize()	260
Syntaxe API	260
Syntaxe BASIC interne	260
Champ d'application	260
Entrée	260
Sortie	261
AmGetFieldSqlName()	261
Syntaxe API	261
Syntaxe BASIC interne	261
Champ d'application	261
Entrée	262
Sortie	262
AmGetFieldStrValue()	262
Syntaxe API	263
Syntaxe BASIC interne	263
Champ d'application	263
Entrée	264
Sortie	264
AmGetFieldType()	264
Syntaxe API	264
Syntaxe BASIC interne	264
Champ d'application	264
Entrée	265
Sortie	265
Remarques	265
AmGetFieldUserType()	265
Syntaxe API	266
Syntaxe BASIC interne	266
Champ d'application	266
Entrée	267

Sortie	267
AmGetForeignKey()	267
Syntaxe API	267
Syntaxe BASIC interne	267
Champ d'application	268
Entrée	268
AmGetIndex()	268
Syntaxe API	268
Syntaxe BASIC interne	268
Champ d'application	268
Entrée	269
AmGetIndexCount()	269
Syntaxe API	269
Syntaxe BASIC interne	269
Champ d'application	269
Entrée	270
Sortie	270
AmGetIndexField()	270
Syntaxe API	270
Syntaxe BASIC interne	270
Champ d'application	270
Entrée	271
AmGetIndexFieldCount()	271
Syntaxe API	271
Syntaxe BASIC interne	271
Champ d'application	271
Entrée	272
Sortie	272
AmGetIndexFlags()	272
Syntaxe API	272
Syntaxe BASIC interne	272
Champ d'application	272
Entrée	273
Sortie	273
Remarques	273
AmGetIndexName()	273
Syntaxe API	274
Syntaxe BASIC interne	274
Champ d'application	274
Entrée	274
Sortie	274
AmGetLink()	275
Syntaxe API	275
Syntaxe BASIC interne	275

Champ d'application	275
Entrée	275
AmGetLinkCardinality()	275
Syntaxe API	276
Syntaxe BASIC interne	276
Champ d'application	276
Entrée	276
Sortie	276
AmGetLinkCount()	276
Syntaxe API	277
Syntaxe BASIC interne	277
Champ d'application	277
Entrée	277
Sortie	277
AmGetLinkDstField()	278
Syntaxe API	278
Syntaxe BASIC interne	278
Champ d'application	278
Entrée	278
AmGetLinkFeatureValue()	278
Syntaxe API	279
Syntaxe BASIC interne	279
Champ d'application	279
Entrée	279
Sortie	279
Exemple	280
AmGetLinkFromName()	280
Syntaxe API	280
Syntaxe BASIC interne	280
Champ d'application	281
Entrée	281
AmGetLinkType()	281
Syntaxe API	281
Syntaxe BASIC interne	281
Champ d'application	281
Entrée	282
Sortie	282
AmGetMainField()	282
Syntaxe API	282
Syntaxe BASIC interne	282
Champ d'application	282
Entrée	283
AmGetNextAssetPin()	283
Syntaxe API	283

Syntaxe BASIC interne	283
Champ d'application	283
Entrée	284
Sortie	284
AmGetNextAssetPort()	284
Syntaxe API	285
Syntaxe BASIC interne	285
Champ d'application	285
Entrée	285
Sortie	286
AmGetNextCableBundle()	286
Syntaxe API	287
Syntaxe BASIC interne	287
Champ d'application	287
Entrée	287
Sortie	288
AmGetNextCablePair()	288
Syntaxe API	288
Syntaxe BASIC interne	288
Champ d'application	288
Entrée	289
Sortie	289
AmGetNTDomains()	289
Syntaxe API	289
Syntaxe BASIC interne	290
Champ d'application	290
Sortie	290
AmGetNTMachinesInDomain()	290
Syntaxe API	290
Syntaxe BASIC interne	291
Champ d'application	291
Entrée	291
Sortie	291
AmGetNTUsersInDomain()	291
Syntaxe API	292
Syntaxe BASIC interne	292
Champ d'application	292
Entrée	292
Sortie	292
AmGetPOLinePrice()	293
Syntaxe API	293
Syntaxe BASIC interne	293
Champ d'application	293
Entrée	293

Sortie	293
AmGetPOLinePriceCur()	294
Syntaxe API	294
Syntaxe BASIC interne	294
Champ d'application	294
Entrée	294
Sortie	295
AmGetPOLinePricing()	295
Syntaxe API	295
Syntaxe BASIC interne	295
Champ d'application	295
Sortie	296
AmGetPOLineReference()	296
Syntaxe API	296
Syntaxe BASIC interne	296
Champ d'application	296
Entrée	297
Sortie	297
AmGetRecordFromMainId()	297
Syntaxe API	297
Syntaxe BASIC interne	297
Champ d'application	297
Entrée	298
Remarques	298
AmGetRecordHandle()	298
Syntaxe API	298
Syntaxe BASIC interne	299
Champ d'application	299
Entrée	299
AmGetRecordId()	299
Syntaxe API	299
Syntaxe BASIC interne	299
Champ d'application	300
Entrée	300
Sortie	300
AmGetRelDstField()	300
Syntaxe API	300
Syntaxe BASIC interne	301
Champ d'application	301
Entrée	301
AmGetRelSrcField()	301
Syntaxe API	301
Syntaxe BASIC interne	301
Champ d'application	302

Entrée	302
AmGetRelTable()	302
Syntaxe API	302
Syntaxe BASIC interne	302
Champ d'application	302
Entrée	303
Sortie	303
AmGetReverseLink()	303
Syntaxe API	303
Syntaxe BASIC interne	303
Champ d'application	303
Entrée	304
AmGetSelfFromMainId()	304
Syntaxe API	304
Syntaxe BASIC interne	304
Champ d'application	304
Entrée	305
Sortie	305
AmGetSourceTable()	305
Syntaxe API	305
Syntaxe BASIC interne	305
Champ d'application	305
Entrée	306
Sortie	306
AmGetTable()	306
Syntaxe API	306
Syntaxe BASIC interne	306
Champ d'application	306
Entrée	307
Sortie	307
AmGetTableCount()	307
Syntaxe API	307
Syntaxe BASIC interne	307
Champ d'application	307
Sortie	308
AmGetTableDescription()	308
Syntaxe API	308
Syntaxe BASIC interne	308
Champ d'application	308
Entrée	309
Sortie	309
AmGetTableFromName()	309
Syntaxe API	309
Syntaxe BASIC interne	310

Champ d'application	310
Entrée	310
Sortie	310
AmGetTableLabel()	310
Syntaxe API	310
Syntaxe BASIC interne	311
Champ d'application	311
Entrée	311
Sortie	311
AmGetTableName()	311
Syntaxe API	312
Syntaxe BASIC interne	312
Champ d'application	312
Entrée	312
Sortie	312
AmGetTableRights()	313
Syntaxe API	313
Syntaxe BASIC interne	313
Champ d'application	313
Entrée	314
Sortie	314
AmGetTableSqlName()	314
Syntaxe API	314
Syntaxe BASIC interne	314
Champ d'application	314
Entrée	315
Sortie	315
AmGetTargetTable()	315
Syntaxe API	315
Syntaxe BASIC interne	315
Champ d'application	316
Entrée	316
Sortie	316
AmGetTrace()	316
Syntaxe API	317
Syntaxe BASIC interne	317
Champ d'application	317
Entrée	317
Sortie	318
AmGetTraceFromHist()	318
Syntaxe API	318
Syntaxe BASIC interne	318
Champ d'application	319
Entrée	319

Sortie	319
AmGetTypedLinkField()	320
Syntaxe API	320
Syntaxe BASIC interne	320
Champ d'application	320
Entrée	320
AmGetVersion()	320
Syntaxe API	321
Syntaxe BASIC interne	321
Champ d'application	321
Sortie	321
AmHasAdminPrivilege()	321
Syntaxe API	322
Syntaxe BASIC interne	322
Champ d'application	322
Sortie	322
AmHasRelTable()	322
Syntaxe API	323
Syntaxe BASIC interne	323
Champ d'application	323
Entrée	323
Sortie	323
AmImportDocument()	324
Syntaxe API	324
Syntaxe BASIC interne	324
Champ d'application	324
Sortie	324
AmIncrementLogLevel()	325
Syntaxe BASIC interne	325
Champ d'application	325
Entrée	325
Sortie	325
AmInsertRecord()	326
Syntaxe API	326
Syntaxe BASIC interne	326
Champ d'application	326
Entrée	326
Sortie	327
AmInstantiateReqLine()	327
Syntaxe API	327
Syntaxe BASIC interne	327
Champ d'application	327
Entrée	328
Sortie	328

Remarques	328
AmInstantiateRequest()	328
Syntaxe API	328
Syntaxe BASIC interne	328
Champ d'application	329
Entrée	329
Sortie	329
AmIsConnected()	329
Syntaxe API	329
Champ d'application	329
Sortie	330
AmIsFieldForeignKey()	330
Syntaxe API	330
Syntaxe BASIC interne	330
Champ d'application	330
Entrée	331
Sortie	331
AmIsFieldIndexed()	331
Syntaxe API	331
Syntaxe BASIC interne	331
Champ d'application	331
Entrée	332
Sortie	332
AmIsFieldPrimaryKey()	332
Syntaxe API	332
Syntaxe BASIC interne	332
Champ d'application	332
Entrée	333
Sortie	333
AmIsLink()	333
Syntaxe API	333
Syntaxe BASIC interne	333
Champ d'application	333
Entrée	334
Sortie	334
AmIsTypedLink()	334
Syntaxe API	334
Syntaxe BASIC interne	334
Champ d'application	334
Entrée	335
Sortie	335
AmLastError()	335
Syntaxe API	335
Syntaxe BASIC interne	335

Champ d'application	335
Sortie	336
AmLastErrorMsg()	336
Syntaxe API	336
Syntaxe BASIC interne	336
Champ d'application	336
Sortie	337
AmListToString()	337
Syntaxe API	337
Syntaxe BASIC interne	337
Champ d'application	337
Entrée	338
Sortie	338
AmLog()	338
Syntaxe BASIC interne	339
Champ d'application	339
Entrée	339
Sortie	339
Exemple	339
AmLoginId()	339
Syntaxe API	340
Syntaxe BASIC interne	340
Champ d'application	340
Sortie	340
Exemple	340
AmLoginName()	341
Syntaxe API	341
Syntaxe BASIC interne	341
Champ d'application	341
Sortie	341
Exemple	342
AmMapSubReqLineAgent()	342
Syntaxe API	342
Syntaxe BASIC interne	342
Champ d'application	342
Entrée	343
Sortie	343
AmMoveCable()	343
Syntaxe API	343
Syntaxe BASIC interne	343
Champ d'application	344
Entrée	344
Sortie	344
AmMoveDevice()	344

Syntaxe API	345
Syntaxe BASIC interne	345
Champ d'application	345
Entrée	345
Sortie	346
AmMsgBox()	346
Syntaxe BASIC interne	346
Champ d'application	346
Entrée	346
Sortie	347
Exemple	347
AmOpenConnection()	347
Syntaxe API	347
Champ d'application	347
Entrée	348
AmOpenScreen()	348
Syntaxe BASIC interne	348
Champ d'application	348
Entrée	348
Sortie	349
AmPagePath()	349
Syntaxe BASIC interne	349
Champ d'application	349
Sortie	350
AmProgress()	350
Syntaxe BASIC interne	350
Champ d'application	350
Entrée	351
Sortie	351
Exemple	351
AmQueryCreate()	351
Syntaxe API	351
Syntaxe BASIC interne	351
Champ d'application	351
AmQueryExec()	352
Syntaxe API	352
Syntaxe BASIC interne	352
Champ d'application	352
Entrée	353
Sortie	353
AmQueryGet()	353
Syntaxe API	353
Syntaxe BASIC interne	353
Champ d'application	353

Entrée	354
Sortie	354
AmQueryNext()	354
Syntaxe API	354
Syntaxe BASIC interne	354
Champ d'application	354
Entrée	355
Sortie	355
AmQuerySetAddMainField()	355
Syntaxe API	355
Syntaxe BASIC interne	355
Champ d'application	356
Entrée	356
Sortie	356
AmQuerySetFullMemo()	356
Syntaxe API	356
Syntaxe BASIC interne	357
Champ d'application	357
Entrée	357
Sortie	357
AmQueryStartTable()	357
Syntaxe API	358
Syntaxe BASIC interne	358
Champ d'application	358
Entrée	358
Sortie	358
AmQueryStop()	358
Syntaxe API	359
Syntaxe BASIC interne	359
Champ d'application	359
Entrée	359
Sortie	359
AmReceiveAllPOLines()	360
Syntaxe API	360
Syntaxe BASIC interne	360
Champ d'application	360
Entrée	360
Sortie	361
AmReceivePOLine()	361
Syntaxe API	361
Syntaxe BASIC interne	361
Champ d'application	361
Entrée	362
Sortie	362

AmRefreshAllCaches()	362
Syntaxe API	362
Syntaxe BASIC interne	363
Champ d'application	363
Sortie	363
AmRefreshLabel()	363
Syntaxe API	363
Syntaxe BASIC interne	363
Champ d'application	364
Entrée	364
Sortie	364
AmRefreshProperty()	364
Syntaxe BASIC interne	365
Champ d'application	365
Entrée	365
Sortie	365
AmRefreshTraceHist()	365
Syntaxe API	366
Syntaxe BASIC interne	366
Champ d'application	366
Entrée	366
Sortie	366
AmReleaseHandle()	366
Syntaxe API	367
Syntaxe BASIC interne	367
Champ d'application	367
Entrée	367
Sortie	367
AmRemoveCable()	367
Syntaxe API	368
Syntaxe BASIC interne	368
Champ d'application	368
Entrée	368
Sortie	369
AmRemoveDevice()	369
Syntaxe API	369
Syntaxe BASIC interne	369
Champ d'application	369
Entrée	370
Sortie	370
AmReturnAsset()	370
Syntaxe API	370
Syntaxe BASIC interne	370
Champ d'application	370

Entrée	371
Sortie	371
AmReturnContract()	371
Syntaxe API	371
Syntaxe BASIC interne	372
Champ d'application	372
Entrée	372
Sortie	372
AmReturnPortfolioItem()	373
Syntaxe API	373
Syntaxe BASIC interne	373
Champ d'application	373
Entrée	373
Sortie	374
AmReturnTraining()	374
Syntaxe API	374
Syntaxe BASIC interne	374
Champ d'application	374
Entrée	375
Sortie	375
AmReturnWorkOrder()	375
Syntaxe API	375
Syntaxe BASIC interne	376
Champ d'application	376
Entrée	376
Sortie	376
AmRevCryptPassword()	377
Syntaxe API	377
Syntaxe BASIC interne	377
Champ d'application	377
Entrée	377
Sortie	377
AmRgbColor()	378
Syntaxe API	378
Syntaxe BASIC interne	378
Champ d'application	378
Entrée	378
Sortie	379
AmRollback()	379
Syntaxe API	379
Syntaxe BASIC interne	380
Champ d'application	380
Sortie	380
AmSetFieldDateValue()	380

Syntaxe API	380
Syntaxe BASIC interne	380
Champ d'application	381
Entrée	381
Sortie	381
AmSetFieldDoubleValue()	381
Syntaxe API	381
Syntaxe BASIC interne	382
Champ d'application	382
Entrée	382
Sortie	382
AmSetFieldLongValue()	382
Syntaxe API	383
Syntaxe BASIC interne	383
Champ d'application	383
Entrée	383
Sortie	384
AmSetFieldStrValue()	384
Syntaxe API	384
Syntaxe BASIC interne	384
Champ d'application	384
Entrée	384
Sortie	385
AmSetLinkFeatureValue()	385
Syntaxe API	385
Syntaxe BASIC interne	385
Champ d'application	385
Entrée	386
Sortie	386
AmSetProperty()	386
Syntaxe BASIC interne	386
Champ d'application	387
Entrée	387
Sortie	387
AmShowCableCrossConnect()	387
Syntaxe BASIC interne	387
Champ d'application	387
Sortie	388
AmShowDeviceCrossConnect()	388
Syntaxe BASIC interne	388
Champ d'application	388
Sortie	388
AmSqlTextConst()	389
Syntaxe API	389

Syntaxe BASIC interne	389
Champ d'application	389
Sortie	389
AmStartTransaction()	390
Syntaxe API	390
Syntaxe BASIC interne	390
Champ d'application	390
Sortie	390
AmStartup()	390
Syntaxe API	391
Champ d'application	391
AmTableDesc()	391
Syntaxe API	391
Syntaxe BASIC interne	391
Champ d'application	391
Entrée	392
Sortie	392
Exemple	392
AmTaxRate()	393
Syntaxe API	393
Syntaxe BASIC interne	393
Champ d'application	393
Entrée	393
Sortie	394
AmUpdateDetail()	394
Syntaxe BASIC interne	394
Champ d'application	394
Entrée	395
Sortie	395
AmUpdateLoginSlot()	395
Syntaxe API	395
Syntaxe BASIC interne	395
Champ d'application	395
Sortie	396
AmUpdateRecord()	396
Syntaxe API	396
Syntaxe BASIC interne	396
Champ d'application	396
Entrée	396
Sortie	397
AmValueOf()	397
Syntaxe BASIC interne	397
Champ d'application	397
Entrée	397

Sortie	397
Exemple	398
AmWizChain()	398
Syntaxe BASIC interne	398
Champ d'application	398
Entrée	399
Sortie	399
AmWorkTimeSpanBetween()	399
Syntaxe API	399
Syntaxe BASIC interne	399
Champ d'application	399
Entrée	400
Sortie	400
Exemple	400
AppendOperand()	401
Syntaxe BASIC interne	401
Champ d'application	401
Entrée	401
Sortie	401
Remarques	402
ApplyNewVals()	402
Syntaxe BASIC interne	402
Champ d'application	402
Entrée	403
Sortie	403
Asc()	403
Syntaxe BASIC interne	403
Champ d'application	403
Entrée	404
Exemple	404
Atn()	404
Syntaxe BASIC interne	404
Champ d'application	404
Entrée	405
Sortie	405
Exemple	405
BasicToLocalDate()	405
Syntaxe BASIC interne	405
Champ d'application	406
Entrée	406
Sortie	406
BasicToLocalTime()	406
Syntaxe BASIC interne	406
Champ d'application	407

Entrée	407
Sortie	407
BasicToLocalTimeStamp()	407
Syntaxe BASIC interne	407
Champ d'application	408
Entrée	408
Sortie	408
Beep()	408
Syntaxe BASIC interne	408
Champ d'application	408
Sortie	409
CDBl()	409
Syntaxe BASIC interne	409
Champ d'application	409
Entrée	410
Sortie	410
Exemple	410
ChDir()	410
Syntaxe BASIC interne	410
Champ d'application	410
Entrée	411
Sortie	411
ChDrive()	411
Syntaxe BASIC interne	411
Champ d'application	411
Entrée	412
Sortie	412
Chr()	412
Syntaxe BASIC interne	412
Champ d'application	412
Entrée	413
Sortie	413
Exemple	413
CInt()	414
Syntaxe BASIC interne	414
Champ d'application	414
Entrée	414
Sortie	414
Exemple	415
CLng()	415
Syntaxe BASIC interne	415
Champ d'application	415
Entrée	415
Sortie	415

Exemple	416
Cos()	416
Syntaxe BASIC interne	416
Champ d'application	416
Entrée	417
Sortie	417
Exemple	417
CountOccurrences()	417
Syntaxe BASIC interne	417
Champ d'application	417
Entrée	418
Sortie	418
Exemple	418
CountValues()	419
Syntaxe BASIC interne	419
Champ d'application	419
Entrée	419
Sortie	419
Exemple	420
CSng()	420
Syntaxe BASIC interne	420
Champ d'application	420
Entrée	420
Sortie	421
Exemple	421
CStr()	421
Syntaxe BASIC interne	421
Champ d'application	421
Entrée	422
Sortie	422
Exemple	422
CurDir()	422
Syntaxe BASIC interne	422
Champ d'application	423
Sortie	423
CVar()	423
Syntaxe BASIC interne	423
Champ d'application	423
Entrée	424
Sortie	424
Date()	424
Syntaxe BASIC interne	424
Champ d'application	424
Sortie	425

DateSerial()	425
Syntaxe BASIC interne	425
Champ d'application	425
Entrée	426
Sortie	426
Exemple	426
DateValue()	427
Syntaxe BASIC interne	427
Champ d'application	427
Entrée	427
Sortie	427
Exemple	428
Day()	428
Syntaxe BASIC interne	428
Champ d'application	428
Entrée	428
Sortie	429
Exemple	429
EscapeSeparators()	429
Syntaxe BASIC interne	429
Champ d'application	429
Entrée	430
Sortie	430
Exemple	430
ExeDir()	430
Syntaxe BASIC interne	431
Champ d'application	431
Sortie	431
Exemple	431
Exp()	431
Syntaxe BASIC interne	432
Champ d'application	432
Entrée	432
Sortie	432
Exemple	432
ExtractValue()	433
Syntaxe BASIC interne	433
Champ d'application	433
Entrée	433
Sortie	433
Exemple	434
FileCopy()	434
Syntaxe BASIC interne	434
Champ d'application	435

Entrée	435
Sortie	435
FileDateTime()	435
Syntaxe BASIC interne	435
Champ d'application	435
Entrée	436
Sortie	436
FileLen()	436
Syntaxe BASIC interne	436
Champ d'application	436
Entrée	437
Sortie	437
Fix()	437
Syntaxe BASIC interne	437
Champ d'application	437
Entrée	438
Sortie	438
Exemple	438
FormatDate()	438
Syntaxe BASIC interne	439
Champ d'application	439
Entrée	439
Sortie	439
Exemple	439
FormatResString()	440
Syntaxe BASIC interne	440
Champ d'application	440
Entrée	440
Sortie	441
Exemple	441
FormatString()	441
Champ d'application	441
Exemple	442
FV()	442
Syntaxe BASIC interne	442
Champ d'application	442
Entrée	443
Sortie	443
Remarques	444
GetListItem()	444
Syntaxe BASIC interne	444
Champ d'application	444
Entrée	444
Sortie	445

Exemple	445
Hex()	445
Syntaxe BASIC interne	445
Champ d'application	445
Entrée	446
Sortie	446
Hour()	446
Syntaxe BASIC interne	446
Champ d'application	446
Entrée	447
Sortie	447
Exemple	447
InStr()	447
Syntaxe BASIC interne	448
Champ d'application	448
Entrée	448
Sortie	448
Exemple	449
Int()	449
Syntaxe BASIC interne	449
Champ d'application	449
Entrée	449
Sortie	450
Exemple	450
IPMT()	450
Syntaxe BASIC interne	450
Champ d'application	450
Entrée	451
Sortie	451
Remarques	452
IsNumeric()	452
Syntaxe BASIC interne	452
Champ d'application	452
Entrée	453
Sortie	453
Kill()	453
Syntaxe BASIC interne	453
Champ d'application	453
Entrée	454
Sortie	454
LCase()	454
Syntaxe BASIC interne	454
Champ d'application	454
Entrée	454

Sortie	454
Exemple	455
Left()	456
Syntaxe BASIC interne	456
Champ d'application	456
Entrée	456
Sortie	456
Exemple	457
LeftPart()	457
Syntaxe BASIC interne	457
Champ d'application	457
Entrée	458
Sortie	458
Exemple	458
LeftPartFromRight()	459
Syntaxe BASIC interne	459
Champ d'application	459
Entrée	459
Sortie	459
Exemple	460
Len()	460
Syntaxe BASIC interne	460
Champ d'application	460
Entrée	461
Sortie	461
Exemple	461
LocalToBasicDate()	462
Syntaxe BASIC interne	462
Champ d'application	462
Entrée	462
Sortie	462
LocalToBasicTime()	463
Syntaxe BASIC interne	463
Champ d'application	463
Entrée	463
Sortie	463
LocalToBasicTimeStamp()	464
Syntaxe BASIC interne	464
Champ d'application	464
Entrée	464
Sortie	464
LocalToUTCDate()	465
Syntaxe BASIC interne	465
Champ d'application	465

Entrée	465
Sortie	465
Log()	466
Syntaxe BASIC interne	466
Champ d'application	466
Entrée	466
Sortie	466
Exemple	467
LTrim()	467
Syntaxe BASIC interne	467
Champ d'application	467
Entrée	467
Sortie	467
Exemple	468
MakeInvertBool()	468
Syntaxe BASIC interne	469
Champ d'application	469
Entrée	469
Sortie	469
Exemple	469
Mid()	470
Syntaxe BASIC interne	470
Champ d'application	470
Entrée	470
Sortie	470
Exemple	471
Minute()	471
Syntaxe BASIC interne	471
Champ d'application	471
Entrée	471
Sortie	472
Exemple	472
MkDir()	472
Syntaxe BASIC interne	472
Champ d'application	472
Entrée	473
Sortie	473
Month()	473
Syntaxe BASIC interne	473
Champ d'application	473
Entrée	473
Sortie	474
Exemple	474
Name()	474

Syntaxe BASIC interne	474
Champ d'application	474
Entrée	475
Sortie	475
Now()	475
Syntaxe BASIC interne	475
Champ d'application	475
Sortie	476
NPER()	476
Syntaxe BASIC interne	476
Champ d'application	476
Entrée	477
Sortie	477
Remarques	478
Oct()	478
Syntaxe BASIC interne	478
Champ d'application	478
Entrée	478
Sortie	478
Exemple	479
ParseDate()	479
Syntaxe BASIC interne	479
Champ d'application	479
Entrée	480
Sortie	480
Exemple	480
ParseDMYDate()	481
Syntaxe BASIC interne	481
Champ d'application	481
Entrée	481
Sortie	481
ParseMDYDate()	482
Syntaxe BASIC interne	482
Champ d'application	482
Entrée	482
Sortie	482
ParseYMDDate()	483
Syntaxe BASIC interne	483
Champ d'application	483
Entrée	483
Sortie	483
PMT()	484
Syntaxe BASIC interne	484
Champ d'application	484

Entrée	484
Sortie	485
Remarques	485
PPMT()	486
Syntaxe BASIC interne	486
Champ d'application	486
Entrée	486
Sortie	487
Remarques	487
PV()	488
Syntaxe BASIC interne	488
Champ d'application	488
Entrée	488
Sortie	489
Remarques	489
Randomize()	490
Syntaxe BASIC interne	490
Champ d'application	490
Entrée	490
Sortie	490
Exemple	491
RATE()	491
Syntaxe BASIC interne	491
Champ d'application	491
Entrée	491
Sortie	492
Remarques	493
RemoveRows()	493
Syntaxe BASIC interne	493
Champ d'application	493
Entrée	494
Sortie	494
Exemple	494
Replace()	494
Syntaxe BASIC interne	495
Champ d'application	495
Entrée	495
Sortie	495
Exemple	496
Right()	496
Syntaxe BASIC interne	496
Champ d'application	496
Entrée	497
Sortie	497

Exemple	497
RightPart()	497
Syntaxe BASIC interne	498
Champ d'application	498
Entrée	498
Sortie	498
Exemple	499
RightPartFromLeft()	499
Syntaxe BASIC interne	499
Champ d'application	499
Entrée	500
Sortie	500
Exemple	500
Rmdir()	501
Syntaxe BASIC interne	501
Champ d'application	501
Entrée	501
Sortie	501
Exemple	502
Rnd()	502
Syntaxe BASIC interne	502
Champ d'application	502
Entrée	502
Sortie	502
Remarques	503
Exemple	503
RTrim()	503
Syntaxe BASIC interne	503
Champ d'application	503
Entrée	504
Sortie	504
Exemple	504
Second()	505
Syntaxe BASIC interne	505
Champ d'application	505
Entrée	505
Sortie	506
Exemple	506
SetSubList()	506
Syntaxe BASIC interne	506
Champ d'application	506
Entrée	507
Sortie	507
Exemple	508
Sgn()	508

Syntaxe BASIC interne	508
Champ d'application	508
Entrée	509
Sortie	509
Exemple	509
Shell()	509
Syntaxe BASIC interne	509
Champ d'application	510
Entrée	510
Sortie	510
Exemple	510
Sin()	510
Syntaxe BASIC interne	511
Champ d'application	511
Entrée	511
Sortie	511
Exemple	511
Space()	512
Syntaxe BASIC interne	512
Champ d'application	512
Entrée	512
Sortie	512
Remarques	513
Exemple	513
Sqr()	513
Syntaxe BASIC interne	513
Champ d'application	513
Entrée	514
Sortie	514
Exemple	514
Str()	514
Syntaxe BASIC interne	514
Champ d'application	514
Entrée	515
Sortie	515
Exemple	515
StrComp()	515
Syntaxe BASIC interne	515
Champ d'application	516
Entrée	516
Sortie	516
String()	516
Syntaxe BASIC interne	516
Champ d'application	517

Entrée	517
Sortie	517
Exemple	517
SubList()	518
Syntaxe BASIC interne	518
Champ d'application	518
Entrée	518
Sortie	519
Exemple	519
Tan()	520
Syntaxe BASIC interne	520
Champ d'application	520
Entrée	520
Sortie	521
Exemple	521
Time()	521
Syntaxe BASIC interne	521
Champ d'application	521
Sortie	522
Timer()	522
Syntaxe BASIC interne	522
Champ d'application	522
Sortie	522
TimeSerial()	523
Syntaxe BASIC interne	523
Champ d'application	523
Entrée	523
Sortie	523
Exemple	524
TimeValue()	524
Syntaxe BASIC interne	525
Champ d'application	525
Entrée	525
Sortie	525
Exemple	525
ToSmart()	526
Syntaxe BASIC interne	526
Champ d'application	526
Entrée	526
Sortie	526
Trim()	527
Syntaxe BASIC interne	527
Champ d'application	527
Entrée	527

Sortie	527
Exemple	528
UCase()	528
Syntaxe BASIC interne	528
Champ d'application	529
Entrée	529
Sortie	529
Exemple	529
UnEscapeSeparators()	530
Syntaxe BASIC interne	530
Champ d'application	530
Entrée	531
Sortie	531
Exemple	531
Union()	531
Syntaxe BASIC interne	531
Champ d'application	532
Entrée	532
Sortie	532
Exemple	532
UTCToLocalDate()	533
Syntaxe BASIC interne	533
Champ d'application	533
Entrée	533
Sortie	533
Val()	534
Syntaxe BASIC interne	534
Champ d'application	534
Entrée	534
Sortie	534
Exemple	535
WeekDay()	535
Syntaxe BASIC interne	535
Champ d'application	535
Entrée	535
Sortie	536
Exemple	536
Year()	536
Syntaxe BASIC interne	536
Champ d'application	536
Entrée	536
Sortie	537

IV. Index	539
Chapitre 9. Fonctions disponibles - Domaine : Tous	541
Chapitre 10. Fonctions disponibles - Domaine : Technique	555
Chapitre 11. Fonctions disponibles - Domaine : Achats	561
Chapitre 12. Fonctions disponibles - Domaine : Fonctionnel	563
Chapitre 13. Fonctions disponibles - Domaine : HelpDesk	565
Chapitre 14. Fonctions disponibles - Domaine : Refacturation	567
Chapitre 15. Fonctions disponibles - Domaine : Câble	569
Chapitre 16. Fonctions disponibles - Domaine : Actions	571
Chapitre 17. Fonctions disponibles - Domaine : Interface Utilisateur 	573
Chapitre 18. Fonctions disponibles - Domaine : Builtin	575
Chapitre 19. Fonctions disponibles - Domaine : Assistants	579

I. Introduction

Vous trouverez dans cette partie les informations suivantes :

- Classification des fonctions
- Conventions
- Définitions
- Typage des fonctions et des paramètres de fonctions

1 Classification des fonctions

CHAPITRE

La classification des fonctions s'effectue à trois niveaux différents. Une fonction donnée peut être classée par :

- Familles de fonctions
- Champs d'application des fonctions
- Domaines fonctionnels

Familles de fonctions

Les fonctions disponibles dans l'environnement AssetCenter peuvent être regroupées en plusieurs grandes familles :

- les fonctions reconnues par AssetCenter : il s'agit essentiellement des fonctions utilisables dans les parties scriptables (en BASIC) du logiciel.
- les fonctions reconnues par la bibliothèque AssetCenterAPI ; ces fonctions peuvent être appelées par des outils externes ou par un programme écrit dans un langage évolué.

Ces grandes familles de fonctions se recoupent. Par exemple, certaines fonctions des API AssetCenter sont utilisables dans les scripts BASIC du logiciel. On dit alors qu'une telle fonction, provenant à l'origine des API AssetCenter, est "exposée" dans les scripts BASIC internes à AssetCenter. Si la syntaxe d'une telle fonction peut alors varier sensiblement, son comportement reste inchangé.

Champs d'application des fonctions

Les fonctions décrites dans ce document sont utilisables dans au moins un des contextes suivants :

- Bibliothèques API AssetCenter. En particulier, les fonctions sont accessibles pour le développement d'applications Get-It.
- Script de configuration d'un champ ou d'un lien (menu contextuel **Configurer l'objet** ou AssetCenter Database Administrator) et par extension **Script de calcul** (Nom SQL : memScript) d'un champ calculé :
 - Valeur par défaut,
 - Obligation de saisie,
 - Historisation,
 - Lecture seule,
 - ...
- Actions de type Script :
 - Script défini dans le champ **Script de l'action** (Nom SQL : Script) d'une action Script.
- Assistants AssetCenter :
 - Script "FINISH.DO" d'un assistant.
 - Script de définition des valeurs des propriétés d'un noeud.

Domaines fonctionnels

Chaque fonction est associée à un domaine fonctionnel. Ce domaine fonctionnel décrit la nature des opérations effectuées par la fonction. Les domaines fonctionnels sont les suivants :

- Builtin : fonctions Basic classiques et fonctions de conversion, de manipulation de chaînes, etc.
- Techniques : connexion à une base de données, gestion des objets tables, champs, liens, index, enregistrements, requêtes.
- Fonctionnel : fonctions génériques, orientées métier.
- Câble.
- Achats.
- Support.
- Refacturation.
- Assistants.
- Actions.
- Graphiques.

2 Conventions

CHAPITRE

Ce chapitre décrit :

- Conventions d'écriture
- Format des constantes de type Date+Heure dans les scripts
- Format des constantes de type Durée


Conventions d'écriture

La syntaxe des fonctions et des exemples proposés respecte les conventions d'écriture suivantes :

- [] Exception : dans les scripts Basic, lorsque les crochets encadrent le chemin d'accès à des données de la base, ils doivent figurer dans le script, comme le montre l'exemple ci-dessous :
- [Lien.Lien.Champ]**
Ces crochets encadrent un paramètre optionnel. Ne les tapez pas dans votre commande.
-

<>	Ces crochets encadrent un paramètre décrit en langage clair. Ne les tapez pas dans votre commande et remplacez le texte qu'ils encadrent par l'information qui doit y figurer.
{}	Ces accolades encadrent des paramètres parmi lesquels un seul doit être choisi. Ne tapez pas les accolades dans votre commande.
	La barre verticale sépare les paramètres possibles qui figurent dans les accolades.
*	L'astérisque ajouté à droite de crochets indique que la formule qu'ils encadrent peut être répétée plusieurs fois.

Les mises en forme suivantes ont des significations particulières :

Police fixe	Commande DOS, paramètre de fonction ou formatage de données.
Exemple	Exemple de code ou de commande.
...	Portion de code ou de commande omise.
Nom d'objet	Les noms de champs, d'onglets, de menus, de fichiers sont en caractères gras.
 Note : Note	Note importante.

Format des constantes de type Date+Heure dans les scripts

Les dates référencées dans les scripts sont exprimées au format international, indépendamment des options d'affichage spécifiées par l'utilisateur :

yyyy/mm/dd hh:mm:ss

Exemple :

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



Note : Le tiret ("-") peut également être utilisé comme séparateur de date.

A propos des dates

Les dates sont exprimées différemment en Basic interne et à partir d'outils externes :

- En Basic, une date peut être exprimée au format international ou sous la forme d'un nombre à virgule flottante (type "Double"). Dans ce dernier cas, la partie entière de ce nombre représente le nombre de jours écoulés depuis le 30/12/1899 à 0:00, la partie décimale représente la fraction écoulée dans le jour courant.
- En externe, les dates sont exprimées sous la forme d'un entier long (type "Long" 32 bits) qui représente le nombre de secondes écoulées depuis le 01/01/1970 à 0:00, indépendamment d'un quelconque fuseau horaire (heure UTC).

Format des constantes de type Durée

Dans les scripts, les durées sont stockées et exprimées en secondes. Par exemple, pour fixer la valeur par défaut d'un champ de type "Durée" à 3 jours, vous devez utiliser le script suivant :

```
RetVal=259200
```

De même, les fonctions qui calculent une durée, comme la fonction "AmWorkTimeSpanBetween()", fournissent un résultat en secondes.



Note : Dans les calculs financiers, AssetCenter tient compte des simplifications habituellement usitées. Dans ce cas seulement, une année vaut 12 mois et un mois vaut 30 jours (d'où : 1 année = 360 jours).

3 Définitions

CHAPITRE

Ce chapitre regroupe les définitions de quelques termes essentiels. Vous y trouverez les définitions suivantes :

- Définition d'une fonction
- Définition du lien virtuel `CurrentUser`
- Définition d'un handle
- Définition d'un code d'erreur

Définition d'une fonction

Une fonction est un programme qui effectue des opérations et renvoie à l'utilisateur une valeur, appelée "valeur de retour" ou "code de retour". Voici un exemple de syntaxe d'appel d'une fonction par le BASIC interne de AssetCenter :

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

Voici à présent la syntaxe d'appel de la même fonction au travers des API AssetCenter :

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

Définition du lien virtuel CurrentUser

Définition

"CurrentUser" peut être considéré comme un lien qui part de toutes les tables et pointe vers l'enregistrement de la table des services et personnes correspondant à l'utilisateur courant.

- Sous la forme "CurrentUser", il pointe sur l'enregistrement correspondant à l'utilisateur courant et renvoie son numéro d'identifiant (son "Id").
- Sous la forme "CurrentUser.Champ", il renvoie la valeur du champ pour l'utilisateur courant.



Note : Ce lien virtuel n'est pas affiché dans la liste des champs et des liens et n'est donc pas accessible directement dans le constructeur de scripts interne à AssetCenter. Vous devez saisir cette expression à la main.

Equivalences

Les fonctions "AmLoginName()" et "AmLoginId()" qui fournissent respectivement le "Nom" (Nom SQL : Name) et le numéro d'identifiant (Nom SQL : lPersId) de l'utilisateur courant peuvent être considérées comme des fonctions dérivées de "CurrentUser". En effet, on a les équivalences suivantes :

- AmLoginName()=[CurrentUser.Name]

- `AmLoginId()=[CurrentUser.lPersId]`

Définition d'un handle

Un handle représente un identifiant unique sur un objet. Dans le contexte de AssetCenter, cet objet peut être un champ, un lien, un index, une requête, un enregistrement, une table ou une connexion. Les handles sont des entiers (type "Long") codés sur 32 bits.



Note : Un handle valide ne peut avoir une valeur nulle (NULL).

Définition d'un code d'erreur

Lorsque l'exécution d'une fonction échoue, un code d'erreur est renvoyé.

A partir d'outils externes

A partir d'outils externes, le code d'erreur et le message qui lui est associé peuvent être récupérés respectivement grâce aux fonctions "AmLastError()" et "AmLastErrorMsg()". Il peut être détruit au moyen de la fonction "AmClearLastError()".



Note : Tout nouvel appel de fonction efface le code d'erreur et le message précédents.

En interne

En interne (dans les scripts Basic par exemple), la description et le code de la dernière erreur peuvent être retrouvés respectivement au moyen des fonctions `Err.Number` et `Err.Description`.



Note : En interne, vous n'avez pas besoin de programmer une gestion des erreurs. Le script erroné s'arrête et un rollback est effectué, si nécessaire, sur la base de données.

Vous pouvez déclencher volontairement un message d'erreur en utilisant la fonction `Err.Raise` dont la syntaxe est la suivante :

```
Err.Raise (<Numéro de l'erreur>, <Message d'erreur>)
```



Note : Lorsque la création ou la modification d'un enregistrement est invalidé par la valeur du champ "Validité" pour la table concernée, il est judicieux de déclencher un message d'erreur au moyen de la fonction `Err.Raise` afin de prévenir l'utilisateur (codes 12006 ou 12007). Si vous ne le faites pas, celui-ci ne comprendra pas nécessairement pourquoi il ne peut ni modifier, ni créer l'enregistrement.

Le tableau ci-dessous répertorie les codes d'erreur les plus fréquents :

Code d'erreur	Signification
12001	Erreur indéfinie
12002	Mauvais paramètre pour une fonction
12003	Handle invalide ou objet détruit
12004	Plus de données disponible. Cette erreur arrive classiquement lors de l'exécution de requêtes. Quand le résultat de la requête ne

renvoie aucune donnée, cette erreur est déclenchée.

Code d'erreur	Signification
12005	Erreur interne du serveur de base de données
12006	Valeur non valide (type incorrect pour un paramètre, etc.)
12007	Enregistrement non valide (par exemple, un champ obligatoire n'a pas été renseigné)
12008	Problèmes de droits d'accès à des données de la base
12009	Fonction obsolète ou non implémentée
12010	Nombre maximum de connexions à la base de données dépassé

4 | Typage des fonctions et des paramètres de fonctions

CHAPITRE

Vous trouverez dans ce chapitre les informations suivantes :

- Liste des types
- Type d'une fonction
- Type d'un paramètre

Liste des types

Le tableau ci-dessous récapitule les différents types possibles pour une fonction ou un paramètre :

Type	Signification
Integer	Nombre entier de -32 768 à +32 767.
Long	Nombre entier de -2 147 483 647 à +2 147 483 646.
Double	Nombre à virgule flottante de 8 octets.
String	Texte pour lequel tous les caractères sont acceptés.
Date	Date ou Date+Heure.

Type	Signification
Variant	Type générique pouvant représenter n'importe quel type.



Note : Ces types ne sont pas tous disponibles à partir d'un outil externe. Seuls les types Long, Double et String sont disponibles. Le Variant n'existe pas et les objets de types Integer et Date sont représentés par un Long.

Type d'une fonction

Le type d'une fonction correspond au type de la valeur retournée par la fonction. Nous vous invitons à faire particulièrement attention à cette information car elle peut être à l'origine d'erreurs de compilation et d'exécution de vos programmes.

Par exemple, vous ne pouvez pas utiliser une fonction renvoyant une valeur d'un certain type dans la définition de la valeur par défaut d'un champ d'un type différent. Essayez par exemple d'affecter ce script de valeur par défaut à n'importe quel champ de type "Date" ou "Date+Heure" :

```
RetVal=AmLoginName( )
```

La fonction "AmLoginName()" renvoie le nom de l'utilisateur connecté sous la forme d'une chaîne de caractères (type "String"). Cette valeur de retour est donc dans un format incompatible avec celui d'un champ de type "Date" et AssetCenter affiche un message d'erreur.

Type d'un paramètre

Les paramètres utilisés dans les fonctions possèdent également un type que vous devez impérativement respecter pour la bonne exécution de la fonction. Dans la syntaxe des fonctions, les paramètres sont préfixés en fonction de leur type. Pour éviter toute confusion possible, les

préfixes utilisés dans cette référence sont différents suivant la syntaxe (API ou Basic) de la fonction. Le tableau ci-dessous propose pour chaque type une équivalence entre le préfixe utilisé dans la syntaxe API et celui utilisé dans la syntaxe Basic :

Type	Préfixe utilisé dans la syntaxe API	Préfixe utilisé dans la syntaxe Basic
Integer	"i"	"i"
Long	"h" pour un handle ou "l" pour un nombre	"l"
Double	"d"	"d"
String	"char*psz"	"str"
Date	"ltn"	"dt"
Variant	"v"	"v"

II. Utilisation des API

Ce chapitre propose un rapide tour d'horizon sur l'utilisation des API.
Il s'articule en trois grands points :

- Préambule
- Méthodologie
- Concepts et exemples

5 Préambule

CHAPITRE

Les API AssetCenter sont fournies sous la forme d'un fichier DLL 32 bits utilisable sous Windows 95/98, Windows NT ou Windows 2000.

Elles ont été testées avec succès dans des environnements de développement suivants :

- Visual Basic 4.0, 5.0 et 6.0,
- Visual C++ 4.0, 5.0 et 6.0,
- Tous les produits de la gamme Microsoft utilisant le VBA (Visual Basic for Applications)



Note : Les API sont à priori compatibles avec tous les outils autorisant l'utilisation de bibliothèques de fonctions externes (DLL).

Avertissement

L'utilisation des API AssetCenter est assujettie à une bonne connaissance du modèle conceptuel de données utilisé par AssetCenter en général et de la structure de la base de données en particulier.

Toutes les informations utiles sur la structure de la base de données sont regroupées dans le manuel intitulé "Manuel de référence : Administration et utilisation avancée", chapitre "Structure de la base de données AssetCenter" ainsi que dans les fichiers "database.txt" et "tables.txt", situés dans le sous-répertoire "infos" du répertoire d'installation de AssetCenter.

Installation

Avant toute utilisation des API, nous vous recommandons de procéder à une installation complète de AssetCenter. Vous pourrez ainsi contrôler facilement l'accès aux bases de données à partir de votre ordinateur, créer et configurer rapidement les bases de données sur lesquelles vous souhaitez travailler. Les API utilisant les mêmes couches de base de données et les mêmes informations de configuration que AssetCenter pour l'accès aux sources de données, vous pouvez rapidement détecter, à partir de l'interface graphique de AssetCenter, les problèmes rencontrés lors de l'utilisation des API.

Les étapes classiques pour l'installation d'un environnement de développement sous AssetCenter sont les suivantes :

- Installation d'une version 32 bits de AssetCenter avec le composant AssetCenter API.
- Configuration de la source de données et test d'accès à la base de données sous AssetCenter.
- Utilisation de votre environnement de développement pour appeler les fonctions des API AssetCenter.

Nous vous recommandons de vous familiariser avec les API AssetCenter en utilisant une base de données de démonstration ou toute autre source ne contenant aucune donnée sensible.

6 Méthodologie

CHAPITRE

Voici, étape par étape, une approche classique de l'utilisation des API AssetCenter :

1. Création d'une requête AQL du type :

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



Note : Une bonne solution pour composer simplement une requête AQL est d'utiliser AssetCenter Export.

2. Récupération des résultats de la requête et des handles sur tous les objets que vous souhaitez utiliser.
3. Utilisation des ces handles pour mettre à jour les informations contenues dans les objets correspondants.

4. Réalisation d'un "Commit" (pour accepter toutes les modifications) ou d'un "Rollback" (pour annuler toutes les modifications) pour la transaction en cours.

7 Concepts et exemples

CHAPITRE

Vous trouverez dans cette partie les informations suivantes :

- Concepts
- Manipuler les dates
- Premier exemple
- Second exemple

Concepts

AssetCenter a été pensé et conçu dans une approche "objet" qui se retrouve également dans les API. Les DLLs de Windows requièrent l'utilisation d'un modèle "flat" proche du C. Les API AssetCenter contournent cette limitation en utilisant des "handles" sur les objets créés par l'utilisateur. Cette approche permet aux langages qui ne sont pas orientés "objet" d'accéder au modèle de AssetCenter.

Avant tout autre chose, votre programme doit utiliser la fonction "AmStartup()" pour initialiser l'appel aux bibliothèques AssetCenter.

De même, la dernière fonction appelée par votre programme doit impérativement être la fonction "AmCleanUp()".

Avant tout accès à un objet d'une base, une connexion valide doit être établie entre l'utilisateur et cette base de données. Cette connexion est identifiée par un "handle" sur un objet "connexion" (ce handle est alors utilisé dans toutes les fonctions des APIs qui interagissent avec la base de données. Il correspond au paramètre "hApiCnxBase"). Cet objet peut alors être utilisé pour créer des requêtes et accéder aux enregistrements.



Note : Notez que tous les objets d'une base de données sont liés à une connexion. Il en résulte que les informations sur les droits d'accès par exemple peuvent être contrôlées.

La première étape consiste donc en la création d'une connexion utilisant une source de données, un login et son mot de passe valides.



Attention : Attention : lorsque vous vous connectez à une base de données AssetCenter au travers des APIs, un jeton de connexion est utilisé.

Manipuler les dates

Pour lire une date, vous avez la possibilité d'utiliser l'une des deux fonctions suivantes sur un champ de type "Date" ou "Date+Heure" :

- "AmGetFieldLongValue()" qui renvoie la date sous la forme d'un "Long" Unix (UTC). Préférez cette fonction pour les calculs faisant intervenir les dates.
- "AmGetFieldStrValue()" qui renvoie la date sous la même forme que le "control panel" de Windows. Cette date respecte les fuseaux horaires. Utilisez cette fonction pour l'affichage.

Premier exemple

L'exemple suivant, rédigé en C, déclare une connexion à la base de démonstration :

```
long lCnx ;
lCnx = AmOpenConnection(ACDemo351FRA, Admin , ) ;
```

"lCnx" représente un "handle" sur un objet "connexion". Ce handle est utilisé pour identifier la connexion que vous venez de déclarer.

Cette connexion peut dès lors être utilisée pour créer des requêtes et accéder à la base de données. L'exemple suivant, rédigé en C, définit une requête sur la table des biens et parcourt les résultats de cette requête :

```
#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'")
        /* Navigates through the result set */
```

```

        while( !lStatus )
        {
            /* Read the first field (AssetTag) of the current
            item in the query */
            lStatus =
            AmGetFieldStrValue(lQuery,0,szModel,SZ_MODEL_LEN-1);
            if( lStatus == 0 )
            {
                printf(' Compaq AssetTag=%s\n',szModel);
                lStatus = AmQueryNext(lQuery);
            }
        }
        /* clean things up */
        AmReleaseHandle(lQuery);
    }
    AmCloseConnection(lCnx);
}
AmCleanup();

```

Second exemple

Les requêtes sont utilisées pour localiser les objets dans la base de données. Lorsque vous devez mettre à jour un enregistrement, un handle sur l'objet "enregistrement" doit être récupéré au moyen d'une requête. L'enregistrement peut alors être traité au moyen des autres fonctions des APIs AssetCenter.

L'exemple suivant illustre la modification d'un champ d'un enregistrement donné :

```

/* 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();

```

Cet exemple illustre la récupération d'un "handle" sur un enregistrement par le biais d'une requête. La requête est alors analysée pour localiser un élément de l'enregistrement (dans cet exemple un champ). Il est également possible d'utiliser la fonction "AmQueryExec()" pour récupérer plusieurs enregistrements en une seule requête, puis de parcourir ces enregistrements et récupérer les "handle" des éléments qui vous intéressent.



Note : Par souci de simplification, cet exemple ne traite pas tous les codes d'erreur possibles.

III. Référence alphabétique

8 Référence alphabétique

CHAPITRE

Abs()

Renvoie la valeur absolue d'un nombre.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la valeur absolue.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

AmActionDde()

Cette fonction lance une requête DDE à destination d'une application qui gère les liens DDE. Grâce à cette fonction, AssetCenter peut piloter une autre application par l'intermédiaire d'un lien DDE. Cette fonction est équivalente à une action de type DDE.

Syntaxe API





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

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strService* : Ce paramètre contient le nom du service DDE proposé par l'exécutable que vous souhaitez solliciter. Veuillez vous reporter à la documentation de cet exécutable pour connaître la liste des services DDE qu'il propose.
- *strTopic* : Ce paramètre contient le thème, c'est-à-dire le contexte dans lequel l'action DDE doit être effectuée.
- *strCommand* : Ce paramètre contient les commandes que l'application externe doit exécuter. Vous devez respecter la syntaxe imposée par l'application externe.
- *strFileName* : Si le service n'est pas présent en mémoire, vous devez le charger en précisant dans ce paramètre le nom de l'exécutable (ou celui d'un fichier quelconque si celui-ci est associé à un exécutable par l'intermédiaire du gestionnaire de fichiers de Windows) qui active le service.

- *strDirectory* : Ce paramètre contient le chemin d'accès du fichier précisé dans *strFileName*.
- *strParameters* : Ce paramètre contient les différents paramètres à fournir à l'exécutable qui active le service lors de son lancement.
- *strTable* : Paramètre optionnel, utilisé dans le cas où l'action est contextuelle. Il indique le nom SQL de la table contenant l'enregistrement auquel s'applique l'action.
- *lRecordId* : Paramètre optionnel, utilisé dans le cas où l'action est contextuelle. Il indique l'identifiant de l'enregistrement auquel s'applique l'action.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmActionExec()

Cette fonction lance une application de type ".exe", ".com", ".bat", ".pif". Vous pouvez également faire référence à des documents de tous types, à condition que leur extension soit associée à un exécutable par le gestionnaire de fichiers de Windows. Cette fonction est équivalente à une action de type "Exécutable".

Syntaxe API





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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strFileName* : Ce paramètre contient le nom de l'exécutable ou celui d'un document quelconque (associé à un exécutable par l'intermédiaire du gestionnaire de fichiers de Windows).
- *strDirectory* : Ce paramètre contient le chemin d'accès du fichier précisé dans le paramètre *strFileName*.
- *strParameters* : Ce paramètre optionnel contient les différents paramètres à fournir à l'exécutable lors de son lancement.
- *strTable* : Paramètre optionnel, utilisé dans le cas où l'action est contextuelle. Il indique le nom SQL de la table contenant l'enregistrement auquel s'applique l'action.
- *lRecordId* : Paramètre optionnel, utilisé dans le cas où l'action est contextuelle. Il indique l'identifiant de l'enregistrement auquel s'applique l'action.

Sortie

- 0 : La fonction s'est exécutée normalement.

- Non nul : Code d'erreur.

Exemple

Cet exemple exécute l'explorer de Windows NT (situé dans le dossier "WinNT" du disque "C") :

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

AmActionMail()

Cette fonction émet un message via l'un des messageries gérées par AssetCenter :

- Messagerie interne.
- Messagerie externe au standard VIM (Lotus Notes, ...).
- Messagerie externe au standard MAPI (Microsoft Exchange, Microsoft Outlook, ...).
- Messagerie externe au standard SMTP (standard Internet).

Syntaxe API





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

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTo* : Ce paramètre contient la liste des adresses des destinataires du message, sous la forme `messaging:adresse`. Le point-virgule est utilisée comme séparateur.
- *strCc* : Ce paramètre contient la liste des adresses des destinataires en copie du message. Le point-virgule est utilisée comme séparateur.
- *strCcc* : Ce paramètre contient la liste des adresses des destinataires en copie cachée du message (ils n'apparaissent pas dans la liste des destinataires). Le point-virgule est utilisée comme séparateur.
- *strSubject* : Ce paramètre contient l'intitulé du message.
- *strMessage* : Ce paramètre contient le corps du message.
- *iPriority* : Ce paramètre définit la priorité d'envoi du message
 - 0 priorité basse.
 - 1 priorité normale.
 - 2 priorité haute.
- *bAcknowledge* : Ce paramètre précise si l'émetteur du message reçoit un accusé de réception :
 - 0 : l'émetteur ne reçoit pas d'accusé de réception.
 - 1 : l'émetteur reçoit un accusé de réception.
- *strRefObject* : Ce paramètre ne sert qu'aux messages adressés à la messagerie interne d'AssetCenter. Il s'agit du nom SQL du lien qu'il faut suivre depuis l'enregistrement correspondant au contexte

d'exécution pour atteindre l'objet référencé. Il peut s'agir du lien virtuel `CurrentUser`.

- *strTable* : Paramètre optionnel, utilisé dans le cas où l'action est contextuelle. Il indique le nom SQL de la table contenant l'enregistrement auquel s'applique l'action.
- *lRecordId* : Paramètre optionnel, utilisé dans le cas où l'action est contextuelle. Il indique l'identifiant de l'enregistrement auquel s'applique l'action.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmActionPrint()

Cette fonction déclenche l'impression d'un rapport sur un enregistrement donné de la base.

Syntaxe API

```
long AmActionPrint(long lReportId, long lRecordId);
```

Syntaxe BASIC interne

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

Champ d'application


Version : 3.00

Utilisable

AssetCenter APIs



Script de configuration d'un champ ou d'un lien

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReportId* : Ce paramètre contient l'identifiant du rapport à imprimer.
- *lRecordId* : Ce paramètre contient l'identifiant de l'enregistrement concerné par le rapport. Par défaut, ce paramètre prend la valeur "0". La table concernée est implicitement définie par le rapport.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmActionPrintPreview()

Cette fonction déclenche un aperçu avant impression d'un rapport sur un enregistrement donné de la base.



Syntaxe BASIC interne

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

Champ d'application

Version : 3.60

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReportId* : Ce paramètre contient l'identifiant du rapport concerné.
- *lRecordId* : Ce paramètre contient l'identifiant de l'enregistrement concerné par le rapport. Par défaut, ce paramètre prend la valeur "0".

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmActionPrintTo()

Cette fonction déclenche l'impression d'un rapport sur un enregistrement donnée de la base et sur une imprimante donnée.

Syntaxe API





```
long AmActionPrintTo(char *strPrinterName, long lReportId, long lRecordId);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strPrinterName* : Ce paramètre contient le nom de l'imprimante sur laquelle s'effectue l'impression.
- *lReportId* : Ce paramètre contient l'identifiant du rapport à imprimer.
- *lRecordId* : Ce paramètre contient l'identifiant de l'enregistrement concerné par le rapport. Par défaut, ce paramètre prend la valeur "0".

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddAllPOLinesToInv()

Cette fonction ajoute l'intégralité d'une commande à une facture fournisseur existante.

Syntaxe API




```
long AmAddAllPOLinesToInv(long hApiCnxBase, long lPordId, long lInvId);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrdId* : Ce paramètre contient l'identifiant de commande à ajouter à la facture fournisseur.
- *lInvId* : Ce paramètre contient l'identifiant de la facture fournisseur à laquelle est ajoutée la commande.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddCatRefAndCompositionToPOrder()

Cette fonction permet d'ajouter le contenu complet d'une référence catalogue à une commande donnée.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrderId* : Ce paramètre contient l'identifiant de la commande à compléter.
- *lCatRefId* : Ce paramètre contient l'identifiant de la référence catalogue.
- *fCatRefQty* : Ce paramètre contient la quantité (dans l'unité associée au produit) à ajouter.
- *lRequestId* : Ce paramètre contient l'identifiant de la demande que cette commande va satisfaire.

- *dUnitPrice* : Ce paramètre contient le prix unitaire du produit de la référence catalogue.
- *strCur* : Ce paramètre contient le code de la devise dans laquelle le prix unitaire est exprimé

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Remarques



Note : Cette fonction permet notamment d'utiliser la composition de produits d'une référence catalogue pour enrichir une commande.

AmAddCatRefToPOrder()

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	
Script FINISH.DO d'un assistant	✓

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmAddEstimLinesToPO()

Cette fonction ajoute toutes les lignes de devis d'un devis à une commande existante.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	
Script FINISH.DO d'un assistant	✓

Entrée

- *lEstimId* : Ce paramètre contient l'identifiant du devis à ajouter à la commande.
- *lPOrdId* : Ce paramètre contient l'identifiant de la commande à laquelle sont ajoutées toutes les lignes de devis du devis.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddEstimLineToPO()

Cette fonction ajoute une ligne de devis à une commande existante.

Syntaxe API




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


Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lEstimLineId* : Ce paramètre contient l'identifiant de la ligne de devis à ajouter à la commande.
- *lPOrdId* : Ce paramètre contient l'identifiant de la commande à laquelle est ajoutée la ligne de devis.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddPOLineToInv()

Cette fonction ajoute une quantité donnée d'élément(s) sur une ligne de commande à une facture fournisseur.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrdLineId* : Ce paramètre contient l'identifiant de la ligne de commande à ajouter à la facture fournisseur.
- *lInvId* : Ce paramètre contient l'identifiant de la facture fournisseur à laquelle des éléments de la ligne de commande sont ajoutés.
- *fQty* : Ce paramètre contient la quantité d'éléments présents sur la ligne de commande à ajouter à la facture fournisseur.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmAddPOrderLineToReceipt()

Cette fonction permet d'ajouter une ligne de commande à une réception. Vous pouvez ainsi réceptionner une ligne de commande au sein d'une réception existante.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application



Version : 4.00

Utilisable

AssetCenter APIs



Script de configuration d'un champ ou d'un lien

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrderLineId* : Ce paramètre contient l'identifiant de la ligne de commande.
- *lRecptId* : Ce paramètre contient l'identifiant de la réception impactée.
- *fQty* : Ce paramètre contient la quantité à réceptionner. Vous pouvez ainsi limiter la quantité réceptionnée par rapport à la quantité commandée (dans l'unité du produit).
- *bCanMerge* : Ce paramètre permet de préciser si oui ou non la ligne peut être fusionnée avec une ligne déjà existante dans la réception.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmAddReceiptLineToInvoice()

Cette fonction permet d'ajouter une ligne de réception à une facture. Vous pouvez ainsi facturer une ligne de réception au sein d'une facture existante.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lRecptLineId* : Ce paramètre contient l'identifiant de la ligne de réception.
- *lInvoiceId* : Ce paramètre contient l'identifiant de la facture impactée.
- *fQty* : Ce paramètre contient la quantité à facturer. Vous pouvez ainsi limiter la quantité facturée par rapport à la quantité reçue (dans l'unité du produit).
- *bCanMerge* : Ce paramètre permet de préciser si oui ou non la ligne peut être fusionnée avec une ligne déjà existante dans la facture.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmAddReqLinesToEstim()

Cette fonction ajoute toutes les lignes de demande d'une demande à un devis existant.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqId* : Ce paramètre contient l'identifiant de la demande à ajouter au devis.
- *lEstimId* : Ce paramètre contient l'identifiant du devis auquel sont ajoutées toutes les lignes de demande de la demande.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddReqLinesToPO()

Cette fonction ajoute toutes les lignes de demande d'une demande à une commande existante. Le fournisseur précisé dans la demande doit être identique à celui de la commande concernée.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqId* : Ce paramètre contient l'identifiant de la demande à ajouter à la commande.
- *lPOrdId* : Ce paramètre contient l'identifiant de la commande à laquelle sont ajoutées les lignes de demande de la demande.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddReqLineToEstim()

Cette fonction ajoute une ligne de demande à un devis existant.

Syntaxe API




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


Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqLineId* : Ce paramètre contient l'identifiant de la ligne de demande à ajouter au devis.
- *lEstimId* : Ce paramètre contient l'identifiant du devis auquel est ajoutée la ligne de demande.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddReqLineToPO()

Cette fonction ajoute une ligne de demande à une commande existante.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqLineId* : Ce paramètre contient l'identifiant de la ligne de demande à ajouter à la commande.
- *lPOrdId* : Ce paramètre contient l'identifiant de la commande à laquelle est ajoutée la ligne de demande.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddRequestLineToPOrder()

Cette fonction permet d'ajouter une ligne de demande à une commande.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lRequestLineId* : Ce paramètre contient l'identifiant de la ligne de demande.
- *lPOrderId* : Ce paramètre contient l'identifiant de la commande impactée.
- *fQty* : Ce paramètre contient la quantité à commander. Vous pouvez ainsi limiter la quantité commandée par rapport à la quantité demandée (dans l'unité du modèle).

- *bCanMerge* : Ce paramètre permet de préciser si oui ou non la ligne peut être fusionnée avec une ligne déjà existante dans la commande.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmAddTemplateLineToPOrder()

Syntaxe API

```
long AmAddTemplateLineToPOrder(long hApiCnxBase,
long lRequestLineId, long lPOrderId, long
lTemplLineId, long lQty, long bCanMerge);
```

Syntaxe BASIC interne

```
Function AmAddTemplateLineToPOrder(lRequestLineId
As Long, lPOrderId As Long, lTemplLineId As Long,
lQty As Long, bCanMerge As Long) As Long
```

Champ d'application

Version : 4.00

Utilisable

AssetCenter APIs



Script de configuration d'un champ ou d'un lien

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddTemplateToPOrder()

Cette fonction permet d'ajouter le contenu complet d'une commande standard à une commande donnée.

Syntaxe API



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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Entrée**

- *lRequestId* : Ce paramètre contient l'identifiant de la ligne de demande à satisfaire par les lignes de commandes qui seront ajoutées.
- *lPOrderId* : Ce paramètre contient l'identifiant de la commande impactée.
- *lTemplateId* : Ce paramètre contient l'identifiant de la commande standard à ajouter.
- *lQty* : Ce paramètre contient la quantité (dans l'unité du produit) à ajouter.
- *bCanMerge* : Ce paramètre permet de préciser si oui ou non la ligne peut être fusionnée avec une ligne déjà existante dans la commande.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmAddTemplateToRequest()

Cette fonction permet d'ajouter le contenu complet d'une demande standard à une demande donnée.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqId* : Ce paramètre contient l'identifiant de la ligne de demande impactée.
- *lTemplateId* : Ce paramètre contient l'identifiant de la demande standard à ajouter.
- *lQty* : Ce paramètre contient la quantité (dans l'unité du produit) à ajouter.
- *bCanMerge* : Ce paramètre permet de préciser si oui ou non la ligne peut être fusionnée avec une ligne déjà existante dans la demande.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmBusinessSecondsInDay()

Calcule le nombre de secondes ouvrées dans une journée en fonction d'un calendrier.

Syntaxe API






```
long AmBusinessSecondsInDay(char
*strCalendarSqlName, long tmDate);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strCalendarSqlName* : Nom SQL du calendrier utilisé pour le calcul.
- *tmDate* : Date à laquelle s'effectue le calcul.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCalcConsolidatedFeature()

Calcule la valeur d'une caractéristique consolidée sur une table identifiée par son nom SQL.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCalcFeatId* : Identifiant de la caractéristique consolidée.
- *strSQLTableName* : Nom SQL de la table pour laquelle la caractéristique consolidée est calculée. La caractéristique doit absolument être définie pour cette table.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCalcDepr()

Cette fonction permet de calculer le montant de l'amortissement sur un bien à une date donnée. Elle renvoie la valeur de l'amortissement à cette date.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs



	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *iType* : Ce paramètre permet d'identifier la nature de l'amortissement. Les valeurs possibles de ce paramètre sont les suivantes :
 - 0 : pas d'amortissement
 - 1 : amortissement linéaire
 - 2 : amortissement dégressif
- *lDuration* : Ce paramètre contient la durée sur laquelle porte l'amortissement du bien. Cette durée est exprimée en secondes.
- *dCoeff* : Ce paramètre contient le coefficient appliqué lors du calcul de l'amortissement dégressif. Il n'est pas interprété dans le cas d'un amortissement linéaire mais doit posséder une valeur quelconque.
- *dPrice* : Ce paramètre contient la valeur initiale du bien sur lequel porte le calcul de l'amortissement.
- *tmStart* : Ce paramètre contient la date à partir de laquelle le bien est amorti.
- *tmDate* : Ce paramètre contient la date à laquelle sont évalués l'amortissement et la valeur résiduelle du bien.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

AmCbkJReplayEvent()

Cette fonction permet de rejouer la règle de refacturation à l'origine d'un événement, après avoir corrigé l'enregistrement à l'origine de l'événement.

Syntaxe API




```
long AmCbkJReplayEvent(long hApiCnxBase, long
lCbkJEventId);
```

Syntaxe BASIC interne

```
Function AmCbkJReplayEvent(lCbkJEventId As Long) As
Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCbkJEventId* : Ce paramètre contient l'identifiant de l'événement de refacturation concerné.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCheckTraceDone()

L'API AmCheckTraceDone détermine si un port (lPortId) ou un faisceau (lBundleId) est connecté à une chaîne de liaisons existante. La direction de la chaîne de liaisons (iTraceDir) indique si la chaîne de liaisons doit être vérifiée suivant la direction utilisateur vers hôte (iTraceDir = 1) ou hôte vers utilisateur (iTraceDir = 0).

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPortId* : ce paramètre est l'identifiant du port à vérifier.
- *lBundleId* : ce paramètre est l'identifiant du faisceau à vérifier.
- *iTraceDir* : ce paramètre précise la direction à vérifier.
 - 1 : Vérifier en direction de l'hôte
 - 0 : Vérifier en direction de l'utilisateur

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCleanup()

Cette fonction doit être appelée à la fin de tout script utilisant les fonctions de modification de la base de données. Elle libère toutes les ressources utilisées.

Syntaxe API

```
void AmCleanup();
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	

Utilisable

Script de configuration d'un champ ou d'un lien

Action de type "Script"

Script d'un assistant

Script FINISH.DO d'un assistant

AmClearLastError()

Cette fonction efface les informations concernant le dernier message d'erreur survenu lors du dernier appel à une fonction.

Syntaxe API

```
long AmClearLastError(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmClearLastError() As Long
```

Champ d'application

Version : 2.52

Utilisable

AssetCenter APIs

Script de configuration d'un champ ou d'un lien

Action de type "Script"

Script d'un assistant

Script FINISH.DO d'un assistant

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCloseAllChildren()

Cette fonction détruit tous les objets créés lors de la connexion courante.

Syntaxe API






```
long AmCloseAllChildren(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmCloseAllChildren() As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCloseConnection()

Met un terme à la session AssetCenter pour une connexion donnée. Tous les objets (requêtes, enregistrements, tables, champs...) créés au cours de la session sont automatiquement détruits et tous leurs handles deviennent obsolètes et sont inutilisables. Le handle de la connexion n'existe plus.

Syntaxe API

```
long AmCloseConnection(long hApiCnxBase);
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCommit()

Cette fonction réalise un "Commit" de toutes les modifications apportées à la base de données associée à la connexion.

Syntaxe API




```
long AmCommit(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmCommit() As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmComputeAllLicAndInstallCounts()

Cette fonction effectue le décompte des licences et des installations logicielles pour tous les enregistrements.

Syntaxe API



```
long AmComputeAllLicAndInstallCounts(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmComputeAllLicAndInstallCounts() As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Sortie**

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmComputeLicAndInstallCounts()

Cette fonction effectue le décompte des licences et des installations logicielles pour un enregistrement.

Syntaxe API

```
long AmComputeLicAndInstallCounts(long hApiCnxBase,
long lSLCountId);
```

Syntaxe BASIC interne

```
Function AmComputeLicAndInstallCounts(lSLCountId
As Long) As Long
```

Champ d'application**Version : 3.00****Utilisable****AssetCenter APIs****Script de configuration d'un champ ou d'un lien****Action de type "Script"****Script d'un assistant****Script FINISH.DO d'un assistant**

Entrée

- *lSLCountId* : Ce paramètre contient l'identifiant du compteur de licences logicielles.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmConnectTrace()

L'API AmConnectTrace permet de connecter un dispositif/câble source à un dispositif/câble destination et de créer un historique de chaîne de liaisons ainsi qu'une opération sur chaîne de liaisons.

Syntaxe API




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

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *iSrcLinkType* : ce paramètre détermine le type de chaîne de liaisons pour le dispositif/câble source.
 - 8 : Câble
 - 9 : Dispositif
- *lSrcPortBunId* : ce paramètre est l'identifiant du port ou faisceau à connecter côté source.
- *lSrcLabelRuleId* : ce paramètre est l'identifiant de la règle d'étiquetage utilisée pour la liaison source.
- *iDestLinkType* : ce paramètre détermine le type de chaîne de liaisons pour le dispositif/câble destination.
 - 8 : Câble
 - 9 : Dispositif
- *lDestPortBunId* : ce paramètre est l'identifiant du port ou faisceau à connecter côté destination.
- *lDestLabelRuleId* : ce paramètre est l'identifiant de la règle d'étiquetage utilisée pour la liaison destination.
- *iTraceDir* : ce paramètre indique la direction de la connexion.
 - 1 : de l'utilisateur vers l'hôte
 - 0 : de l'hôte vers l'utilisateur
- *lDutyId* : ce paramètre est l'identifiant de la fonction de la liaison de type câble.

- *strComment* : ce paramètre est l'étiquette de l'opération sur chaîne de liaisons.
- *lCabTraceOutId* : ce paramètre est l'identifiant de compte-rendu de chaîne de liaisons de câble.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertCurrency()

Cette fonction effectue la conversion entre deux devises à une date et avec une précision données.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *tmDate* : Ce paramètre contient la date de conversion. Elle permet de connaître le taux de conversion effectif à cette date.
- *strSrcName* : Ce paramètre contient la devise source de la conversion, c'est-à-dire celle que vous souhaitez convertir.
- *strDstName* : Ce paramètre contient la devise de destination de la conversion, c'est-à-dire celle dans laquelle sera exprimée la devise source.
- *dVal* : Ce paramètre contient le montant (exprimé dans l'unité monétaire de la devise source) à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les devises utilisées comme paramètres pour la fonction (`strSrcName` et `strDstName`) doivent impérativement être définies

sous AssetCenter. De même un taux de change valide doit exister à la date à laquelle s'effectue la conversion (paramètre tmDate).

Exemple

L'exemple suivant effectue la conversion de 5000 FRF en dollars, à la date du 02/11/98.

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

AmConvertDateBasicToUnix()

Cette fonction convertit une date au format Basic (type "Date") en une date au format Unix (type "Long"). Cette fonction est inopérante à partir d'un outil externe car les deux types sont alors équivalents.

Syntaxe API




```
long AmConvertDateBasicToUnix(long hApiCnxBase,
long tmTime);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *tmTime* : Ce paramètre contient la date à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertDateIntlToUnix()

Cette fonction convertit une date au format international (type "Date") en une date au format Unix (type "Long").

Syntaxe API






```
long AmConvertDateIntlToUnix(long hApiCnxBase, char
*strDate);
```

Syntaxe BASIC interne

```
Function AmConvertDateIntlToUnix(strDate As String)
As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strDate* : Ce paramètre contient la date à convertir, au format international (yyyy-mm-dd hh:mm:ss).

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertDateStringToUnix()

Convertit une date au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows) en un "Long" Unix.

Syntaxe API

```
long AmConvertDateStringToUnix(long hApiCnxBase,
char *strDate);
```

Syntaxe BASIC interne

```
Function AmConvertDateStringToUnix(strDate As
String) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strDate* : Date au format chaîne à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertDateUnixToBasic()

Cette fonction convertit une date au format Unix (type "Long") en une date au format Basic (type "Date"). Cette fonction est inopérante à partir d'un outil externe car les deux types sont alors équivalents.

Syntaxe API






```
long AmConvertDateUnixToBasic(long hApiCnxBase,
long lTime);
```

Syntaxe BASIC interne

```
Function AmConvertDateUnixToBasic(lTime As Long)
As Date
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lTime* : Ce paramètre contient la date à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertDateUnixToIntl()

Cette fonction convertit une date au format Unix (type "Long") en une date au format international (yyyy-mm-dd hh:mm:ss).

Syntaxe API






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

Syntaxe BASIC interne

```
Function AmConvertDateUnixToIntl(lUnixDate As Long)
As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lUnixDate* : Ce paramètre contient la date à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertDateUnixToString()

Convertit une date au format "Long" Unix en une date au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows).

Syntaxe API






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

Syntaxe BASIC interne

```
Function AmConvertDateUnixToString(lUnixDate As
Long) As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lUnixDate* : Date au format "Long" Unix à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertDoubleToString()

Cette fonction convertit un nombre en double précision en une chaîne de caractères. La chaîne est formatée conformément aux options régionales (de nombre) définies dans le panneau de contrôle de Windows.

Syntaxe API





```
long AmConvertDoubleToString(double dSrc, char
*ptrDst, long lDst);
```

Syntaxe BASIC interne

```
Function AmConvertDoubleToString(dSrc As Double)
As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Entrée**

- *dSrc* : Ce paramètre contient le nombre en double précision à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertMonetaryToString()

Cette fonction convertit une valeur monétaire en une chaîne de caractères. La chaîne est formatée conformément aux options régionales (monétaires) définies dans le panneau de contrôle de Windows.

Syntaxe API






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

Syntaxe BASIC interne

```
Function AmConvertMonetaryToString(dSrc As Double)  
As String
```


Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dSrc* : Ce paramètre contient la valeur monétaire à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertStringToDouble()

Cette fonction convertit une chaîne de caractères (dans un format conforme à celui défini dans le panneau de contrôle de Windows) en un nombre en double précision.

Syntaxe API






```
double AmConvertStringToDouble(char *strSrc);
```

Syntaxe BASIC interne

```
Function AmConvertStringToDouble(strSrc As String)
As Double
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strSrc* : Ce paramètre contient la chaîne de caractères à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmConvertStringToMonetary()

Cette fonction convertit une chaîne de caractères (dans un format conforme à celui défini dans le panneau de contrôle de Windows) en une valeur monétaire.

Syntaxe API






```
double AmConvertStringToMonetary(char *strSrc);
```

Syntaxe BASIC interne

```
Function AmConvertStringToMonetary(strSrc As  
String) As Double
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strSrc* : Ce paramètre contient la chaîne de caractères à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCounter()


Cette fonction renvoie la valeur du compteur *strCounterName* incrémentée de 1 et exprimée sur *iWidth* chiffres. Des zéros sont ajoutés en préfixe si *iWidth* est supérieur à la valeur du compteur.

Syntaxe BASIC interne

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

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strCounterName* : Nom du compteur tel qu'il est défini sous AssetCenter (accès par le menu **Administration/ Compteurs**).
- *iWidth* : La valeur de ce paramètre force le formatage du résultat de la fonction, en l'exprimant sur n chiffres.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant renvoie la valeur du compteur "BonsLivraison" exprimée sur 5 chiffres :

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

Si par exemple le compteur "BonsLivraison" vaut "18", le résultat est le suivant :

```
00019
```

AmCreateAssetPort()

L'API `AmCreateAssetPort` crée un nouveau port pour un dispositif (`lAssetId`). Le nouveau port contient le nombre donné de broches (`iPinCount`) du type de connecteur de câble donné (`lCabCnxTypeId`). L'état des broches doit être "Disponible". Les broches qui seront ajoutées au port seront triées par numéro de séquence. Suivant la direction du port (`bPinPortDir`), les broches disponibles sont triées par ordre croissant (`bPinPortDir=0`) or décroissant (`bPinPortDir=1`). cette fonction assigne la fonction (`lDutyId`) donnée au nouveau port.

Syntaxe API




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

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lAssetId* : ce paramètre est l'identifiant du dispositif.
- *lCabCnxTypeId* : ce paramètre est l'identifiant du type de connexion de câble.
- *lDutyId* : ce paramètre est l'identifiant du type de fonction du port.
- *iPinCount* : ce paramètre est le nombre de broches qui seront utilisées dans le nouveau port.
- *bPinPortDir* : ce paramètre précise la direction du port.
- *iConnStatus*
- *bConsecutivePins*
- *iPrevPinSeq*
- *bLogError*

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCreateAssetsAwaitingDelivery()

Cette fonction permet de créer les biens qui sont en attente de réception

Syntaxe API

```
long AmCreateAssetsAwaitingDelivery(long
hApiCnxBase, long lPOrdId);
```

Syntaxe BASIC interne

```
Function AmCreateAssetsAwaitingDelivery(lPOrdId As
Long) As Long
```

Champ d'application

Version : 3.61

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	
Script FINISH.DO d'un assistant	✓

Entrée

- *lPOrdId* : Ce paramètre contient l'identifiant de la commande concernée

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCreateCable()

L'API AmCreateCable crée un nouveau câble. Le câble est créé suivant un modèle donné (lModelId), le rôle du câble (strCableRole), sa règle d'étiquetage (lLabelRuleId), sa localisation utilisateur (lUserId), et sa localisation hôte (lHostId). Si le projet (lProjectId) et l'intervention (lWorkOrderId) prennent des valeurs, le nouveau câble est ajouté au projet et à l'intervention avec son commentaire (strComment). ce commentaire décrit l'action qui sera accomplie sur le câble. (i.e. "Mettre en place un nouveau câble").

Syntaxe API




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

Syntaxe BASIC interne

```
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
```


Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lModelId* : ce paramètre est l'identifiant du modèle de câble.
- *lUserId* : ce paramètre est l'identifiant de la localisation côté utilisateur.
- *lHostId* : ce paramètre est l'identifiant de la localisation côté hôte.
- *strCableRole* : ce paramètre définit le rôle du câble.
- *lProjectId* : ce paramètre est le projet associé à la mise en place du câble.
- *lWorkOrderId* : ce paramètre est l'identifiant de l'intervention associée à la mise en place du câble.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention (précisée par *lWorkOrderId*).
- *lLabelRuleId* : ce paramètre contient l'identifiant de la règle d'étiquetage qui sera appliquée lors de la création de l'étiquette pour le câble.
- *strLabel* : ce paramètre précise l'étiquette apposée sur le câble.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCreateCableBundle()

L'API AmCreateCableBundle crée un nouveau faisceau pour un câble donné (lCableId). Le nouveau faisceau contient le nombre donné de paires de câble (iPairCount) du type donné de paire de câble (lPairType). L'état des paires doit être "Disponible". Cette fonction assigne la fonction donnée (lDutyId) au nouveau faisceau.

Syntaxe API



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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Entrée**

- *lCableId* : ce paramètre est l'identifiant du câble (cet identifiant doit exister dans la table des câbles).
- *lPairTypeId* : ce paramètre est l'identifiant du type de paire de câble.
- *lDutyId* : ce paramètre est l'identifiant de la fonction du faisceau.
- *iPairCount* : ce paramètre définit le nombre de paires du faisceau.
- *iStartPairSeq*
- *bLogError*

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateCableLink()

L'API `AmCreateCableLink` crée une nouvelle liaison câble pour un câble (`lCableId`) et faisceau (`lBundleId`) donnés. La fonction de la liaison câble est établie grâce à la fonction donnée (`lDutyId`). La règle d'étiquetage

de la liaison câble est établie grâce à la règle d'étiquetage donnée (lLabelRuleId).



Note : L'étiquette n'est pas mise à jour grâce à la règle d'étiquetage donnée, AmRefreshLabel() doit être appelée séparément.

Si une liaison précédente (lPrevLinkId) est spécifiée, une liaison parente est créée entre les deux enregistrements pour lesquels la liaison précédente est la liaison fille.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCableId* : ce paramètre est l'identifiant du câble pour la connexion.
- *lDutyId* : ce paramètre est l'identifiant de la fonction de connexion.
- *lBundleId* : ce paramètre est l'identifiant du faisceau de câble à connecter.
- *lPrevLinkId* : ce paramètre définit l'identifiant de la liaison câble utilisée par la connexion. Une valeur de 0 le rend facultatif.
- *iTraceDir* : ce paramètre définit la direction de la connexion.
 - 0=hôte vers utilisateur
 - 1=utilisateur vers hôte
- *lLabelRuleId* : ce paramètre est l'identifiant de la règle d'étiquetage utilisée.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateDelivFromPO()

Cette fonction effectue la réception d'une commande à partir d'une commande et renvoie l'identifiant de la fiche de réception créée.

Syntaxe API




```
long AmCreateDelivFromPO(long hApiCnxBase, long lPOrdId);
```

Syntaxe BASIC interne

```
Function AmCreateDelivFromPO(lPOrdId As Long) As
Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrdId* : Ce paramètre contient l'identifiant de la commande à réceptionner.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateDevice()

L'API AmCreateDevice crée un nouveau dispositif. Le dispositif est créé grâce au modèle (lModelId) et localisation (lLocationId) donnés. La règle d'étiquetage dépend de la règle donnée (lLabelRuleId).



Note : L'étiquette n'est pas mise à jour grâce à la règle d'étiquetage donnée, AmRefreshLabel() doit être appelée séparément.

Si le projet (lProjectId) et l'intervention (lWorkOrderId) prennent des valeurs, le nouveau bien est ajouté au projet et à l'intervention avec le commentaire contenu dans strComment. Ce commentaire décrit l'action effectuée sur le bien (i.e. "Installer un nouveau bien").

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application



Version : 4.00

Utilisable

AssetCenter APIs



Utilisable

Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lModelId* : ce paramètre contient l'identifiant du modèle du nouveau dispositif.
- *lLocationId* : ce paramètre contient l'identifiant de la localisation du nouveau dispositif.
- *lProjectId* : ce paramètre contient l'identifiant du projet. Il peut prendre la valeur 0.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention. Il peut prendre la valeur 0.
- *lLabelRuleId* : ce paramètre contient l'identifiant de la règle d'étiquetage qui sera utilisée par le bien.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateDeviceLink()

L'API AmCreateDeviceLink crée une liaison câble de type dispositif pour un dispositif (lAssetId) et un port (lPortId) donnés. La règle d'étiquetage de la liaison câble est établie grâce à la règle d'étiquetage donnée (lLabelRuleId).



Note : L'étiquette n'est pas mise à jour grâce à la règle d'étiquetage donnée, AmRefreshLabel() doit être appelée séparément.

Si une liaison précédente (lPrevLinkId) est spécifiée, une liaison parente est créée entre les deux enregistrements. Si la direction de la chaîne de liaisons est utilisateur vers hôte (iTraceDir = 1), alors la liaison précédente est fille. Si la direction de la chaîne de liaisons est hôte vers utilisateur (iTraceDir = 0), alors la liaison précédente est parent.

Syntaxe API

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

Syntaxe BASIC interne



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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	

Utilisable

Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lAssetId* : ce paramètre contient l'identifiant du bien qui sera connecté.
- *lPortId* : ce paramètre contient l'identifiant du port qui sera connecté.
- *lPrevLinkId* : ce paramètre contient l'identifiant de la liaison du dispositif permettant la connexion.
- *iTraceDir* : ce paramètre précise la direction de la connexion.
 - 0=hôte vers utilisateur
 - 1=utilisateur vers hôte
- *lLabelRuleId* : ce paramètre contient l'identifiant de la règle d'étiquetage utilisée pour la nouvelle connexion.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateEstimFromReq()

Cette fonction effectue la création d'une devis à partir d'une demande d'achat et renvoie l'identifiant du devis créé.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqId* : Ce paramètre contient l'identifiant de la demande d'achat qui sert à la création du devis.
- *lSuppId* : Ce paramètre contient l'identifiant du fournisseur du devis qui sera créé à l'issue de l'exécution de la fonction.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCreateEstimsFromAllReqLines()

Cette fonction crée un devis à partir d'une demande et renvoie l'identifiant du devis créé.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqId* : Ce paramètre contient l'identifiant de la demande à l'origine du devis.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.
- *lDefSuppId* : Ce paramètre contient l'identifiant du fournisseur par défaut pour le devis.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCreateInvFromPO()

Cette fonction crée une facture fournisseur à partir d'une commande et renvoie l'identifiant de la facture fournisseur créée.

Syntaxe API

```
long AmCreateInvFromPO(long hApiCnxBase, long lPOrdId);
```

Syntaxe BASIC interne

```
Function AmCreateInvFromPO(lPOrdId As Long) As Long
```

Champ d'application



Version : 3.00

Utilisable

AssetCenter APIs



Utilisable

Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrdId* : Ce paramètre contient l'identifiant de la commande à l'origine de la facture.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateLink()

Cette fonction modifie un lien d'un enregistrement et le fait pointer vers un nouvel enregistrement (*hApiRecDest*) dans la table de destination. Elle crée donc un lien entre deux enregistrements.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient le handle sur l'enregistrement contenant le lien à modifier.
- *strLinkName* : Ce paramètre contient le nom SQL du lien à modifier.
- *hApiRecDest* : Ce paramètre contient un handle sur l'enregistrement de destination du lien.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCreatePOFromEstim()

Cette fonction effectue la création d'une commande à partir d'un devis et renvoie l'identifiant de la commande créée.

Syntaxe API




```
long AmCreatePOFromEstim(long hApiCnxBase, long
lEstimId);
```

Syntaxe BASIC interne

```
Function AmCreatePOFromEstim(lEstimId As Long) As
Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lEstimId* : Ce paramètre contient l'identifiant du devis qui sert à la création de la commande.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreatePOFromReq()

Cette fonction effectue la création d'une commande à partir d'une demande d'achat et renvoie l'identifiant de la commande créée.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqId* : Ce paramètre contient l'identifiant de la demande d'achat qui sert à la création de la commande.
- *lSuppId* : Ce paramètre contient l'identifiant du fournisseur de la commande qui sera créée à l'issue de l'exécution de la fonction.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCreatePOrderFromRequest()

Cette fonction permet de créer une commande à partir d'une demande.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lRequestId* : Ce paramètre contient l'identifiant de la demande concernée.
- *lSupplierId* : Ce paramètre contient l'identifiant du fournisseur pour la commande.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreatePOrdersFromRequest()

Cette fonction permet de créer toutes les commandes nécessaires à la satisfaction d'une demande donnée.

Syntaxe API

```
long AmCreatePOrdersFromRequest(long hApiCnxBase,  
long lRequestId);
```

Syntaxe BASIC interne

```
Function AmCreatePOrdersFromRequest(lRequestId As  
Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	
Script FINISH.DO d'un assistant	✓

Entrée

- *lRequestId* : Ce paramètre contient l'identifiant de la demande concernée

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCreatePOsFromAllReqLines()

Cette fonction crée toutes les commandes à partir des lignes de demande d'une demande.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lReqId* : Ce paramètre contient l'identifiant de la demande à partir de laquelle les commandes vont être créées.
- *bMergeLines* : Ce paramètre permet de préciser si les lignes de demande identiques doivent être combinées (*bMergeLines*=1) pour n'en créer qu'une seule. Les quantités décrites sur les lignes à combiner sont alors ajoutées et une seule ligne est créée.
- *lDefSuppId* : Ce paramètre contient l'identifiant du fournisseur par défaut des éléments demandés. Ce paramètre est optionnel et possède "0" comme valeur par défaut..

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCreateProjectCable()

L'API `AmCreateProjectCable` ajoute un câble (`ICableId`) à un projet (`lProjectId`) et une intervention (`lWorkOrderId`). Un commentaire (`strComment`) explique l'action effectuée (i.e. "Installer un nouveau câble").

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lProjectId* : ce paramètre contient l'identifiant du projet qui obtient le nouveau câble.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention sur le câble.
- *lCableId* : ce paramètre contient l'identifiant du câble.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCreateProjectDevice()

L'API AmCreateProjectDevice ajoute un dispositif (lAssetId) à un projet (lProjectId) et une intervention (lWorkOrderId). Un commentaire (strComment) explique l'action effectuée (i.e. "Installer un nouveau dispositif").

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	



Entrée

- *lProjectId* : ce paramètre contient l'identifiant du projet qui obtient le nouveau dispositif.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention qui obtient le nouveau dispositif.
- *lAssetId* : ce paramètre contient l'identifiant du nouveau dispositif en tant que bien.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateProjectTrace()

L'API `AmCreateProjectTrace` ajoute une chaîne de liaisons (`strTrace`) à un projet (`lProjectId`) et une intervention (`lWorkOrderId`). La fonction de la chaîne de liaisons est établie suivant la fonction donnée (`lDutyId`). Le type de chaîne de liaisons (`iTraceType`) indique si la chaîne de liaisons est une connexion (`lTraceType = 1`) ou une déconnexion (`lTraceType = 2`). L'étiquette de la liaison utilisateur modifiée (`strModLinkLabel`) identifie quelle partie de la chaîne de liaisons est modifiée. Un

commentaire (strComment) explique l'action effectuée (i.e. "Connecter ces dispositifs").

Syntaxe API




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

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lProjectId* : ce paramètre contient l'identifiant du projet qui obtient l'information de la chaîne de liaisons.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention qui obtient l'information de la chaîne de liaisons.
- *iTraceType* : ce paramètre précise le type de chaîne de liaisons.

- 1=connexion
- 2=déconnexion
- *lDutyId* : ce paramètre contient l'identifiant de la fonction. Elle apparaît dans une intervention.
- *strModLinkLabel* : ce paramètre est un commentaire qui sera utilisé pour l'intervention.
- *strTrace* : ce paramètre est la chaîne de compte-rendu de la chaîne de liaisons qui sera utilisée pour l'intervention.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateReceiptFromPOrder()

Cette fonction permet de créer une réception à partir d'une commande.

Syntaxe API




```
long AmCreateReceiptFromPOrder(long hApiCnxBase,
long lPOrderId);
```

Syntaxe BASIC interne

```
Function AmCreateReceiptFromPOrder(lPOrderId As
Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrderId* : Ce paramètre contient l'identifiant de la commande concernée.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateRecord()

Cette fonction crée un enregistrement vide dans une table en tenant compte des valeurs par défaut. Ce nouvel enregistrement ne possède aucune existence dans la base de données tant qu'il n'a pas été inséré.

Syntaxe API




```
long AmCreateRecord(long hApiCnxBase, char
*strTable);
```

Syntaxe BASIC interne

```
Function AmCreateRecord(strTable As String) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTable* : Ce paramètre contient le nom SQL de la table dans laquelle vous souhaitez créer l'enregistrement.

Exemple

L'exemple suivant crée une personne dans la base de données :

```
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()

Cette fonction permet de créer tous les objets d'un cycle d'achat : Demande, Commande, Réception, Facture.

Syntaxe API




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

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *fQty* : Ce paramètre contient la quantité (en unités de conditionnement) à commander, réceptionner, puis facturer.
- *lCatRefId* : Ce paramètre contient l'identifiant de la référence catalogue.
- *dUnitPrice* : Ce paramètre contient le prix unitaire de la référence catalogue.
- *strCur* : Ce paramètre contient le code de la devise pour le prix de la référence catalogue.
- *lRequesterId* : Ce paramètre contient l'identifiant du demandeur.

- *lCostId* : Ce paramètre contient l'identifiant du centre de coût impacté.
- *lUserId* : Ce paramètre contient l'identifiant de l'utilisateur de l'élément commandé.
- *lStockId* : Ce paramètre contient l'identifiant du stock de livraison de l'élément.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques

Equivalent à la séquence d'appels : `amCreateRequestToReceipt`, `amCreateOrUpdateInvoiceFromReceipt`.

AmCreateRequestToPOrder()

Cette fonction permet de créer les objets d'un cycle d'achat : Demande, Commande.

Syntaxe API




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

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *fQty* : Ce paramètre contient la quantité (en unités de conditionnement) à commander.
- *lCatRefId* : Ce paramètre contient l'identifiant de la référence catalogue.
- *dUnitPrice* : Ce paramètre contient le prix unitaire de la référence catalogue.
- *strCur* : Ce paramètre contient le code de la devise pour le prix de la référence catalogue.
- *lRequesterId* : Ce paramètre contient l'identifiant du demandeur.
- *lCostId* : Ce paramètre contient l'identifiant du centre de coût impacté.
- *lUserId* : Ce paramètre contient l'identifiant de l'utilisateur de l'élément commandé.
- *lStockId* : Ce paramètre contient l'identifiant du stock de livraison de l'élément.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateRequestToReceipt()

Cette fonction permet de créer les objets d'un cycle d'achat : Demande, Commande, Reception.

Syntaxe API

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

Syntaxe BASIC interne

```
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
```

Champ d'application



Version : 4.00

Utilisable

AssetCenter APIs



Utilisable

Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *fQty* : Ce paramètre contient la quantité (en unités de conditionnement) à commander, puis réceptionner.
- *lCatRefId* : Ce paramètre contient l'identifiant de la référence catalogue.
- *dUnitPrice* : Ce paramètre contient le prix unitaire de la référence catalogue.
- *strCur* : Ce paramètre contient le code de la devise pour le prix de la référence catalogue.
- *lRequesterId* : Ce paramètre contient l'identifiant du demandeur.
- *lCostId* : Ce paramètre contient l'identifiant du centre de coût impacté.
- *lUserId* : Ce paramètre contient l'identifiant de l'utilisateur de l'élément commandé.
- *lStockId* : Ce paramètre contient l'identifiant du stock de livraison de l'élément.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques

Equivalent à la séquence d'appels : `amCreateRequestToPOrder`,
`amCreateReceiptFromPOrder`.

AmCreateReturnFromReceipt()

Cette fonction permet de créer une fiche de retour à partir d'une fiche de réception.

Syntaxe API




```
long AmCreateReturnFromReceipt(long hApiCnxBase,
long lRecptId);
```

Syntaxe BASIC interne

```
Function AmCreateReturnFromReceipt(lRecptId As
Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lRecptId* : Ce paramètre contient l'identifiant de la réception.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmCreateTraceHist()

L'API `AmCreateTraceHist` sert à créer l'historique de chaîne de liaisons et l'opération sur chaîne de liaisons à partir d'une connexion existante depuis un dispositif/câble source vers un dispositif/câble destination.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application



Version : 4.00

Utilisable

AssetCenter APIs



Utilisable

Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lSrcLinkId* : ce paramètre contient l'identifiant de la liaison attribué à la liaison source.
- *lDestLinkId* : ce paramètre contient l'identifiant de la liaison attribué à la liaison destination.
- *iTraceDir* : ce paramètre précise la direction de la connexion.
 - 0=hôte vers utilisateur
 - 1=utilisateur vers hôte
- *lCabTraceOutId* : ce paramètre contient l'identifiant de compte-rendu de chaîne de liaisons de câble.
- *strComment* : ce paramètre est le commentaire qui sera associé à l'opération de chaîne de liaisons.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmCryptPassword()

Cette fonction crypte le mot de passe d'un utilisateur, identifié par son login et son mot de passe.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strUser* : Ce paramètre contient le login de l'utilisateur dont vous souhaitez crypter le mot de passe.
- *strPasswd* : Ce paramètre contient, en clair, le mot de passe à crypter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la

fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCurrentDate()

Cette fonction renvoie la date courante sur le poste client.

Syntaxe API






```
long AmCurrentDate();
```

Syntaxe BASIC interne

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

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCurrentIsoLang()

Cette fonction renvoie, conformément à la norme ISO, la langue utilisée dans AssetCenter ("fr" pour le français, "en" pour l'anglais,...).

Syntaxe API






```
long AmCurrentIsoLang(char *pstrLanguage, long
lLanguage);
```

Syntaxe BASIC interne

```
Function AmCurrentIsoLang() As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCurrentLanguage()

Cette fonction renvoie la langue utilisée dans AssetCenter ("FR" pour le français, "US" pour l'anglais,...).

Syntaxe API






```
long AmCurrentLanguage(char *pstrLanguage, long lLanguage);
```

Syntaxe BASIC interne

```
Function AmCurrentLanguage() As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmCurrentServerDate()

Cette fonction renvoie la date courante sur le serveur.

Syntaxe API






```
long AmCurrentServerDate(long hApiCnxBase);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmDateAdd()

Cette fonction calcule une nouvelle date en fonction d'une date de départ à laquelle est ajoutée une durée réelle.

Syntaxe API






```
long AmDateAdd(long tmStart, long tsDuration);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *tmStart* : Ce paramètre contient la date à laquelle sera ajoutée une durée.
- *tsDuration* : Ce paramètre contient la durée à ajouter à la date *tmStart*.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmDateAddLogical()

Cette fonction calcule une nouvelle date en fonction d'une date de départ à laquelle est ajoutée une durée logique (un mois comporte 30 jours).

Syntaxe API






```
long AmDateAddLogical(long tmStart, long
tsDuration);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *tmStart* : Ce paramètre contient la date à laquelle sera ajoutée une durée.
- *tsDuration* : Ce paramètre contient la durée à ajouter à la date *tmStart*.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmDateDiff()

Cette fonction calcule en secondes la durée écoulée entre deux dates.

Syntaxe API

```
long AmDateDiff(long tmEnd, long tmStart);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

AssetCenter APIs	Utilisable 
------------------	--

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmEnd* : Ce paramètre contient la date de fin de la période sur laquelle est effectué le calcul.
- *tmStart* : Ce paramètre contient la date de début de la période sur laquelle est effectué le calcul.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant calcule la durée écoulée entre le 01/01/98 et le 01/01/99.

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

AmDbGetDate()

Cette fonction renvoie le résultat au format date, de l'exécution d'une requête AQL sans curseur. Si la requête ne renvoie aucun résultat, la valeur 0 est retournée sans déclencher d'erreur.

Syntaxe API






```
long AmDbGetDate(long hApiCnxBase, char *strQuery);
```

Syntaxe BASIC interne

```
Function AmDbGetDate(strQuery As String) As Date
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strQuery* : Ce paramètre contient l'intégralité de la requête AQL dont on veut récupérer le résultat.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmDbGetDouble()

Cette fonction renvoie le résultat (sous la forme d'un nombre en double précision), de l'exécution d'une requête AQL. Si la requête ne renvoie aucun résultat, la valeur 0 est retournée sans déclencher d'erreur.

Syntaxe API






```
double AmDbGetDouble(long hApiCnxBase, char
*strQuery);
```

Syntaxe BASIC interne

```
Function AmDbGetDouble(strQuery As String) As
Double
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strQuery* : Ce paramètre contient l'intégralité de la requête AQL dont on veut récupérer le résultat.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmDbGetList()

Cette fonction renvoie, sous la forme d'une liste, le résultat de l'exécution d'une requête AQL. Le nombre d'éléments sélectionnés par la requête AQL est limité à 99.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strQuery* : Ce paramètre contient la requête AQL que vous souhaitez exécuter.
- *strColSep* : Ce paramètre contient le caractère utilisé comme séparateur de colonnes dans le résultat donné par la fonction.
- *strLineSep* : Ce paramètre contient le caractère utilisé comme séparateur de lignes dans le résultat donné par la fonction.
- *strIdSep* : Ce paramètre contient le caractère utilisé comme séparateur d'identifiant dans le résultat donné par la fonction.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmDbGetListEx()

Cette fonction renvoie, sous la forme d'une liste, le résultat de l'exécution d'une requête AQL. A la différence de la fonction `AmDbGetList`, cette fonction n'est pas limitée dans le nombre d'éléments sélectionnés par la requête AQL.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strQuery* : Ce paramètre contient la requête AQL que vous souhaitez exécuter.
- *strColSep* : Ce paramètre contient le caractère utilisé comme séparateur de colonnes dans le résultat donné par la fonction.
- *strLineSep* : Ce paramètre contient le caractère utilisé comme séparateur de lignes dans le résultat donné par la fonction.
- *strIdSep* : Ce paramètre contient le caractère utilisé comme séparateur d'identifiant dans le résultat donné par la fonction.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

AmDbGetLong()

Cette fonction renvoie le résultat de l'exécution d'une requête AQL. Si la requête ne renvoie aucun résultat, la valeur 0 est retournée sans déclencher d'erreur.

Syntaxe API






```
long AmDbGetLong(long hApiCnxBase, char *strQuery);
```

Syntaxe BASIC interne

```
Function AmDbGetLong(strQuery As String) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strQuery*: Ce paramètre contient l'intégralité de la requête AQL dont on veut récupérer le résultat.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant renvoie le numéro d'identifiant d'un fournisseur de produit :

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

AmDbGetPk()

Cette fonction renvoie la clé primaire d'une table en fonction de la clause WHERE d'une requête AQL. Si la requête ne renvoie aucun résultat, la valeur 0 est retournée sans déclencher d'erreur.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTableName* : Nom SQL de la table dont on veut récupérer la clé primaire.
- *strWhere* : Clause WHERE d'une requête AQL.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmDbGetString()

Cette fonction renvoie, sous la forme d'une chaîne formatée, le résultat de l'exécution d'une requête AQL. Le nombre d'éléments sélectionnés par la requête AQL est limité à 99.



Attention: N'utilisez pas cette fonction pour récupérer la valeur d'un simple champ de type chaîne. Cette fonction est à rapprocher de la fonction `AmDbGetList` ou de la fonction `AmDbGetListEx`.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strQuery* : Ce paramètre contient la requête AQL que vous souhaitez exécuter.
- *strColSep* : Ce paramètre contient le caractère utilisé comme séparateur de colonnes dans la chaîne finale.
- *strLineSep* : Ce paramètre contient le caractère utilisé comme séparateur de lignes dans la chaîne finale.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques

Dans la syntaxe API, le paramètre `lResult` doit contenir la taille attendue du résultat de l'exécution de la fonction.

Exemple

```
Dim sQry As String
Dim lErr As Long
Dim sStrResult As String * 100
    sQry = "SELECT IDNo FROM AmEmplDept WHERE
lEmplDeptId = 128"
    lErr = AmDbGetString(g_lCnx, sQry, sStrResult,
100, "|", "|")
    MsgBox "sStrResult: " + sStrResult
```

AmDbGetStringEx()

Cette fonction renvoie, sous la forme d'une chaîne formatée, le résultat de l'exécution d'une requête AQL. A la différence de la fonction `AmDbGetString`, cette fonction n'est pas limitée dans le nombre d'éléments sélectionnés par la requête AQL.



Attention: N'utilisez pas cette fonction pour récupérer la valeur d'un simple champ de type chaîne. Cette fonction est à rapprocher de la fonction `AmDbGetList` ou de la fonction `AmDbGetListEx`.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strQuery* : Ce paramètre contient la requête AQL que vous souhaitez exécuter.
- *strColSep* : Ce paramètre contient le caractère utilisé comme séparateur de colonnes dans la chaîne finale.
- *strLineSep* : Ce paramètre contient le caractère utilisé comme séparateur de lignes dans la chaîne finale.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmDeadline()

Cette fonction calcule une date d'échéance en fonction d'un calendrier, d'une date de début et d'un nombre de secondes ouvrées écoulées.

Syntaxe API






```
long AmDeadline(char *strCalendarSqlName, long
tmStart, long tsDuration);
```

Syntaxe BASIC interne

```
Function AmDeadline(strCalendarSqlName As String,
tmStart As Date, tsDuration As Long) As Date
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strCalendarSqlName* : Ce paramètre contient le nom SQL du calendrier des périodes ouvrées utilisé comme base pour le calcul de la date d'échéance.
- *tmStart* : Ce paramètre contient la date de début de la période.
- *tsDuration* : Ce paramètre contient le nombre de secondes ouvrées écoulées à partir de la date de début de période.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant calcule la date d'échéance en fonction du calendrier de nom SQL "Calendar_Paris", d'une date de début de période fixée au 01/09/1998 à 8h00 et d'un nombre de secondes écoulées de 450000.

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

Cet exemple renvoie la date d'échéance, à savoir le 22/09/1998 à 18h00.

AmDecrementLogLevel()


Cette fonction permet de remonter d'un cran dans l'arborescence d'une fenêtre de journal de la page terminale d'un assistant.

Syntaxe BASIC interne

```
Function AmDecrementLogLevel() As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmDefAssignee()

Cette fonction recherche le numéro d'identifiant du chargé de groupe par défaut pour un groupe de support donné.

Syntaxe API






```
long AmDefAssignee(long hApiCnxBase, long  
lGroupId);
```

Syntaxe BASIC interne

```
Function AmDefAssignee(lGroupId As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lGroupId* : Ce paramètre contient le numéro d'identifiant d'un groupe de support.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple générique suivant renvoie l'identifiant du chargé par défaut d'un groupe de support :

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

Vous pouvez directement saisir la valeur numérique de l'identifiant, comme dans cet exemple :

```
AmDefAssignee(24)
```

AmDefaultCurrency()

Cette fonction renvoie la devise par défaut utilisée sous AssetCenter.

Syntaxe API






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

Syntaxe BASIC interne

```
Function AmDefaultCurrency() As String
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmDefEscalationScheme()

Cette fonction recherche la procédure d'escalade par défaut en fonction de la localisation et de la gravité du dossier de support.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strLocFullName* : Ce paramètre contient le nom complet de la localisation.

- *lSeverityLvl* : Ce paramètre contient la valeur de la gravité.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple générique suivant renvoie l'identifiant de la procédure d'escalade par défaut en fonction de la localisation et de la gravité :

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

Vous pouvez directement saisir les valeurs des paramètres, comme dans cet exemple :

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

AmDefGroup()

Cette fonction renvoie le numéro d'identifiant du groupe de support par défaut en fonction d'un type de problème, d'une localisation et d'un contrat de maintenance.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lProblemClassId* : Ce paramètre contient le numéro d'identifiant d'un type de problème.
- *strLocFullName* : Ce paramètre contient le nom complet d'une localisation.
- *lAssetMainCntId* : Ce paramètre contient le numéro d'identifiant d'un contrat de maintenance.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple générique suivant calcule le numéro d'identifiant du groupe de support par défaut en fonction des trois paramètres : type de problème, localisation et contrat de maintenance.

```
AmDefGroup([ProblemClass.IdClassId],[Asset.Location.FullName],[Asset.MaintCtrId])
```

Vous pouvez directement saisir la valeur numérique des paramètres utilisant des numéros d'identifiant, comme dans cet exemple :

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

AmDeleteLink()

Cette fonction détruit un lien d'un enregistrement.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord*: Ce paramètre contient le handle sur l'enregistrement contenant le lien à détruire.
- *strLinkName*: Ce paramètre contient le nom SQL du lien à détruire.
- *hApiRecDest*: Ce paramètre contient un handle sur l'enregistrement de destination du lien qui doit être détruit.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmDeleteRecord()

Cette fonction détruit un enregistrement dans la base de données.

Syntaxe API




```
long AmDeleteRecord(long hApiRecord);
```

Syntaxe BASIC interne

```
Function AmDeleteRecord(hApiRecord As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient un handle sur l'enregistrement que vous souhaitez détruire.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmDisconnectTrace()

L'API AmDisconnectTrace déconnecte la chaîne de liaisons entre un noeud utilisateur (lEndId) et un noeud hôte (lStartId) dans la table des liaisons de câbles. Si tout noeud se trouve à la fin d'une chaîne de liaisons, il sera supprimé de la table des liaisons de câbles. Il crée aussi des entrées d'historique de chaîne de liaisons et des entrées d'opérations sur chaîne de liaisons suite à la déconnexion.

Syntaxe API

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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	
Script FINISH.DO d'un assistant	✓

Entrée

- *lStartId* : ce paramètre contient l'identifiant de la connexion hôte lequel va être déconnecté.
- *lEndId* : ce paramètre contient l'identifiant de la connexion utilisateur lequel va être déconnecté.
- *strComment* : ce paramètre est la chaîne de l'opération sur la chaîne de liaisons montrant les nouvelles connexions et déconnexions.
- *lCabTraceOutId* : ce paramètre contient l'identifiant de compte-rendu de chaîne de liaisons de câble.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmDuplicateRecord()

Cette fonction permet de dupliquer un enregistrement.

Syntaxe API




```
long AmDuplicateRecord(long hApiRecord, long bInsert);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient le handle de l'enregistrement à dupliquer.
- *bInsert* : Ce paramètre permet de préciser si vous souhaitez insérer l'enregistrement dupliqué immédiatement (=1) ou non (=0).

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmEndOfNthBusinessDay()

Donne la dernière heure ouvrée du nième jour (identifié par l'entier *lDayCount*) à compter d'une date donnée et en respectant un calendrier.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strCalendarSqlName* : Nom du calendrier utilisé pour le calcul.
- *tmStart* : Date de début du calcul.
- *lDayCount* : Nombre de jours ouvrés entiers à ajouter à dStart pour le calcul.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

AmEvalScript()

Cette fonction permet d'évaluer un script par son nom à partir du contexte courant. Cette fonction possède deux utilisations :

- Evaluer un script système (Valeur par défaut, Obligatoire...)
- Appeler une fonction d'une bibliothèque de scripts.

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strScriptName* : Ce paramètre contient le nom du script à évaluer. Dans le premier cas d'utilisation, il s'agit du nom du script système (DefVal, ...). Dans le deuxième cas d'utilisation, il s'agit du nom d'une fonction faisant partie d'une bibliothèque de scripts (le nom de la bibliothèque est alors spécifié par le paramètre `strObject`).

- *strObject* : Ce paramètre contient l'objet auquel se rapporte le script. Il peut s'agir du nom SQL d'un champ ou d'un nom de bibliothèque.
- *strPath* : Ce paramètre optionnel permet de préciser un chemin (lien.lien.lien...) qui permet de décaler le contexte d'évaluation du script. Ce paramètre est inopérant dans le deuxième cas d'utilisation de la fonction.
- . . . : Pour l'appel à une fonction d'une bibliothèque de scripts, permet de passer des paramètres à la fonction appelée.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmExecTransition()

Cette fonction déclenche une transition valide à partir de la page en cours.


Syntaxe BASIC interne

```
Function AmExecTransition(strTransName As String)
As Long
```

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTransName* : Ce paramètre contient le nom de la transition tel qu'il est défini dans le script de l'assistant. Une erreur est renvoyée si la transition n'existe pas. La fonction est inopérante (et ne renvoie pas d'erreur) si la transition n'est pas valide.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmExecuteActionById()

Cette fonction exécute une action identifiée par son identifiant.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lActionId* : Ce paramètre contient l'identifiant de l'action à exécuter.
- *strTableName* : Dans le cas d'une action contextuelle, ce paramètre contient le nom SQL de la table sur laquelle l'action est exécutée. Si ce paramètre est omis, dans le cas d'une action contextuelle, la fonction échouera. Pour une action non contextuelle, ce paramètre n'est pas interprété et donc facultatif.
- *lRecordId* : Ce paramètre contient l'identifiant de l'enregistrement sur lequel porte éventuellement l'action. Pour une action non contextuelle, ce paramètre n'est pas interprété et donc facultatif.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmExecuteActionByName()

Cette fonction exécute une action identifiée par son nom SQL.

Syntaxe API




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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strSqlName* : Ce paramètre contient le nom SQL de l'action à exécuter.
- *strTableName* : Dans le cas d'une action contextuelle, ce paramètre contient le nom SQL de la table sur laquelle l'action est exécutée. Si ce paramètre est omis, dans le cas d'une action contextuelle, la fonction échouera. Pour une action non contextuelle, ce paramètre n'est pas interprété et donc facultatif.
- *lRecordId* : Ce paramètre contient l'identifiant de l'enregistrement sur lequel porte éventuellement l'action. Pour une action non contextuelle, ce paramètre n'est pas interprété et donc facultatif.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmExportDocument()

Cette fonction permet d'exporter un document attaché à un enregistrement.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lDocId* : Ce paramètre contient l'identifiant du document à exporter.

- *strFileName* : Ce paramètre contient le nom du document à exporter, tel qu'il est stocké dans le champ FileName de la table des documents.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmFindCable()

L'API AmFindCable trouve le câble disponible suivant tiré entre la localisation d'un utilisateur (IUserId) et d'un hôte (IHostId) donnés. Le câble doit être du type (strCabType) et du rôle (strCableRole) spécifiés. Aussi, l'état du câble doit prendre la valeur "Disponible". Les câbles sont triés par ordre croissant d'identifiant et seuls les câbles plus grands que l'identifiant (lPrevCabId) du câble précédent sont sélectionnés.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPrevCableId* : ce paramètre contient l'identifiant du câble précédent.
- *strCabType* : ce paramètre définit le type du câble à chercher.
- *lUserId* : ce paramètre contient l'identifiant de la localisation utilisateur.
- *lHostId* : ce paramètre contient l'identifiant de la localisation hôte.
- *strCableRole* : ce paramètre est le rôle du câble à localiser.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmFindDevice()

L'API `AmFindDevice` trouve un dispositif d'un type (`strDevType`) donné dans une localisation (`lLocationId`) donnée. Les dispositifs sont triés par ordre croissant d'identifiant et seuls ceux plus grands que le dispositif (`lPrevDeviceId`) précédent sont sélectionnés.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPrevDeviceId* : ce paramètre contient l'identifiant du précédent dispositif recherché. La valeur 0 est utilisée pour démarrer une recherche.
- *strDeviceType* : ce paramètre définit le type du dispositif à localiser.
- *lLocationId* : ce paramètre contient l'identifiant de la localisation à chercher.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmFindRootLink()

Syntaxe API






```
long AmFindRootLink(long hApiCnxBase, long
lLinkId);
```

Syntaxe BASIC interne

```
Function AmFindRootLink(lLinkId As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmFindTermDevice()

L'API `AmFindTermDevice` trouve le dispositif disponible suivant dans un répartiteur donné (`lTermField`) pour un rôle de câble donné (`strCableRole`). Les dispositifs sont triés par ordre croissant de numéro de séquence et seuls les biens plus grands que le numéro de séquence précédent (`strPrevTermSeq`) sont sélectionnés. Aussi, pour les dispositifs à base de broches (`bPinBased=1`), on compare le nombre total de broches requises (`iPinPortCount`) au nombre total de broches restant sur le dispositif. Pour les dispositifs à base de ports (`bPinBased=0`) on s'assure qu'il y a au moins un port restant sur le dispositif et que le côté hôte ou utilisateur de ce port est disponible grâce au drapeau (`bCheckAvail = 0 - user device, bCheckAvail = 1 - host device`).

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *iPrevTermSeq* : ce paramètre est la séquence précédente du répartiteur recherché. La valeur 0 est utilisée pour démarrer une recherche.
- *lTermFieldId* : ce paramètre contient l'identifiant du répartiteur.
- *strCableRole* : ce paramètre est le rôle du câble à localiser.
- *bPinBased* : ce paramètre précise si le dispositif est à base de broches ou de ports.
- *iPinPortCount* : pour les dispositifs à base de broches, ce paramètre est le nombre total de broches requis pour créer un port virtuel. Pour les dispositifs à base de ports, ce paramètre est 1 puisque cette API est appelée pour chaque port requis.
- *bCheckAvail* : ce paramètre sert à déterminer quel côté du port doit être disponible.
 - 0=dispositif utilisateur, vérifie l'hôte disponible
 - 1=dispositif hôte, vérifie l'utilisateur disponible

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmFindTermField()

L'API `AmFindTermField` trouve un répartiteur qui fournit la fonction (`lDutyId`) donnée depuis la localisation (`lLocationId`) donnée. Elle continuera à trouver des répartiteurs additionnels dans une localisation donnée et pour une fonction donnée si la valeur de `lPrevTermFieldId` est plus grande que 0.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lDutyId* : ce paramètre définit la fonction à localiser.
- *lLocationId* : ce paramètre contient l'identifiant de la localisation à chercher.
- *lPrevTermFieldId* : ce paramètre contient l'identifiant du répartiteur.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGenSqlName()

Cette fonction génère un nom SQL valide à partir d'une chaîne texte classique. Les espaces sont remplacés par des underscores ("_"). Cette fonction est particulièrement utile pour définir la valeur par défaut du nom SQL d'une caractéristique à partir de son nom.

Syntaxe API






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

Syntaxe BASIC interne

```
Function AmGenSqlName(strText As String) As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strText* : Chaîne de caractères à partir de laquelle vous souhaitez générer un nom SQL.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant définit la valeur par défaut du nom SQL d'un objet de nom "Label" de la base de données AssetCenter :

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

AmGetComputeString()

Cette fonction renvoie la chaîne de description d'un enregistrement donné en fonction d'un modèle.

Syntaxe API






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

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTableName* : Ce paramètre contient le nom SQL de la table de l'enregistrement dont on souhaite récupérer la chaîne de description.
- *lRecordId* : Ce paramètre contient l'identifiant de l'enregistrement au sein de la table.
- *strTemplate* : Ce paramètre contient, sous forme de chaîne de caractères, le modèle utilisé pour la chaîne de description.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
dim strCS As String
strCS = amGetComputeString("amEmplDept",
[lEmplDeptId], "[Name], [FirstName]")
print strCS
```

AmGetCurrentNTDomain()

Syntaxe API



```
long AmGetCurrentNTDomain(char *pstrDomain, long
lDomain);
```




Syntaxe BASIC interne

```
Function AmGetCurrentNTDomain() As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmGetCurrentNTUser()

Cette fonction permet de récupérer le login de l'utilisateur connecté à Windows (NT ou 2000).

Syntaxe API

```
long AmGetCurrentNTUser(char *pstrUser, long lUser);
```





Syntaxe BASIC interne

```
Function AmGetCurrentNTUser() As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFeat()

Cette fonction crée un objet caractéristique à partir du handle d'une table et retourne le handle de l'objet caractéristique créé.

Syntaxe API






```
long AmGetFeat(long hApiTable, long lPos);
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle sur une table.
- *lPos* : Ce paramètre contient la position de la caractéristique à l'intérieur de la table.

AmGetFeatCount()

Cette fonction renvoie le nombre de caractéristiques sur la table précisée dans le paramètre *hApiTable*.

Syntaxe API





```
long AmGetFeatCount(long hApiTable);
```

Syntaxe BASIC interne

```
Function AmGetFeatCount(hApiTable As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Entrée**

- *hApiTable* : Ce paramètre contient un handle sur une table.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetField()

Cette fonction crée un objet champ à partir du handle d'une requête, d'un enregistrement ou d'une table et retourne le handle de l'objet champ créé.

Syntaxe API






```
long AmGetField(long hApiObject, long lPos);
```

Syntaxe BASIC interne

```
Function AmGetField(hApiObject As Long, lPos As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête, un enregistrement ou une table.
- *lPos* : Ce paramètre contient la position du champ (son index) à l'intérieur de l'objet.

AmGetFieldCount()

Cette fonction renvoie le nombre de champs contenus dans l'objet courant.

Syntaxe API



```
long AmGetFieldCount(long hApiObject);
```

Syntaxe BASIC interne

```
Function AmGetFieldCount(hApiObject As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur un enregistrement, une requête ou une table valides.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldDateValue()

Cette fonction renvoie la valeur d'un champ contenu dans l'objet courant. Cette valeur est renvoyée au format "Date" (à partir d'un outil externe, il s'agit d'un Long).

Syntaxe API






```
long AmGetFieldDateValue(long hApiObject, long lFieldPos);
```

Syntaxe BASIC interne

```
Function AmGetFieldDateValue(hApiObject As Long, lFieldPos As Long) As Date
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête ou un enregistrement.
- *lFieldPos* : Ce paramètre contient le numéro du champ à l'intérieur de l'objet courant.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldDescription()

Cette fonction renvoie, sous la forme d'une chaîne de caractères (format "String"), la description d'un champ identifié par un handle.

Syntaxe API






```
long AmGetFieldDescription(long hApiField, char
*pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetFieldDescription(hApiField As Long)
As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le champ dont on souhaite connaître la description longue.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldDoubleValue()

Cette fonction renvoie la valeur d'un champ contenu dans l'objet courant. Cette valeur est renvoyée au format "Double".

Syntaxe API






```
double AmGetFieldDoubleValue(long hApiObject, long
lFieldPos);
```

Syntaxe BASIC interne

```
Function AmGetFieldDoubleValue(hApiObject As Long,
lFieldPos As Long) As Double
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête ou un enregistrement.
- *lFieldPos* : Ce paramètre contient le numéro du champ à l'intérieur de l'objet courant.

Sortie

En cas d'erreur, deux cas de figure se présentent :


```

dleiã
pãh6
ãmaã
ãots
e l
mo n
LQS
e d
a l
ellat
ãããã
e l
pãhc
e uq
esiãp
e l
pãhc
.tããã

```

Syntaxe API

```

long AmGetFieldFormat(long hApiField, char
*pstrBuffer, long lBuffer);

```

Syntaxe BASIC interne






```

Function AmGetFieldFormat(hApiField As Long) As
String

```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le champ dont on souhaite connaître le "UserType".

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldFormatFromName()

Cette fonction renvoie le format du "UserType" d'un champ, à partir de son nom.

Syntaxe API

```
long AmGetFieldFormatFromName(long hApiCnxBase,
char *strTableName, char *strFieldName, char
*pFieldFormat, long lpFieldFormat);
```

Syntaxe BASIC interne

```
Function AmGetFieldFormatFromName(strTableName As
String, strFieldName As String) As String
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strTableName* : Ce paramètre contient le nom SQL de la table contenant le champ concerné par l'opération.
- *strFieldName* : Ce paramètre contient le nom SQL du champ.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldFromName()

Cette fonction crée un objet champ à partir de son nom et retourne le handle de l'objet champ créé.

Syntaxe API






```
long AmGetFieldFromName(long hApiObject, char
*strName);
```

Syntaxe BASIC interne

```
Function AmGetFieldFromName(hApiObject As Long,
    strName As String) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête, un enregistrement ou une table.
- *strName* : Ce paramètre contient le nom du champ.

AmGetFieldLabel()

Cette fonction renvoie, sous la forme d'une chaîne de caractères (format "String"), le libellé d'un champ identifié par un handle.

Syntaxe API






```
long AmGetFieldLabel(long hApiField, char
    *pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetFieldLabel(hApiField As Long) As
    String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le champ dont on souhaite connaître le libellé.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldLabelFromName()

Cette fonction renvoie le label d'un champ à partir de son nom SQL.

Syntaxe API






```
long AmGetFieldLabelFromName(long hApiCnxBase, char
*strTableName, char *strFieldName, char
*pFieldLabel, long lpFieldLabel);
```

Syntaxe BASIC interne

```
Function AmGetFieldLabelFromName(strTableName As String, strFieldName As String) As String
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTableName* : Ce paramètre contient le nom SQL de la table contenant le champ concerné par l'opération.
- *strFieldName* : Ce paramètre contient le nom SQL du champ.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldLongValue()

Cette fonction renvoie la valeur d'un champ contenu dans l'objet courant.

Syntaxe API






```
long AmGetFieldLongValue(long hApiObject, long lFieldPos);
```

Syntaxe BASIC interne

```
Function AmGetFieldLongValue(hApiObject As Long, lFieldPos As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête ou un enregistrement.
- *lFieldPos* : Ce paramètre contient le numéro du champ à l'intérieur de l'objet courant.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Si vous utilisez cette fonction pour récupérer la valeur d'un champ de type date, heure ou date+heure, l'entier long renvoyé par la fonction représente le nombre de secondes écoulées depuis le 01/01/1970 à 00:00:00.

AmGetFieldName()

Cette fonction renvoie le nom d'un champ contenu dans l'objet courant.

Syntaxe API






```
long AmGetFieldName(long hApiObject, long  
lFieldPos, char *pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetFieldName(hApiObject As Long,  
lFieldPos As Long) As String
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête, un enregistrement ou une table.
- *lFieldPos* : Ce paramètre contient le numéro du champ à l'intérieur de l'objet courant. Par exemple, la valeur "0" désigne le premier champ.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldRights()

Cette fonction renvoie les droits utilisateurs d'un champ de l'objet courant. Ces droits sont renvoyés sous la forme d'une chaîne composée de trois caractères précisant les droits en lecture/ insertion/ mise à jour :

- "r" : désigne l'autorisation en lecture.
- "i" : désigne l'autorisation en insertion.
- "u" : désigne l'autorisation en mise à jour.

Par exemple, pour un champ en lecture seule, la fonction renverra la valeur "r".

Syntaxe API






```
long AmGetFieldRights(long hApiObject, long
lFieldPos, char *pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetFieldRights(hApiObject As Long,
lFieldPos As Long) As String
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête, un enregistrement ou une table.
- *lFieldPos* : Ce paramètre contient le numéro du champ à l'intérieur de l'objet courant.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldSize()

Cette fonction renvoie la taille d'un champ.

Syntaxe API






```
long AmGetFieldSize(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetFieldSize(hApiField As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le champ dont on veut connaître la taille.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldSqlName()

Cette fonction renvoie, sous la forme d'une chaîne de caractères (format "String"), le nom SQL d'un champ identifié par un handle.

Syntaxe API






```
long AmGetFieldSqlName(long hApiField, char
*pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetFieldSqlName(hApiField As Long) As
String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le champ dont on souhaite connaître le nom SQL.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldStringValue()

Cette fonction renvoie la valeur d'un champ contenu dans l'objet courant. Cette valeur est renvoyée au format chaîne.

Attention : Quand cette fonction est utilisée au travers des APIs AssetCenter, elle attend deux paramètres supplémentaires *strBuffer* et *lBuffer*, qui définissent respectivement une chaîne de caractères utilisée comme buffer pour le stockage de la chaîne récupérée et la taille de ce buffer. La chaîne *strBuffer* doit être formatée (remplie de caractères) et posséder la taille définie par *lBuffer*. La portion de code suivante est incorrecte, la chaîne utilisée comme buffer n'étant pas dimensionnée :

```
Dim strBuffer as String
Dim lRec as Long
Dim lBuffer as Long
lBuffer=20
lRec=AmGetFieldStringValue(1, 0, strBuffer, lBuffer)
```

Voici la portion de code corrigée :

```
Dim strBuffer as String
Dim lRec as Long
Dim lBuffer as Long
strBuffer=String(21, " ") ' Le buffer est
dimensionné à 21 caractères (" ")
lBuffer=20
lRec=AmGetFieldStrValue(1, 0, strBuffer, lBuffer)
```

Lorsque vous formatez la chaîne du buffer au moyen de la fonction "String", n'utilisez jamais "0" comme caractère de remplissage. Dimensionnez le buffer avant chaque appel de la fonction AmGetFieldStrValue, en particulier si cette fonction se trouve dans une boucle et utilise toujours la même chaîne comme buffer.

Syntaxe API






```
long AmGetFieldStrValue(long hApiObject, long
lFieldPos, char *pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetFieldStrValue(hApiObject As Long,
lFieldPos As Long) As String
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête ou un enregistrement.
- *lFieldPos* : Ce paramètre contient le numéro du champ à l'intérieur de l'objet courant.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetFieldType()

Cette fonction renvoie le type d'un champ.

Syntaxe API

```
long AmGetFieldType(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetFieldType(hApiField As Long) As Long
```

Champ d'application

Version : 2.52

Utilisable
AssetCenter APIs 

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *hApiField* : Ce paramètre contient un handle sur le champ dont on veut connaître le type.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Le tableau ci-dessous donne la correspondance entre les valeurs retournées par la fonction `AmGetFieldType` et les types de champs :

AmGetFieldType()

Cette fonction renvoie le "UserType" (cf. fichier **database.txt**) d'un champ identifié par un handle, sous la forme d'un entier long. Les valeurs valides renvoyées sont récapitulées dans le tableau ci-dessous :

nelV
 sicta
 rialc
 tlaD
 rebN
 /seX
 o N
 yenM
 etaD
 mitaD
 mtsyS
 diti
 tsil
 mt suE
 diti
 tsil
 @mP
 miT
 naps
 elbaT
 r o
 deif
 LQS
 eman

Syntaxe API

```
long AmGetFieldUserType(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetFieldUserType(hApiField As Long) As Long
```





Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs



	Utilisable
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le champ dont on souhaite connaître le "UserType".

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetForeignKey()

Récupère le handle sur la clé externe d'un lien, lui même identifié par son handle.

Syntaxe API






```
long AmGetForeignKey(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetForeignKey(hApiField As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Handle sur le lien concerné par l'opération.

AmGetIndex()

Cette fonction crée un objet index à partir du handle d'une requête, d'un enregistrement ou d'une table et retourne le handle de l'objet index créé.

Syntaxe API

```
long AmGetIndex(long hApiTable, long lPos);
```

Syntaxe BASIC interne

```
Function AmGetIndex(hApiTable As Long, lPos As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *hApiTable* : Ce paramètre contient un handle sur une table.
- *lPos* : Ce paramètre contient la position de l'index à l'intérieur de la table.

AmGetIndexCount()

Cette fonction renvoie le nombre d'index contenus dans la table précisée dans le paramètre *hApiTable*.

Syntaxe API

```
long AmGetIndexCount(long hApiTable);
```

Syntaxe BASIC interne

```
Function AmGetIndexCount(hApiTable As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *hApiTable* : Ce paramètre contient un handle sur une table.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetIndexField()

Cette fonction renvoie un handle sur un champ identifié par sa position au sein de l'index (le lpos ème champ de l'index).

Syntaxe API



```
long AmGetIndexField(long hApiIndex, long lPos);
```

Syntaxe BASIC interne

```
Function AmGetIndexField(hApiIndex As Long, lPos  
As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiIndex* : Ce paramètre contient un handle valide sur l'index concerné par l'opération.
- *lPos* : Ce paramètre contient la position du champ au sein de l'index.

AmGetIndexFieldCount()

Cette fonction renvoie le nombre de champs qui composent un index.

Syntaxe API






```
long AmGetIndexFieldCount(long hApiIndex);
```

Syntaxe BASIC interne

```
Function AmGetIndexFieldCount(hApiIndex As Long)
As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiIndex* : Ce paramètre contient un handle valide sur l'index concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetIndexFlags()

Cette fonction renvoie les paramètres d'un index.

Syntaxe API





```
long AmGetIndexFlags(long hApiIndex);
```

Syntaxe BASIC interne

```
Function AmGetIndexFlags(hApiIndex As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Entrée**

- *hApiIndex* : Ce paramètre contient un handle valide sur l'index concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques

La valeur retournée par la fonction est la résultante d'une combinaison logique (OR) des valeurs suivantes :

- 1 : l'index autorise les doublons,
- 2 : l'index autorise la valeur nulle,
- 4 : l'index ne respecte pas la casse.

Ainsi, si la fonction renvoie la valeur 3, vous pouvez en déduire que l'index accepte les doublons et la valeur nulle (1 OR 2 = 3).

AmGetIndexName()

Cette fonction renvoie le nom d'un index.

Syntaxe API






```
long AmGetIndexName(long hApiIndex, char
*pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetIndexName(hApiIndex As Long) As
String
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiIndex* : Ce paramètre contient un handle valide sur l'index dont on souhaite connaître le nom.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmGetLink()

Cette fonction crée un objet lien à partir du handle d'une table et retourne le handle de l'objet lien créé.

Syntaxe API






```
long AmGetLink(long hApiTable, long lPos);
```

Syntaxe BASIC interne

```
Function AmGetLink(hApiTable As Long, lPos As Long)
As Long
```

Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle sur une table.
- *lPos* : Ce paramètre contient la position du lien (son index) à l'intérieur de l'objet.

AmGetLinkCardinality()

Cette fonction renvoie la cardinalité d'un lien.

Syntaxe API






```
long AmGetLinkCardinality(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetLinkCardinality(hApiField As Long)
As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le lien dont vous souhaitez connaître la cardinalité.

Sortie

- 1 : Le lien est de cardinalité 1-1.
- 2 : Le lien est de cardinalité 1-n.

AmGetLinkCount()

Cette fonction renvoie le nombre de liens contenus dans la table courante.

Syntaxe API






```
long AmGetLinkCount(long hApiTable);
```

Syntaxe BASIC interne

```
Function AmGetLinkCount(hApiTable As Long) As Long
```

Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle sur une table valide.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetLinkDstField()

Cette fonction renvoie le champ (clé étrangère) sur lequel pointe le lien défini par le paramètre *hApiField*.

Syntaxe API






```
long AmGetLinkDstField(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetLinkDstField(hApiField As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le lien concerné par l'opération.

AmGetLinkFeatureValue()

Renvoie la valeur d'une caractéristique de type "Lien".

Syntaxe API






```
long AmGetLinkFeatureValue(long hApiObject, long
lFieldPos, long lRecordId);
```

Syntaxe BASIC interne

```
Function AmGetLinkFeatureValue(hApiObject As Long,
lFieldPos As Long, lRecordId As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur une requête ou un enregistrement.
- *lFieldPos* : Ce paramètre contient le numéro du champ à l'intérieur de l'objet courant.
- *lRecordId* : Ce paramètre contient le numéro d'identifiant de l'enregistrement dont on veut récupérer la valeur pour la caractéristique.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
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()

Cette fonction crée un objet lien à partir de son nom et retourne le handle de l'objet lien créé.

Syntaxe API






```
long AmGetLinkFromName(long hApiTable, char
*strName);
```

Syntaxe BASIC interne

```
Function AmGetLinkFromName(hApiTable As Long,
strName As String) As Long
```


Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle sur une table.
- *strName* : Ce paramètre contient le nom SQL du lien.

AmGetLinkType()

Cette fonction renvoie le type d'un lien.

Syntaxe API




```
long AmGetLinkType(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetLinkType(hApiField As Long) As Long
```

Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le lien dont on veut connaître le type.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetMainField()

Cette fonction crée un objet champ, correspondant au champ principal d'une table donnée. Elle renvoie un handle sur le champ ainsi créé.

Syntaxe API






```
long AmGetMainField(long hApiTable);
```

Syntaxe BASIC interne

```
Function AmGetMainField(hApiTable As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle sur la table dont on cherche le champ principal.

AmGetNextAssetPin()

L'API AmGetNextAssetPin trouve la broche disponible suivante sur un dispositif (lAssetId). Son numéro de séquence trie les broches. Suivant la direction du port (bPinPortDir), les broches disponibles sont triées par ordre croissant (bPinPortDir = 0) ou décroissant (bPinPortDir = 1).

Syntaxe API

```
long AmGetNextAssetPin(long hApiCnxBase, long
lAssetId, long bPinPortDir, long iPrevPinSeq);
```

Syntaxe BASIC interne

```
Function AmGetNextAssetPin(lAssetId As Long,
bPinPortDir As Long, iPrevPinSeq As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *lAssetId* : ce paramètre contient l'identifiant du dispositif.
- *bPinPortDir* : ce paramètre est la direction suivant laquelle chercher.
 - 0=croissant
 - 1=décroissant
- *iPrevPinSeq*

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetNextAssetPort()

L'API `AmGetNextAssetPort` trouve le port disponible suivant sur un dispositif (`lAssetId`) fournissant une fonction donnée (`lDutyId`) ou pas de fonction du tout. L'état du port doit être "Disponible". Des drapeaux booléens précisent si le côté utilisateur (`bCheckUser`) et/ou le côté hôte (`bCheckHost`) du port doivent être vérifiés. L'API compare la valeur utilisateur (`bUserAvail`) et/ou les valeurs hôte (`bHostAvail`) si le drapeau

booléen prend la valeur vraie. Les ports sont triés selon leur numéro de séquence. Suivant la direction du port (bPinPortDir), les ports disponibles sont triés par ordre croissant (bPinPortDir = 0) ou décroissant (bPinPortDir = 1).

Syntaxe API






```
long AmGetNextAssetPort(long hApiCnxBase, long
lAssetId, long lCabCnxTypeId, long lDutyId, long
bCheckUser, long bCheckHost, long bUserAvail, long
bHostAvail, long bPinPortDir, long iPrevPortSeq);
```

Syntaxe BASIC interne

```
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
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lAssetId* : ce paramètre contient l'identifiant du dispositif à chercher.

- *lCabCnxTypeId* : ce paramètre contient l'identifiant du type de connexion de câble pour le port.
- *lDutyId* : ce paramètre contient l'identifiant de la fonction du port.
- *bCheckUser* : ce paramètre est un drapeau vérifiant le côté utilisateur.
- *bCheckHost* : ce paramètre est un drapeau vérifiant le côté hôte.
- *bUserAvail* : ce paramètre définit l'état de disponibilité du côté utilisateur à vérifier.
- *bHostAvail* : ce paramètre définit l'état de disponibilité du côté hôte à vérifier.
- *bPinPortDir* : ce paramètre définit la direction de broche à vérifier.
 - 0=croissant
 - 1=décroissant
- *iPrevPortSeq*

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetNextCableBundle()

L'API `AmGetNextCableBundle` trouve le faisceau disponible suivant sur un câble (`lCableId`) fournissant une fonction donnée (`lDutyId`) ou pas de fonction du tout. L'état du faisceau doit être "Disponible". Des drapeaux booléens précisent si le côté utilisateur (`bCheckUser`) et/ou le côté hôte (`bCheckHost`) du faisceau doivent être vérifiés. L'API

compare la valeur utilisateur (`bUserAvail`) et/ou les valeurs hôte (`bHostAvail`) si le drapeau booléen prend la valeur vraie.

Syntaxe API






```
long AmGetNextCableBundle(long hApiCnxBase, long
lCableId, long lDutyId, long bCheckUser, long
bCheckHost, long bUserAvail, long bHostAvail);
```

Syntaxe BASIC interne

```
Function AmGetNextCableBundle(lCableId As Long,
lDutyId As Long, bCheckUser As Long, bCheckHost As
Long, bUserAvail As Long, bHostAvail As Long) As
Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- `lCableId` : ce paramètre contient l'identifiant du câble à vérifier.
- `lDutyId` : ce paramètre contient l'identifiant de la fonction à localiser.
- `bCheckUser` : ce paramètre établit de vérifier la connexion du faisceau du côté utilisateur .
- `bCheckHost` : ce paramètre établit de vérifier la connexion du faisceau du côté hôte.

- *bUserAvail* : ce paramètre définit l'état de connexion du côté utilisateur à localiser.
- *bHostAvail* : ce paramètre définit l'état de connexion du côté hôte à localiser.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetNextCablePair()

L'API `AmGetNextCablePair` trouve la paire de câble disponible suivante dans un câble (`lCableId`) d'un type donné (`lPairTypeId`). Les paires sont triées par identifiant de paire de câble.

Syntaxe API






```
long AmGetNextCablePair(long hApiCnxBase, long  
lCableId, long lPairTypeId, long iStartPairSeq);
```

Syntaxe BASIC interne

```
Function AmGetNextCablePair(lCableId As Long,  
lPairTypeId As Long, iStartPairSeq As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCableId* : ce paramètre contient l'identifiant du câble à chercher.
- *lPairTypeId* : ce paramètre définit le type de paire du câble à localiser.
- *iStartPairSeq*

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetNTDomains()

Cette fonction permet de récupérer le domaine de l'utilisateur connecté à la base de données.

Syntaxe API






```
long AmGetNTDomains(char *pstrDomains, long lDomains);
```

Syntaxe BASIC interne

```
Function AmGetNTDomains() As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmGetNTMachinesInDomain()

Cette fonction permet de récupérer la liste des machines d'un domaine en une colonne (noms des machines séparés par des virgules). Si le domaine est vide, la fonction retourne ERR_CANCEL(2), mais l'exécution n'est pas interrompue.

Syntaxe API






```
long AmGetNTMachinesInDomain(char *strDomain, char *pstrMachines, long lMachines);
```

Syntaxe BASIC interne

```
Function AmGetNTMachinesInDomain(strDomain As
String) As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strDomain* : Ce paramètre contient le nom du domaine à explorer.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetNTUsersInDomain()

Cette fonction permet de récupérer la liste des utilisateurs sur un domaine. La liste est retournée en deux colonnes (login,fullname). '|' est utilisé comme séparateur de colonnes, '\n' comme séparateur de lignes.

Syntaxe API






```
long AmGetNTUsersInDomain(char *strDomain, char
*pstrUsers, long lUsers);
```

Syntaxe BASIC interne

```
Function AmGetNTUsersInDomain(strDomain As String)
As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strDomain* : Ce paramètre contient le nom du domaine à explorer.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetPOLinePrice()

Cette fonction permet de calculer le prix d'une ligne de commande.

Syntaxe API




```
double AmGetPOLinePrice(long hApiCnxBase, long
lPOrdLineId);
```

Syntaxe BASIC interne

```
Function AmGetPOLinePrice(lPOrdLineId As Long) As
Double
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrdLineId* : Ce paramètre contient l'identifiant de la ligne de commande.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetPOLinePriceCur()

Cette fonction permet de retrouver le code devise applicable à une ligne de commande

Syntaxe API




```
long AmGetPOLinePriceCur(long hApiCnxBase, long
lPordLineId, char *pstrPrice, long lPrice);
```

Syntaxe BASIC interne

```
Function AmGetPOLinePriceCur(lPordLineId As Long)
As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPordLineId* : Ce paramètre contient l'identifiant de la ligne de commande.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmGetPOLinePricing()

Syntaxe API




```
double AmGetPOLinePricing(long hApiCnxBase, long
lPOrdLineId, char *pstrPriceCur, char *pstrRef);
```

Syntaxe BASIC interne

```
Function AmGetPOLinePricing(lPOrdLineId As Long,
pstrPriceCur As String, pstrRef As String) As
Double
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmGetPOLineReference()

Cette fonction permet de récupérer le libellé de la référence catalogue correspondant à la ligne de commande.

Syntaxe API




```
long AmGetPOLineReference(long hApiCnxBase, long
lPordLineId, char *pstrRef, long lRef);
```

Syntaxe BASIC interne

```
Function AmGetPOLineReference(lPordLineId As Long)
As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPOrdLineId* : Ce paramètre contient l'identifiant de la ligne de commande.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetRecordFromMainId()

Cette fonction renvoie le numéro d'identifiant d'un enregistrement identifié par une valeur de la clé primaire de la table contenant cet enregistrement.

Syntaxe API




```
long AmGetRecordFromMainId(long hApiCnxBase, char  
*strTable, long lId);
```

Syntaxe BASIC interne

```
Function AmGetRecordFromMainId(strTable As String,  
lId As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTable* : Ce paramètre contient le nom SQL de la table contenant l'enregistrement concerné.
- *lId* : Ce paramètre contient la valeur de la clé primaire de la table pour cet enregistrement.

Remarques

Cette fonction renvoie systématiquement un handle d'enregistrement, excepté lorsque la table n'existe pas. S'il n'existe aucun enregistrement dans la table spécifiée, une erreur surviendra à chaque nouvelle exécution de fonction utilisant le handle renvoyé par cette fonction.

AmGetRecordHandle()

Cette fonction renvoie le handle d'un enregistrement qui est le résultat courant d'une requête identifiée par son handle. Cet enregistrement pourra être utilisé pour écrire dans la base de données. Cette fonction n'est opérante que si la requête contient la clé primaire de l'enregistrement.

Syntaxe API




```
long AmGetRecordHandle(long hApiQuery);
```

Syntaxe BASIC interne

```
Function AmGetRecordHandle(hApiQuery As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur un objet requête.

AmGetRecordId()

Cette fonction renvoie le numéro d'identifiant d'un enregistrement identifié par son handle. Dans le cas d'un enregistrement en cours d'insertion, cette valeur sera 0.

Syntaxe API




```
long AmGetRecordId(long hApiRecord);
```

Syntaxe BASIC interne

```
Function AmGetRecordId(hApiRecord As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient un handle valide sur l'enregistrement dont on souhaite connaître le numéro d'identifiant.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetRelDstField()

Cette fonction renvoie un handle sur le champ destination d'un lien.

Syntaxe API






```
long AmGetRelDstField(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetRelDstField(hApiField As Long) As
Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le lien concerné par l'opération.

AmGetRelSrcField()

Cette fonction renvoie un handle sur le champ source d'un lien.

Syntaxe API






```
long AmGetRelSrcField(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetRelSrcField(hApiField As Long) As
Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le lien concerné par l'opération.

AmGetRelTable()

Cette fonction renvoie un handle sur la table de relation d'un lien de cardinalité N-N.

Syntaxe API



```
long AmGetRelTable(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetRelTable(hApiField As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le lien concerné par l'opération.

Sortie

En cas d'erreur, cette fonction retourne un handle non valide (de valeur nulle).

AmGetReverseLink()

Cette fonction renvoie le handle du lien inverse du lien spécifié par le handle contenu dans le paramètre *hApiField*.

Syntaxe API



```
long AmGetReverseLink(long hApiField);
```



Syntaxe BASIC interne

```
Function AmGetReverseLink(hApiField As Long) As  
Long
```

Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le lien dont on veut connaître le lien inverse.

AmGetSelfFromMainId()

Renvoie la chaîne de description pour un enregistrement d'une table donnée.

Syntaxe API






```
long AmGetSelfFromMainId(long hApiCnxBase, char
*strTableName, long lId, char *pstrRecordDesc, long
lRecordDesc);
```

Syntaxe BASIC interne

```
Function AmGetSelfFromMainId(strTableName As
String, lId As Long) As String
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTableName* : Ce paramètre contient le nom SQL de la table contenant l'enregistrement concerné par l'opération.
- *lId* : Ce paramètre contient le numéro d'identifiant de l'enregistrement concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetSourceTable()

Renvoie le handle de la table source du lien indiqué dans le paramètre *hApiField*.

Syntaxe API






```
long AmGetSourceTable(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetSourceTable(hApiField As Long) As  
Long
```

Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le lien dont on veut connaître la table source.

Sortie

En cas d'erreur, cette fonction retourne un handle non valide (de valeur nulle).

AmGetTable()

Cette fonction renvoie le handle d'une table identifiée par sa position (son numéro) dans la connexion courante.

Syntaxe API

```
long AmGetTable(long hApiCnxBase, long lPos);
```





Syntaxe BASIC interne

```
Function AmGetTable(lPos As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPos* : Ce paramètre contient la position de la table dans la connexion courante. Ses valeurs sont comprises entre "0" et AmGetTableCount.

Sortie

En cas d'erreur, cette fonction retourne un handle non valide (de valeur nulle).

AmGetTableCount()

Cette fonction renvoie le nombre de tables de la base de données sur laquelle porte la connexion courante.

Syntaxe API

```
long AmGetTableCount(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmGetTableCount() As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetTableDescription()

Cette fonction renvoie, sous la forme d'une chaîne de caractères (format "String"), la description longue d'une table identifiée par un handle.

Syntaxe API






```
long AmGetTableDescription(long hApiTable, char
*pstrDesc, long lDesc);
```

Syntaxe BASIC interne

```
Function AmGetTableDescription(hApiTable As Long)
As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle valide sur la table dont on souhaite connaître la description longue.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetTableFromName()

Cette fonction renvoie le handle d'une table identifiée par son nom SQL dans la connexion courante.

Syntaxe API






```
long AmGetTableFromName(long hApiCnxBase, char *strName);
```

Syntaxe BASIC interne

```
Function AmGetTableFromName(strName As String) As
Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strName* : Ce paramètre contient la position le nom SQL de la table dont on veut récupérer le handle.

Sortie

En cas d'erreur, cette fonction retourne un handle non valide (de valeur nulle).

AmGetTableLabel()

Cette fonction renvoie, sous la forme d'une chaîne de caractères (format "String"), le libellé d'une table identifiée par un handle.

Syntaxe API






```
long AmGetTableLabel(long hApiTable, char
*pstrLabel, long lLabel);
```

Syntaxe BASIC interne

```
Function AmGetTableLabel(hApiTable As Long) As
String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle valide sur la table dont on souhaite connaître le libellé.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmGetTableName()

Renvoie le nom SQL d'une table sous la forme d'une chaîne de caractères.

Syntaxe API






```
long AmGetTableName(long hApiTable, char
*pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetTableName(hApiTable As Long) As
String
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Handle valide sur la table dont vous souhaitez récupérer le nom.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmGetTableRights()

Cette fonction renvoie, sous la forme d'une chaîne de caractères (format "String"), les droits utilisateur sur une table identifiée par un handle. La chaîne renvoyée est composée au maximum de deux caractères qui indiquent l'état des droits en création et en destruction :

- "c" indique que l'utilisateur possède les droits en création sur la table.
- "d" indique que l'utilisateur possède les droits en destruction sur la table.

Ainsi, par exemple :

- "c" indique que l'utilisateur possède uniquement un droit en création sur la table.
- "cd" indique que l'utilisateur possède les droits en création et en destruction sur la table.

Syntaxe API






```
long AmGetTableRights(long hApiTable, char
*pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetTableRights(hApiTable As Long) As
String
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle valide sur la table pour laquelle on souhaite connaître les droits utilisateurs.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetTableSqlName()

Cette fonction renvoie, sous la forme d'une chaîne de caractères (format "String"), le nom SQL d'une table identifiée par un handle.

Syntaxe API

```
long AmGetTableSqlName(long hApiTable, char
*pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmGetTableSqlName(hApiTable As Long) As
String
```





Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs



	Utilisable
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiTable* : Ce paramètre contient un handle valide sur la table dont on souhaite connaître le nom SQL.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetTargetTable()

Retourne le nom SQL de la table de destination d'un lien.

Syntaxe API






```
long AmGetTargetTable(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetTargetTable(hApiField As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Handle sur le lien concerné par l'opération.

Sortie

En cas d'erreur, cette fonction retourne un handle non valide (de valeur nulle).

AmGetTrace()

L'API AmGetTrace détecte la chaîne de liaisons entre deux noeuds (lUserId, lHostId) dans la table des liaisons de câbles. La direction de la chaîne de liaisons (iTraceDir) précise si la chaîne de liaisons doit être utilisateur vers hôte (iTraceDir = 1) ou hôte vers utilisateur (iTraceDir = 0). Le type de chaîne de liaisons (iTraceType) indique si la chaîne de liaisons est une connexion (iTraceType = 1) ou une déconnexion (iTraceType = 2). L'indicateur de chaîne de liaisons complète (bFullTrace) précise si la chaîne de liaisons inclut seulement des noeuds modifiés (bFullTrace=0) ou la chaîne de liaisons complète (bFullTrace=1)

Syntaxe API






```
long AmGetTrace(long hApiCnxBase, long lUserId,
long lHostId, long iTraceDir, long iTraceType, long
bFullTrace, char *pstrTrace, long lTrace);
```

Syntaxe BASIC interne

```
Function AmGetTrace(lUserId As Long, lHostId As
Long, iTraceDir As Long, iTraceType As Long,
bFullTrace As Long) As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lUserId* : ce paramètre contient l'identifiant de la liaison de connexion de départ.
- *lHostId* : ce paramètre contient l'identifiant de la liaison de connexion d'arrivée.
- *iTraceDir* : ce paramètre précise la direction de la connexion.
 - 0=hôte vers utilisateur
 - 1=utilisateur vers hôte
- *iTraceType* : ce paramètre précise le type de connexion.
 - 1=connexion
 - 2=déconnexion

- *bFullTrace* : ce paramètre spécifie d'ignorer la chaîne de liaisons partielle et de retourner la chaîne de la chaîne de liaisons entière.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetTraceFromHist()

L'API `AmGetTraceFromHist` sert à calculer une chaîne de caractères à partir de l'historique de chaîne de liaisons et à l'aide des opérations sur chaîne de liaisons afin de distinguer la connectivité nouvelle de celle existante.

Syntaxe API




```
long AmGetTraceFromHist(long hApiCnxBase, long  
lProjTraceOutId, long iTraceDir, char  
*strDelimiter, char *pstrTraceint, long lTraceint,  
long bUpdateFlag);
```

Syntaxe BASIC interne

```
Function AmGetTraceFromHist(lProjTraceOutId As  
Long, iTraceDir As Long, strDelimiter As String,  
bUpdateFlag As Long) As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lProjTraceOutId* : ce paramètre contient l'identifiant de la chaîne de liaisons du projet.
- *iTraceDir* : ce paramètre précise la direction de la connexion.
 - 0=hôte vers utilisateur
 - 1=utilisateur vers hôte
- *strDelimiter* : ce paramètre est le délimiteur de chaînes qui montre les connexions et déconnexions existantes.
- *bUpdateFlag* : ce paramètre optionnel met à jour le champ `amCabTraceOut.TraceString`.
 - 0=faux
 - 1=vrai

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmGetTypedLinkField()

Renvoie un handle sur le champ dont la valeur est le nom SQL de la table de destination du lien typé indiqué dans le paramètre *hApiField*.

Syntaxe API






```
long AmGetTypedLinkField(long hApiField);
```

Syntaxe BASIC interne

```
Function AmGetTypedLinkField(hApiField As Long) As Long
```

Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le lien typé à l'origine de l'opération.

AmGetVersion()

Cette fonction renvoie le numéro de compilation de la version d'AssetCenter sous la forme d'une chaîne de caractères.

Syntaxe API






```
long AmGetVersion(char *pstrBuf, long lBuf);
```

Syntaxe BASIC interne

```
Function AmGetVersion() As String
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmHasAdminPrivilege()

Cette fonction renvoie la valeur "TRUE" (valeur différente de 0) si l'utilisateur connecté possède les droits administratifs.

Syntaxe API






```
long AmHasAdminPrivilege(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmHasAdminPrivilege() As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmHasRelTable()

Cette fonction permet de tester si un lien possède ou non une table de relation.

Syntaxe API






```
long AmHasRelTable(long hApiField);
```

Syntaxe BASIC interne

```
Function AmHasRelTable(hApiField As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle valide sur le lien concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmImportDocument()

Syntaxe API




```
long AmImportDocument(long hApiCnxBase, long
lDocObjId, char *strTableName, char *strFileName,
char *strCategory, char *strDesignation);
```

Syntaxe BASIC interne

```
Function AmImportDocument(lDocObjId As Long,
strTableName As String, strFileName As String,
strCategory As String, strDesignation As String)
As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmIncrementLogLevel()

Cette fonction affiche le message *strMsg* dans une fenêtre d'historique et crée un noeud dans la page finale d'un assistant.


Tous les prochains messages apparaîtront sous ce noeud.

Syntaxe BASIC interne

```
Function AmIncrementLogLevel(strMsg As String,
iType As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strMsg* : Ce paramètre contient le texte du message à afficher.
- *iType* : Ce paramètre définit l'icône associée au message. Les valeurs possibles sont "1" pour une erreur, "2" pour un avertissement et "4" pour une information.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmInsertRecord()

Cette fonction insère un enregistrement précédemment créé dans la base de données. Seuls les enregistrements créés au moyen de la fonction `AmCreateRecord` peuvent être insérés dans la base de données. Les enregistrements accédés au moyen d'une requête ne peuvent être insérés.

Syntaxe API




```
long AmInsertRecord(long hApiRecord);
```

Syntaxe BASIC interne

```
Function AmInsertRecord(hApiRecord As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient un handle sur l'enregistrement que vous souhaitez insérer dans la base de données.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmInstantiateReqLine()

Cette fonction permet d'instancier directement une ligne de demande donnée.

Syntaxe API




```
long AmInstantiateReqLine(long hApiCnxBase, long
lRequestId, long bFinal, long lPOrderLineId,
float fQty);
```

Syntaxe BASIC interne

```
Function AmInstantiateReqLine(lRequestId As
Long, bFinal As Long, lPOrderLineId As Long, fQty
As Single) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lRequestLineId* : Ce paramètre contient l'identifiant de la ligne de demande.
- *bFinal* : Ce paramètre permet de préciser si oui ou non vous souhaitez finaliser l'affectation.
- *lPOrderLineId* : Ce paramètre contient l'identifiant de la ligne de commande.
- *fQty* : Ce paramètre contient la quantité à instancier.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Remarques

La fonction permet de créer les éléments demandés sans passer par le cycle d'achat. Si *bFinal* = FALSE, alors l'élément sera créé avec l'état d'affectation En attente de réception.

AmInstantiateRequest()

Cette fonction permet d'instancier directement le contenu complet d'une demande donnée.

Syntaxe API




```
long AmInstantiateRequest(long hApiCnxBase, long  
lRequestId, long lMulFactor);
```

Syntaxe BASIC interne

```
Function AmInstantiateRequest(lRequestId As Long,  
lMulFactor As Long) As Long
```


Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lRequestId* : Ce paramètre contient l'identifiant de demande.
- *lMulFactor* : Ce paramètre permet de préciser le nombre d'instanciations à effectuer.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmIsConnected()

Cette fonction teste si la connexion courante est valide.

Syntaxe API

```
long AmIsConnected(long hApiCnxBase);
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	

Utilisable

Script de configuration d'un champ ou d'un lien

Action de type "Script"

Script d'un assistant

Script FINISH.DO d'un assistant

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmlsFieldForeignKey()

Cette fonction permet de déterminer si un champ est une clé étrangère de la base de données.

Syntaxe API






```
long AmIsFieldForeignKey(long hApiField);
```

Syntaxe BASIC interne

```
Function AmIsFieldForeignKey(hApiField As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le champ qui doit être identifié.

Sortie

- 1 : Le champ est une clé étrangère.
- 0 : Le champ n'est pas une clé étrangère.

AmlsFieldIndexed()

Cette fonction permet de déterminer si un champ est indexé ou non.

Syntaxe API

```
long AmIsFieldIndexed(long hApiField);
```

Syntaxe BASIC interne

```
Function AmIsFieldIndexed(hApiField As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le champ qui doit être identifié.

Sortie

- 1 : Le champ est indexé.
- 0 : Le champ n'est pas indexé.

AmlsFieldPrimaryKey()

Cette fonction permet de déterminer si un champ est une clé primaire de la base de données.

Syntaxe API






```
long AmIsFieldPrimaryKey(long hApiField);
```

Syntaxe BASIC interne

```
Function AmIsFieldPrimaryKey(hApiField As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Ce paramètre contient un handle sur le champ qui doit être identifié.

Sortie

- 1 : Le champ est une clé primaire.
- 0 : Le champ n'est pas une clé primaire.

AmlsLink()

Détermine si l'objet identifié par son handle est un lien ou un champ.

Syntaxe API

```
long AmIsLink(long hApiField);
```

Syntaxe BASIC interne

```
Function AmIsLink(hApiField As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *hApiField* : Handle sur l'objet concerné par l'opération.

Sortie

- 1 : L'objet est un lien.
- 0 : L'objet est un champ.

AmlsTypedLink()

Détermine si l'objet identifié par son handle est un lien typé ou non.

Syntaxe API

```
long AmlsTypedLink(long hApiField);
```

Syntaxe BASIC interne

```
Function AmlsTypedLink(hApiField As Long) As Long
```

Champ d'application

Version : 3.02

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiField* : Handle sur l'objet concerné par l'opération.

Sortie

- 1 : L'objet est un lien typé.
- 0 : L'objet n'est pas un lien typé.

AmLastError()

Cette fonction renvoie le dernier code d'erreur généré par la dernière fonction exécutée dans le contexte de la connexion correspondante.

Syntaxe API





```
long AmLastError(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmLastError() As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Sortie**

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmLastErrorMsg()

Cette fonction renvoie le dernier message d'erreur survenu lors de la connexion courante.

Syntaxe API

```
long AmLastErrorMsg(long hApiCnxBase, char
*pstrBuffer, long lBuffer);
```

Syntaxe BASIC interne

```
Function AmLastErrorMsg() As String
```

Champ d'application

Version : 2.52

Utilisable**AssetCenter APIs****Script de configuration d'un champ ou d'un lien****Action de type "Script"**

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmListToString()

Cette fonction convertit le résultat d'une chaîne de caractères obtenue au moyen de la fonction `AmDbGetList` en une chaîne de caractères affichable telle qu'elle apparaîtrait avec la fonction `AmDbGetString`.

Syntaxe API

```
long AmListToString(char *return, long lreturn,
char *strSource, char *strColSep, char *strLineSep,
char *strIdSep);
```

Syntaxe BASIC interne

```
Function AmListToString(strSource As String,
strColSep As String, strLineSep As String, strIdSep
As String) As String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strSource* : Ce paramètre contient la chaîne de caractères à convertir.
- *strColSep* : Ce paramètre contient le caractère utilisé comme séparateur de colonnes dans la chaîne à convertir.
- *strLineSep* : Ce paramètre contient le caractère utilisé comme séparateur de lignes dans la chaîne à convertir.
- *strIdSep* : Ce paramètre contient le caractère utilisé comme séparateur d'identifiant dans la chaîne à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmLog()


Cette fonction affiche le message *strMessage* dans une fenêtre d'historique.

Syntaxe BASIC interne

```
Function AmLog(strMessage As String, iLogType As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strMessage* : Ce paramètre contient le texte du message à afficher.
- *iLogType* : Ce paramètre définit l'icône associée au message. Les valeurs possibles sont "1" pour une erreur, "2" pour un avertissement et "4" pour une information.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
AmLog("Ceci est un message")
```

AmLoginId()

Cette fonction renvoie l'identifiant de l'utilisateur connecté.

Syntaxe API






```
long AmLoginId(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmLoginId() As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant définit l'identifiant de l'utilisateur connecté comme valeur par défaut pour un champ de la base de données :

```
RetVal=AmLoginId( )
```

AmLoginName()

Cette fonction renvoie le nom de login de l'utilisateur connecté.

Syntaxe API






```
long AmLoginName(long hApiCnxBase, char *return,
long lreturn);
```

Syntaxe BASIC interne

```
Function AmLoginName() As String
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant définit le nom de login de l'utilisateur connecté comme valeur par défaut pour un champ de la base de données :

```
RetVal=AmLoginName()
```

AmMapSubReqLineAgent()

Cette fonction permet d'établir les liens possibles entre les sous-lignes d'une ligne de demande et celles d'une ligne de commande.

Syntaxe API






```
long AmMapSubReqLineAgent(long hApiCnxBase, long lRequestLineId, long lPorderLineId);
```

Syntaxe BASIC interne

```
Function AmMapSubReqLineAgent(lRequestLineId As Long, lPorderLineId As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lRequestLineId* : Ce paramètre contient l'identifiant de la ligne de demande.
- *lPorderLineId* : Ce paramètre contient l'identifiant de la ligne de demande.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmMoveCable()

L'API AmMoveCable transfère un câble (lCableId) de sa localisation actuelle à une localisation de destination donnée (lToLocId). Si le projet (lProjectId) et l'intervention (lWorkOrderId) prennent des valeurs, le câble est ajouté au projet et à l'intervention avec le commentaire donné (strComment). Ce commentaire décrit l'action qui sera accomplie sur le câble (i.e. "Transférer le câble d'ici jusqu'à là").

Syntaxe API




```
long AmMoveCable(long hApiCnxBase, long lCableId,
long lToLocId, long lProjectId, long lWorkOrderId,
char *strComment);
```

Syntaxe BASIC interne

```
Function AmMoveCable(lCableId As Long, lToLocId As
Long, lProjectId As Long, lWorkOrderId As Long,
strComment As String) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCableId* : ce paramètre contient l'identifiant du câble à transférer.
- *lToLocId* : ce paramètre contient l'identifiant de la nouvelle localisation du câble.
- *lProjectId* : ce paramètre contient l'identifiant du projet.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmMoveDevice()

L'API AmMoveDevice transfère un dispositif (IDeviceId) de sa localisation actuelle jusqu'à une localisation de destination donnée (lToLocationId). Si le projet (lProjectId) et l'intervention (lWorkOrderId) prennent des valeurs, le dispositif est ajouté au projet et à l'intervention avec le commentaire donné (strComment). Ce commentaire décrit

l'action qui sera accomplie sur le dispositif (i.e. "Transférer le dispositif d'ici jusqu'à là").

Syntaxe API




```
long AmMoveDevice(long hApiCnxBase, long lDeviceId,
long lToLocationId, long lProjectId, long
lWorkOrderId, char *strComment);
```

Syntaxe BASIC interne

```
Function AmMoveDevice(lDeviceId As Long,
lToLocationId As Long, lProjectId As Long,
lWorkOrderId As Long, strComment As String) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lDeviceId* : ce paramètre contient l'identifiant du dispositif qui sera transféré.
- *lToLocationId* : ce paramètre contient l'identifiant de la nouvelle localisation du dispositif.
- *lProjectId* : ce paramètre contient l'identifiant du projet.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention.

- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmMsgBox()





Cette fonction affiche une boîte de dialogue contenant un message.

Syntaxe BASIC interne

```
Function AmMsgBox(strMessage As String, lMode As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strMessage* : Ce paramètre contient le message affiché dans la boîte de dialogue.
- *lMode* : Ce paramètre contient le type de boîte de dialogue affiché (0 pour une boîte simple avec un bouton OK, 1 pour une boîte avec

les boutons OK et Annuler, 2 pour une boîte avec le seul bouton Annuler).

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
AmMsgBox("Déménagement effectué")
```

AmOpenConnection()

Crée une connexion sur une base de données AssetCenter. *strDataSource* doit être une source de données valide (les sources de données apparaissent dans la boîte de connexion d'AssetCenter). Vous pouvez ouvrir plusieurs connexions sur une même base ou sur des bases de données différentes.

Syntaxe API

```
long AmOpenConnection(char *strDataSource, char *strUser, char *strPwd);
```

Champ d'application

Version : 2.52

Utilisable

AssetCenter APIs



Script de configuration d'un champ ou d'un lien

Action de type "Script"

Script d'un assistant

Script FINISH.DO d'un assistant

Entrée

- *strDataSource* : Nom de la source de données pour la connexion.
- *strUser* : Nom de l'utilisateur pour le connexion.
- *strPwd* : Mot de passe de l'utilisateur sur la base de données.

AmOpenScreen()




Cette fonction permet d'ouvrir un écran sous AssetCenter.

Syntaxe BASIC interne

```
Function AmOpenScreen(strScreenId As String,
strContext As String, strFilter As String, iMode
As Long, strBindField As String) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strScreenId* : Ce paramètre contient le nom SQL de la vue de l'écran système ou utilisateur que vous souhaitez ouvrir (dans cet ordre de priorité).
- *strContext* : Ce paramètre optionnel contient la liste des identifiants des enregistrements sélectionnés dans la liste à l'ouverture de l'écran.

- *strFilter* : Ce paramètre contient un filtre AQL appliqué sur la liste à l'ouverture de l'écran.
- *iMode* : Ce paramètre contient le mode d'ouverture de l'écran : consultation, édition, etc. Les valeurs possibles sont : 0 (Pas d'action en cours), 1 (Pas d'action en cours), 2 (Modification en cours), 3 (Création en cours), 4 (Duplication en cours), 5 (Ajout en cours), 6 (Choix en cours).
- *strBindField* : Ce paramètre permet d'ouvrir un écran avec un filtre et un mode comme pour l'ouverture d'une fenêtre liée. Il prend le nom SQL du champ source ou bien la valeur CurrentSrcChoice pour utiliser le contexte en cours.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmPagePath()

Cette fonction renvoie, sous la forme d'une chaîne, le chemin de l'assistant, c'est-à-dire la liste des pages parcourues sans tenir compte des retours en arrière.

Syntaxe BASIC interne

```
Function AmPagePath() As String
```

Champ d'application


Version : 3.00

Utilisable

AssetCenter APIs

Script de configuration d'un champ ou d'un lien

Action de type "Script"

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmProgress()


Cette fonction affiche, dans la page finale d'un assistant, une barre de progression représentant un pourcentage.

Syntaxe BASIC interne

```
Function AmProgress(iProgress As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *iProgress* : Ce paramètre contient le pourcentage (entre 0 et 100) de complétion qui détermine la taille de la barre de progression.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
AmProgress(85)
```

Cette fonction affiche une barre de progression représentant un pourcentage de 85%.

AmQueryCreate()

Cette fonction crée un objet requête dans la connexion courante. Cet objet peut ensuite être utilisé pour envoyer des commandes AQL au serveur de base de données.

Syntaxe API

```
long AmQueryCreate(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmQueryCreate() As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

AmQueryExec()

Cette fonction exécute une requête AQL. Elle renvoie le premier résultat de la requête. Le résultat suivant peut être obtenu au moyen de la fonction AmQueryNext.

Lorsque la requête transmise par cette fonction renvoie un champ de type "Memo" (enregistrement de la table de nom SQL amComment), la taille de ce dernier est tronquée à 255 caractères.

Syntaxe API

```
long AmQueryExec(long hApiQuery, char
*strQueryCommand);
```

Syntaxe BASIC interne

```
Function AmQueryExec(hApiQuery As Long,
strQueryCommand As String) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur l'objet requête auquel sont transmises les commandes AQL.
- *strQueryCommand* : Ce paramètre contient le corps de la requête AQL sous forme de chaîne.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmQueryGet()

Cette fonction exécute une requête AQL sans curseur (un seul résultat). Elle ne renvoie qu'une seule ligne de résultats.

Syntaxe API




```
long AmQueryGet(long hApiQuery, char
*strQueryCommand);
```

Syntaxe BASIC interne

```
Function AmQueryGet(hApiQuery As Long,
strQueryCommand As String) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur l'objet requête auquel sont transmises les commandes AQL.
- *strQueryCommand* : Ce paramètre contient le corps de la requête AQL sous forme de chaîne.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmQueryNext()

Cette fonction renvoie le résultat suivant d'une requête préalablement exécutée au moyen de la fonction AmQueryExec.

Syntaxe API

```
long AmQueryNext(long hApiQuery);
```





Syntaxe BASIC interne

```
Function AmQueryNext(hApiQuery As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur l'objet requête auquel sont transmises les commandes AQL.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmQuerySetAddMainField()

Cette fonction permet de passer une requête dans un mode où le champ principal de la table est automatiquement ajouté à la liste des champs à retourner. Une telle requête ne retournera jamais l'enregistrement d'identifiant nul.

Syntaxe API






```
long AmQuerySetAddMainField(long hApiQuery, long
bAddMainField);
```

Syntaxe BASIC interne

```
Function AmQuerySetAddMainField(hApiQuery As Long,
bAddMainField As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur un objet requête.
- *bAddMainField* : Ce paramètre peut avoir deux valeurs :
 - True : Le champ principal de la table est ajouté,
 - False : Le champ principal de la table n'est pas ajouté.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmQuerySetFullMemo()

Par défaut, lors de l'exécution de la fonction `AmQueryExec`, la requête tronque les champs de type Memo à 254 caractères. Cette fonction passe la requête dans un mode où elle ramènera les valeurs des champs Memo dans leur intégralité.

Syntaxe API

```
long AmQuerySetFullMemo(long hApiQuery, long bFullMemo);
```

Syntaxe BASIC interne

```
Function AmQuerySetFullMemo(hApiQuery As Long,
    bFullMemo As Long) As Long
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur un objet requête.
- *bFullMemo* : Ce paramètre peut avoir deux valeurs :
 - True : La requête renvoie l'intégralité du champ Memo,
 - False : La requête tronque les champs Memo à 254 caractères.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmQueryStartTable()

Cette fonction renvoie un handle sur la table sur laquelle porte une requête identifiée par son handle.

Syntaxe API






```
long AmQueryStartTable(long hApiQuery);
```

Syntaxe BASIC interne

```
Function AmQueryStartTable(hApiQuery As Long) As  
Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur un objet requête.

Sortie

En cas d'erreur, cette fonction retourne un handle non valide (de valeur nulle).

AmQueryStop()

Cette fonction interrompt l'exécution d'une requête identifiée par son handle. Cette requête doit avoir été préalablement lancée au moyen de la fonction AmQueryExec.

Syntaxe API






```
long AmQueryStop(long hApiQuery);
```

Syntaxe BASIC interne

```
Function AmQueryStop(hApiQuery As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiQuery* : Ce paramètre contient un handle valide sur un objet requête.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmReceiveAllPOLines()

Cette fonction effectue la réception de tous les éléments sur une ligne de commande.



Note : Attention : les lignes de réception sont créées par un agent au moment du "commit" de la transaction. Vous ne pouvez pas y accéder avant.

Syntaxe API

```
long AmReceiveAllPOLines(long hApiCnxBase, long
lPordId, long lDelivId);
```

Syntaxe BASIC interne

```
Function AmReceiveAllPOLines(lPordId As Long,
lDelivId As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPordId* : Ce paramètre contient l'identifiant de la ligne de commande contenant les éléments à réceptionner.

- *lDelivId* : Ce paramètre contient l'identifiant de la fiche de réception qui recevra tous les éléments présents sur la ligne de commande.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmReceivePOLine()

Cette fonction effectue la réception d'une certaine quantité d'éléments sur une ligne de commande et renvoie le numéro d'identifiant de la ligne de réception.



Note : Attention : les lignes de réception sont créées par un agent au moment du "commit" de la transaction. Vous ne pouvez pas y accéder avant.

Syntaxe API

```
long AmReceivePOLine(long hApiCnxBase, long
lPordLineId, long lDelivId, float fQty);
```

Syntaxe BASIC interne

```
Function AmReceivePOLine(lPordLineId As Long,
lDelivId As Long, fQty As Single) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	
Script FINISH.DO d'un assistant	✓

Entrée

- *lPordLineId* : Ce paramètre contient l'identifiant de la ligne de commande contenant les éléments à réceptionner.
- *lDelivId* : Ce paramètre contient l'identifiant de la fiche de réception qui recevra une certaine quantité des éléments présents sur la ligne de commande.
- *fQty* : Ce paramètre contient la quantité d'éléments sur la ligne de commande à réceptionner dans la fiche de réception.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmRefreshAllCaches()

Cette fonction rafraîchit l'ensemble des caches utilisés sous AssetCenter.

Syntaxe API





```
long AmRefreshAllCaches(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmRefreshAllCaches() As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmRefreshLabel()

L'API AmRefreshLabel rafraîchit la chaîne de l'étiquette d'un enregistrement donné (lMainId) dans une table donnée (strTableName).

Syntaxe API




```
long AmRefreshLabel(long hApiCnxBase, long lMainId,
char *strTableName, char *pstrLabel, long lLabel);
```

Syntaxe BASIC interne

```
Function AmRefreshLabel(lMainId As Long,
strTableName As String) As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lMainId* : ce paramètre contient l'identifiant qui sera rafraîchi.
- *strTableName* : ce paramètre précise le nom de la table associé à lMainId.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmRefreshProperty()

Réévalue la valeur d'une propriété identifiée par le paramètre *strVarName*. Si cette propriété utilise un script, celui-ci est à nouveau exécuté.

L'arbre de dépendance est remis à jour, le cas échéant.

Syntaxe BASIC interne

```
Function AmRefreshProperty(strVarName As String)
As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strVarName* : Nom de la propriété (de l'assistant) que vous souhaitez réévaluer.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmRefreshTraceHist()

L'API AmRefreshTraceHist rafraîchit un historique complet de chaîne de liaisons de projet. Elle possède aussi un paramètre optionnel qui rafraîchit les entrées "individuelles" d'historique de chaîne de liaisons. Si ce paramètre n'est pas présent, l'historique complet de chaîne de liaisons sera rafraîchi.

Syntaxe API




```
long AmRefreshTraceHist(long hApiCnxBase, long
lCabTraceOutId, long lTraceHistId);
```

Syntaxe BASIC interne

```
Function AmRefreshTraceHist(lCabTraceOutId As Long,
lTraceHistId As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCabTraceOutId* : ce paramètre contient l'identifiant de compte-rendu de chaîne de liaisons de câble.
- *lTraceHistId* : ce paramètre optionnel rafraîchit les entrées "individuelles" d'historique de chaîne de liaisons.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmReleaseHandle()

Cette fonction libère le handle et tous les sous-handles d'un objet.

Syntaxe API






```
long AmReleaseHandle(long hApiObject);
```

Syntaxe BASIC interne

```
Function AmReleaseHandle(hApiObject As Long) As  
Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiObject* : Ce paramètre contient un handle sur l'objet concerné.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmRemoveCable()

L'API AmRemoveCable enlève un câble (ICableId) de sa localisation actuelle. L'état du câble est changé en "Indisponible". Si le projet (IProjectId) et l'intervention (IWorkOrderId) prennent des valeurs, le câble est ajouté au projet et à l'intervention avec le commentaire donné

(*strComment*). Ce commentaire décrit l'action qui sera accomplie sur le câble (i.e. "Enlever un câble de sa localisation actuelle").

Syntaxe API




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

Syntaxe BASIC interne

```
Function AmRemoveCable(lCableId As Long, lProjectId
As Long, lWorkOrderId As Long, strComment As
String) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCableId* : ce paramètre contient l'identifiant du câble à enlever.
- *lProjectId* : ce paramètre contient l'identifiant du projet.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmRemoveDevice()

L'API AmRemoveDevice enlève un dispositif (lDeviceId) de sa localisation actuelle. L'état du câble est changé en "Indisponible". Si le projet (lProjectId) et l'intervention (lWorkOrderId) prennent des valeurs, le câble est ajouté au projet et à l'intervention avec le commentaire donné (strComment). Ce commentaire décrit l'action qui sera accomplie sur le dispositif (i.e. "Enlever un dispositif de sa localisation actuelle").

Syntaxe API




```
long AmRemoveDevice(long hApiCnxBase, long
lDeviceId, long lProjectId, long lWorkOrderId, char
*strComment);
```

Syntaxe BASIC interne

```
Function AmRemoveDevice(lDeviceId As Long,
lProjectId As Long, lWorkOrderId As Long,
strComment As String) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lDeviceId* : ce paramètre contient l'identifiant du dispositif à enlever.
- *lProjectId* : ce paramètre contient l'identifiant du projet.
- *lWorkOrderId* : ce paramètre contient l'identifiant de l'intervention.
- *strComment* : ce paramètre est le commentaire qui sera joint à l'intervention.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmReturnAsset()

Cette fonction permet de retourner un bien.

Syntaxe API

```
long AmReturnAsset(long hApiCnxBase, long lAstId,  
long lReturnId, long bCanMerge);
```

Syntaxe BASIC interne

```
Function AmReturnAsset(lAstId As Long, lReturnId  
As Long, bCanMerge As Long) As Long
```

Champ d'application



Version : 4.00

AssetCenter APIs

Utilisable



Utilisable

Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lAstId* : Ce paramètre contient l'identifiant du bien à retourner.
- *lReturnId* : Ce paramètre contient l'identifiant de la fiche de retour.
- *bCanMerge* : Ce paramètre permet de préciser si le retour peut être fusionné avec une ligne déjà existante dans la fiche de retour.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmReturnContract()

Cette fonction permet de retourner un contrat.

Syntaxe API




```
long AmReturnContract(long hApiCnxBase, long lCntrId, long lReturnId, long bCanMerge);
```

Syntaxe BASIC interne

```
Function AmReturnContract(lCntrId As Long,
    lReturnId As Long, bCanMerge As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lCntrId* : Ce paramètre contient l'identifiant du contrat à retourner.
- *lReturnId* : Ce paramètre contient l'identifiant de la fiche de retour.
- *bCanMerge* : Ce paramètre permet de préciser si le retour peut être fusionné avec une ligne déjà existante dans la fiche de retour.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmReturnPortfolioItem()

Cette fonction permet de retourner un élément de parc.

Syntaxe API




```
long AmReturnPortfolioItem(long hApiCnxBase, long
lPfId, float fQty, long lFromRecptLineId, long
lReturnId, long bCanMerge);
```

Syntaxe BASIC interne

```
Function AmReturnPortfolioItem(lPfId As Long, fQty
As Single, lFromRecptLineId As Long, lReturnId As
Long, bCanMerge As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lPfId* : Ce paramètre contient l'identifiant de l'élément de parc à retourner.
- *fQty* : Ce paramètre contient la quantité (dans l'unité du modèle) à retourner.
- *lFromRecptLineId* : Ce paramètre contient l'identifiant de la ligne de réception source.

- *lReturnId* : Ce paramètre contient l'identifiant de la fiche de retour.
- *bCanMerge* : Ce paramètre permet de préciser si le retour peut être fusionné avec une ligne déjà existante dans la fiche de retour.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmReturnTraining()

Cette fonction permet de retourner une formation.

Syntaxe API

```
long AmReturnTraining(long hApiCnxBase, long
lTrainingId, long lReturnId, long bCanMerge);
```

Syntaxe BASIC interne



```
Function AmReturnTraining(lTrainingId As Long,
lReturnId As Long, bCanMerge As Long) As Long
```

Champ d'application

Version : 4.00

Utilisable
AssetCenter APIs 

Utilisable

Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lTrainingId* : Ce paramètre contient l'identifiant de la formation à retourner.
- *lReturnId* : Ce paramètre contient l'identifiant de la fiche de retour.
- *bCanMerge* : Ce paramètre permet de préciser si le retour peut être fusionné avec une ligne déjà existante dans la fiche de retour.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmReturnWorkOrder()

Cette fonction permet de retourner une intervention.

Syntaxe API




```
long AmReturnWorkOrder(long hApiCnxBase, long lWOId, long lReturnId, long bCanMerge);
```

Syntaxe BASIC interne

```
Function AmReturnWorkOrder(lWOId As Long, lReturnId
As Long, bCanMerge As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lWOId* : Ce paramètre contient l'identifiant de l'intervention à retourner.
- *lReturnId* : Ce paramètre contient l'identifiant de la fiche de retour.
- *bCanMerge* : Ce paramètre permet de préciser si le retour peut être fusionné avec une ligne déjà existante dans la fiche de retour.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmRevCryptPassword()

Cette fonction décrypte un mot de passe crypté.

Syntaxe API






```
long AmRevCryptPassword(long hApiCnxBase, char
*return, long lreturn, char *strPassword);
```

Syntaxe BASIC interne

```
Function AmRevCryptPassword(strPassword As String)
As String
```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strPassword* : Ce paramètre contient le mot de passe à décrypter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

AmRgbColor()

Cette fonction donne la valeur RGB de la couleur correspondant au paramètre *strText*.

Syntaxe API






```
long AmRgbColor(char *strText);
```

Syntaxe BASIC interne

```
Function AmRgbColor(strText As String) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strText*: Ce paramètre contient le nom d'une couleur :
 - White
 - ltGray
 - Gray
 - Dkgray

- Black
- Red
- Green
- Blue
- Yellow
- Cyan
- Magenta
- Dkyellow
- Dkgreen
- Dkcyan
- Dkblue
- Dkmagenta
- Dkred

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmRollback()

Cette fonction annule toutes les modifications effectuées avant la déclaration de début de transaction (effectuée via la fonction `AmStartTransaction`).

Syntaxe API




```
long AmRollback(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmRollback() As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmSetFieldValue()

Cette fonction modifie un champ d'un enregistrement. Aucune mise à jour de la base de données n'est effectuée par cette fonction. La modification sera effectuée lors de la mise à jour ou l'insertion de l'enregistrement, ou encore lors du commit de la transaction.

Syntaxe API




```
long AmSetFieldValue(long hApiRecord, char *strFieldName, long tmValue);
```

Syntaxe BASIC interne

```
Function AmSetFieldValue(hApiRecord As Long, strFieldName As String, tmValue As Date) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient le handle sur l'enregistrement contenant le champ à modifier.
- *strFieldName* : Ce paramètre contient le nom SQL du champ à modifier.
- *tmValue* : Ce paramètre contient la nouvelle valeur du champ au format "Date".

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmSetFieldDoubleValue()

Cette fonction modifie en mémoire un champ d'un enregistrement. Aucune mise à jour de la base de données n'est effectuée par cette fonction.

Syntaxe API




```
long AmSetFieldDoubleValue(long hApiRecord, char *strFieldName, double dValue);
```

Syntaxe BASIC interne

```
Function AmSetFieldDoubleValue(hApiRecord As Long,
    strFieldName As String, dValue As Double) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient le handle sur l'enregistrement contenant le champ à modifier.
- *strFieldName* : Ce paramètre contient le nom SQL du champ à modifier.
- *dValue* : Ce paramètre contient la nouvelle valeur du champ au format "Double".

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmSetFieldLongValue()

Cette fonction modifie un champ d'un enregistrement. Aucune mise à jour de la base de données n'est effectuée par cette fonction. Pour modifier la valeur d'un champ date, heure ou date+heure, vous devez

donner comme nouvelle valeur le nombre de secondes écoulées depuis le 01/01/1970 à 00:00:00.

Syntaxe API




```
long AmSetFieldLongValue(long hApiRecord, char
*strFieldName, long lValue);
```

Syntaxe BASIC interne

```
Function AmSetFieldLongValue(hApiRecord As Long,
strFieldName As String, lValue As Long) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient le handle sur l'enregistrement contenant le champ à modifier.
- *strFieldName* : Ce paramètre contient le nom SQL du champ à modifier. Vous pouvez également donner le nom SQL d'une caractéristique, d'un champ de type "Commentaire" ou encore d'un champ de script.
- *lValue* : Ce paramètre contient la nouvelle valeur du champ.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmSetFieldStrValue()

Cette fonction modifie un champ d'un enregistrement. Aucune mise à jour de la base de données n'est effectuée par cette fonction.

Syntaxe API




```
long AmSetFieldStrValue(long hApiRecord, char
*strFieldName, char *strValue);
```

Syntaxe BASIC interne

```
Function AmSetFieldStrValue(hApiRecord As Long,
strFieldName As String, strValue As String) As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *hApiRecord* : Ce paramètre contient le handle sur l'enregistrement contenant le champ à modifier.

- *strFieldName* : Ce paramètre contient le nom SQL du champ à modifier.
- *strValue* : Ce paramètre contient la nouvelle valeur du champ au format "String" (chaîne de caractères).

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmSetLinkFeatureValue()

Cette fonction fixe la valeur d'une caractéristique de type lien pour un enregistrement donné.

Syntaxe API



```
long AmSetLinkFeatureValue(long hApiRecord, char
*strFeatSqlName, char *strDstSelfValue, long
lDstId);
```

Syntaxe BASIC interne

```
Function AmSetLinkFeatureValue(hApiRecord As Long,
strFeatSqlName As String, strDstSelfValue As
String, lDstId As Long) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

Utilisable

Script d'un assistant

Script FINISH.DO d'un assistant



Entrée

- *hApiRecord* : Ce paramètre contient l'identifiant de l'enregistrement auquel est associée la caractéristique de type lien.
- *strFeatSqlName* : Ce paramètre contient le nom SQL de la caractéristique de type lien dont on souhaite fixer la valeur. Ce nom SQL est toujours préfixé par "fv_".
- *strDstSelfValue* : Ce paramètre contient la valeur de la caractéristique telle qu'elle sera affichée pour l'enregistrement. Il s'agit de la valeur "Self" de l'enregistrement d'identifiant *lDstId*. Si vous renseignez ce paramètre avec une valeur non valide ou non existante, l'intégrité de la base de données risque d'être corrompue.
- *lDstId* : Ce paramètre contient l'identifiant de l'enregistrement sur lequel pointe la caractéristique de type lien.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Am SetProperty()



Cette fonction fixe la valeur d'une propriété identifiée par son nom. Elle met également à jour l'arbre de dépendances de cette propriété.

Syntaxe BASIC interne

```
Function AmSetProperty(strVarName As String, vValue
As Variant) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strVarName* : Ce paramètre contient le nom de la propriété dont on souhaite fixer la valeur.
- *vValue* : Ce paramètre contient la nouvelle valeur pour la propriété.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmShowCableCrossConnect()




Syntaxe BASIC interne

```
Function AmShowCableCrossConnect(lCableId As Long)
As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.




AmShowDeviceCrossConnect()

Syntaxe BASIC interne

```
Function AmShowDeviceCrossConnect(lDeviceId As Long) As Long
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmSqlTextConst()

Syntaxe API






```
long AmSqlTextConst(char *return, long lreturn,
char *str);
```

Syntaxe BASIC interne

```
Function AmSqlTextConst(str As String) As String
```

Champ d'application

Version : 4.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

AmStartTransaction()

Cette fonction démarre une nouvelle transaction avec la base de données associée à la connexion. La prochaine commande de "Commit" ou de "Rollback" validera ou annulera toutes les modifications apportées à la base de données.

Syntaxe API

```
long AmStartTransaction(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmStartTransaction() As Long
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	✓
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	✓
Script d'un assistant	
Script FINISH.DO d'un assistant	✓

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmStartup()


Cette fonction doit être appelée avant tout autre fonction. Elle initialise l'appel aux bibliothèques AssetCenter.

Syntaxe API

```
void AmStartup();
```

Champ d'application

Version : 2.52

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

AmTableDesc()

Cette fonction génère une chaîne de format "<Description de la table> (<Nom SQL de la table>)" à partir du nom SQL de la table.

Syntaxe API

```
long AmTableDesc(long hApiCnxBase, char *return,
long lreturn, char *strSqlName);
```

Syntaxe BASIC interne

```
Function AmTableDesc(strSqlName As String) As
String
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strSqlName* : Nom SQL de la table pour laquelle on veut générer une chaîne de description. Si ce paramètre contient un nom SQL non valide, la fonction renvoie un point d'interrogation ("?").

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant génère une chaîne de description pour la table des biens (Nom SQL : `amAsset`) :

```
AmTableDesc ( "amAsset " )
```

Le résultat est le suivant :

```
Biens (amAsset)
```


AmTaxRate()

Cette fonction calcule un taux de taxe en fonction d'un type de taxe, d'une juridiction fiscale et d'une date.

Syntaxe API






```
double AmTaxRate(char *strTaxRateName, long
lTaxLocId, long tmDate, double dValue);
```

Syntaxe BASIC interne

```
Function AmTaxRate(strTaxRateName As String,
lTaxLocId As Long, tmDate As Date, dValue As
Double) As Double
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strTaxRateName* : Ce paramètre contient le nom SQL du type de taxe utilisé pour calculer le taux de taxe.
- *lTaxLocId* : Ce paramètre contient le numéro d'identifiant de la juridiction fiscale concernée par le type de taxe.
- *tmDate* : Ce paramètre contient la date à laquelle vous souhaitez évaluer le taux de taxe.

- *dValue* : ?

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

AmUpdateDetail()

Cette fonction est utilisée dans les assistants d'aide à la saisie. La définition du contexte (la table pour laquelle un enregistrement est mis à jour ou renseigné à l'aide de l'assistant) n'est donc pas nécessaire. La fonction met à jour un ou renseigne des champs ou des liens du contexte en fonction d'une valeur. Cette fonction est interdite dans les assistants non modaux.

Syntaxe BASIC interne

```
Function AmUpdateDetail(strFieldName As String,
varValue As Variant) As Long
```

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs

Script de configuration d'un champ ou d'un lien

Action de type "Script"

Script d'un assistant

Utilisable**Script FINISH.DO d'un assistant****Entrée**

- *strFieldName* : Ce paramètre contient le nom SQL du champ à mettre à jour.
- *varValue* : Ce paramètre contient la nouvelle valeur du champ.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmUpdateLoginSlot()

Cette fonction force la mise à jour des informations relatives au jeton de connexion de l'utilisateur connecté.

Syntaxe API

```
long AmUpdateLoginSlot(long hApiCnxBase);
```

Syntaxe BASIC interne

```
Function AmUpdateLoginSlot() As Long
```

Champ d'application**Version : 3.00****Utilisable****AssetCenter APIs****Script de configuration d'un champ ou d'un lien****Action de type "Script"**

Utilisable

Script d'un assistant

Script FINISH.DO d'un assistant

**Sortie**

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmUpdateRecord()

Cette fonction permet de mettre à jour un enregistrement.

Syntaxe API

```
long AmUpdateRecord(long hApiRecord);
```

Syntaxe BASIC interne

```
Function AmUpdateRecord(hApiRecord As Long) As Long
```

Champ d'application

Version : 2.52

Utilisable

AssetCenter APIs

Script de configuration d'un champ ou d'un lien

Action de type "Script"

Script d'un assistant

Script FINISH.DO d'un assistant

**Entrée**

- *hApiRecord* : Ce paramètre contient un handle sur l'enregistrement que vous souhaitez mettre à jour.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmValueOf()

Utilisée au sein d'un assistant, cette fonction renvoie la valeur de la propriété identifiée par le paramètre *strVarName*.

Syntaxe BASIC interne

```
Function AmValueOf(strVarName As String) As Variant
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strVarName* : Ce paramètre contient le nom de la propriété dont on veut connaître la valeur.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant renvoie la valeur de la propriété "Page1.Label" :

```
AmValueOf ( "Page1.Label" )
```

Utilisez cette fonction avec prudence car elle brise la chaîne de dépendances de la propriété qu'elle traite.

AmWizChain()

Cette fonction exécute un assistant B au sein d'un assistant A. En fin d'exécution de l'assistant B, la fonction rend la main à l'assistant A.

Syntaxe BASIC interne

```
Function AmWizChain(strWizSqlName As String) As Long
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strWizSqlName* : Nom SQL de l'assistant à exécuter.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

AmWorkTimeSpanBetween()

Cette fonction renvoie la durée ouvrée entre deux dates. Cette durée est exprimée en secondes et respecte les informations d'un calendrier des périodes ouvrées.

Syntaxe API






```
long AmWorkTimeSpanBetween(char
*strCalendarSqlName, long tmEnd, long tmStart);
```

Syntaxe BASIC interne

```
Function AmWorkTimeSpanBetween(strCalendarSqlName
As String, tmEnd As Date, tmStart As Date) As Date
```

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strCalendarSqlName* : Ce paramètre contient le nom SQL du calendrier des périodes ouvrées utilisé comme base pour le calcul de la durée ouvrée écoulée entre les deux dates. Si ce paramètre est omis, la durée calculée ne tient compte d'aucune période ouvrée.
- *tmEnd* : Ce paramètre contient la date de fin de la période sur laquelle on effectue le calcul de la durée ouvrée.
- *tmStart* : Ce paramètre contient la date de début de la période sur laquelle on effectue le calcul de la durée ouvrée.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant calcule la durée ouvrée entre le 01/09/1998 à 8h00 et le 24/09/1998 à 19h00. Le calendrier utilisé, de nom SQL "Calendar_Paris" définit les périodes ouvrées suivantes :

- Du lundi au jeudi de 8h00 à 12h00, puis de 14h00 à 18h00.
- Le vendredi de 8h00 à 12h00, puis de 14h00 à 17h00.

```
AmWorkTimeSpanBetween("Calendar_Paris", "1998/09/24  
19:00:00", "1998/09/01 08:00:00")
```

Cet exemple renvoie la valeur 507600 qui représente le nombre de secondes ouvrées écoulées entre les deux dates.

AppendOperand()

Concatène une chaîne en fonction des paramètres passés à la fonction.
Le résultat a la forme suivante :

```
strExpr
strOperator
strOperand
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strExpr* : Expression à concaténer.
- *strOperator* : Opérateur à concaténer.
- *strOperand* : Opérande à concaténer.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Si l'un des paramètres strExpr ou strOperand est omis, strOperator n'est pas utilisé dans la concaténation.

ApplyNewVals()

Affecte des valeurs identiques pour les cellules identifiées d'un contrôle "ListBox".

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strValues* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strNewVals* : Nouvelle valeur à affecter aux cellules concernées.
- *strRows* : Identifiants des lignes à traiter. Les identifiants sont séparés par une virgule.
- *strRowFormat* : Instructions de formatage de la sous-liste. Les instructions sont séparées par le caractère "|". Chaque instruction représente le numéro de la colonne qui contiendra *strNewVals*.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Asc()

Renvoie le code ASCII du premier caractère d'une chaîne.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strAsc* : Chaîne de caractères sur laquelle opère la fonction.

Exemple

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

Atn()

Renvoie l'arc tangente d'un nombre, exprimé en radians

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Nombre dont vous souhaitez connaître l'arc tangente.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

BasicToLocalDate()





Cette fonction convertit une date au format Basic en une date au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows).

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strDateBasic* : Date au format Basic à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

BasicToLocalTime()

Cette fonction convertit une heure au format Basic en une heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows).

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strTimeBasic* : Heure au format Basic à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

BasicToLocalTimeStamp()

Cette fonction convertit un ensemble Date+Heure au format Basic en un ensemble Date+Heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows).

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strTSBasic* : Date+Heure au format Basic à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Beep()

Emet un son (un beep) sur la machine.





Syntaxe BASIC interne

```
Function Beep()
```

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Cdbl()

Convertit une expression en un double ("Double").





Syntaxe BASIC interne

```
Function Cdbl(dValue As Double) As Double
```

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Expression à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

ChDir()

Change le répertoire courant.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strDirectory* : Nouveau répertoire courant.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

ChDrive()

Change le lecteur courant.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable
AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strDrive* : Nouveau lecteur.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Chr()

Renvoie la chaîne correspondant au code ASCII passé par le paramètre *iChr*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable
AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *iChr* : Code ASCII du caractère.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
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()





Convertit une expression en un entier ("Integer").

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *iValue* : Expression à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

CLng()





Convertit une expression en un long ("Long").

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lValue* : Expression à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

Cos()





Renvoie le cosinus d'un nombre, exprimé en radians.

Syntaxe BASIC interne

Function Cos(dValue As Double) As Double

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le cosinus.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

CountOccurrences()

Compte le nombre d'occurrences d'une chaîne de caractères à l'intérieur d'une autre chaîne.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

Utilisable

AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strSearched* : Chaîne de caractères à l'intérieur de laquelle s'effectue la recherche.
- *strPattern* : Chaîne de caractères à rechercher à l'intérieur de *strSearched*.
- *strEscChar* : Caractère d'échappement. Si la fonction rencontre ce caractère à l'intérieur de la chaîne *strSearched*, la recherche s'arrête.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
  MyStr=CountOccurences("toi|moi|toi,moi|toi",
"toi", ",") :'Renvoie la valeur "2"
  MyStr=CountOccurences("toi|moi|toi,moi|toi",
"toi", "|") :'Renvoie la valeur "1"
```

CountValues()

Compte le nombre d'éléments dans une chaîne de caractères en tenant compte d'un séparateur et d'un caractère d'échappement.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strSearched* : Chaîne de caractères à traiter.
- *strSeparator* : Séparateur utilisé pour délimiter les éléments.
- *strEscChar* : Caractère d'échappement. Si ce caractère préfixe un séparateur, ce dernier sera ignoré.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
MyStr=CountValues("toi|moi|toi\|moi|toi", "|",
"\") : 'Renvoie la valeur 4
MyStr=CountValues("toi|moi|toi\|moi|toi", "|",
"") : 'Renvoie la valeur 5
```

CSng()





Convertit une expression en un nombre à virgule flottante ("Float").

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *fValue* : Expression à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

CStr()

Convertit une expression en une chaîne de caractères ("String").

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓



Entrée

- *strValue* : Expression à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

CurDir()

Renvoie le chemin courant.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

CVar()

Convertit une expression en un variant ("Variant").



Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *vValue* : Expression à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Date()



Renvoie la date courante du système.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

DateSerial()





Cette fonction renvoie une date formatée en fonction des paramètres *iYear*, *iMonth* et *iDay*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *iYear* : Année. Si sa valeur est comprise entre 0 et 99, ce paramètre décrit les années de 1900 à 1999. Pour toutes les autres années, vous devez utiliser un nombre de quatre chiffres (par exemple 1800).
- *iMonth* : Mois.
- *iDay* : Jour.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

Chacun de ces paramètres peut prendre pour valeur une expression numérique représentant un nombre de jours, de mois ou d'années. Ainsi l'exemple suivant :

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

renvoie la valeur :

```
1989/1/7
```

Lorsque la valeur d'un paramètre est en dehors de l'intervalle de valeurs généralement admis (c'est à dire 1-31 pour les jours, 1-12 pour les mois, ...), elle est convertie vers le paramètre immédiatement supérieur. Ainsi, si vous entrez "35" comme valeur pour le paramètre `iDay`, ce dernier sera interprété comme 1 mois et un jour.

L'exemple suivant :

```
DateSerial (1999-50, 9-5, 1-2)
```

renvoie la valeur :

1949/3/30

DateValue()

Cette fonction renvoie la partie date d'une valeur "Date+Heure"

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmDate* : Date au format "Date+Heure".

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant :

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

renvoie la valeur :

```
1999/09/24
```

Day()

Renvoie le jour contenu dans le paramètre *tmDate*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

EscapeSeparators()



Préfixe un ou plusieurs caractère(s) défini(s) comme séparateur(s) par un caractère d'échappement.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

	Utilisable
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strSource* : Chaîne de caractères à traiter.
- *strSeparators* : Liste des séparateurs à préfixer. Si vous souhaitez déclarer plusieurs séparateurs, vous devez les séparer par le caractère utilisé comme caractère d'échappement (indiqué dans le paramètre *strEscChar*).
- *strEscChar* : Caractère d'échappement. Il préfixera tous les séparateurs définis dans *strSeparators*.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
MyStr=EscapeSeparators("toi|moi|toi,moi|toi",
"|\",", "\") : 'Renvoie la valeur
"toi\|moi\|toi\,moi\|toi"
```

ExeDir()





Cette fonction renvoie le chemin complet de l'exécutable.

Syntaxe BASIC interne

Function ExeDir() As String

Champ d'application

Version : 3.60

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

Exp()





Renvoie l'exponentielle d'un nombre.

Syntaxe BASIC interne

Function Exp(*dValue* As Double) As Double

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Nombre dont vous souhaitez connaître l'exponentielle.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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


ExtractValue()

Extrait d'une chaîne de caractères les valeurs délimitées par un séparateur. La valeur récupérée est alors effacée de la chaîne source. Cette opération tient compte d'un éventuel caractère d'échappement. Si le séparateur n'est pas trouvé dans la chaîne source, l'intégralité de la chaîne est renvoyée et la chaîne source est entièrement effacée.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *pstrData* : Chaîne source à traiter.
- *strSeparator* : Caractère utilisé comme séparateur dans la chaîne source.
- *strEscChar* : Caractère d'échappement. Si ce caractère préfixe le séparateur, ce dernier est ignoré.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
  MyStr=ExtractValue("toi,moi", ",", "\") : 'Renvoie
"toi" et laisse "moi" dans la chaîne source
  MyStr=ExtractValue(",toi,moi", ",", "\")
:'Renvoie "" et laisse "toi,moi" dans la chaîne
source
  MyStr=ExtractValue("toi", ",", "\") : 'Renvoie
"toi" et laisse "" dans la chaîne source
  MyStr=ExtractValue("toi\,moi", ",", "\")
:'Renvoie "toi\,moi" et laisse "" dans la chaîne
source
  MyStr=ExtractValue("toi\,moi", ",", "") : 'Renvoie
"toi\" et laisse "moi" dans la chaîne source
  RetVal=""
```

FileCopy()

Copie un fichier ou un répertoire.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strSource* : Chemin complet du fichier ou du répertoire à copier.
- *strDest* : chemin complet du fichier ou du répertoire de destination.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

FileDateTime()

Renvoie la date et l'heure d'un fichier sous la forme d'un "Long".

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strFileName* : Chemin complet du fichier concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

FileLen()

Revoie la taille d'un fichier.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strFileName* :Chemin complet du fichier concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Fix()

Renvoie la partie entière (premier entier supérieur dans le cas d'un nombre négatif) d'un nombre à virgule.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la partie entière.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

FormatDate()

Formate une date en fonction de l'expression contenue dans le paramètre *strFormat*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmFormat* : Date à formater.
- *strFormat* : Expression contenant les instructions de formatage.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple de code suivant montre comment formater une date :

```
Dim MyDate
MyDate="2000/03/14"
```

```
RetVal=FormatDate(MyDate, "dddd d mmmm yyyy")
:'Renvoie "Tuesday 14 March 2000"
```

FormatResString()





Cette fonction traite une chaîne source en remplaçant les variables \$1, \$2, \$3, \$4 et \$5 respectivement par les chaînes contenues dans les paramètres *strParamOne*, *strParamTwo*, *strParamThree*, *strParamFour* et *strParamFive*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strResString* : Chaîne source à traiter.
- *strParamOne* : Chaîne de remplacement de la variable \$1.
- *strParamTwo* : Chaîne de remplacement de la variable \$2.
- *strParamThree* : Chaîne de remplacement de la variable \$3.

- *strParamFour* : Chaîne de remplacement de la variable \$4.
- *strParamFive* : Chaîne de remplacement de la variable \$5.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant :

```
FormatResString( "je$1il$2vous$3", "tu", "nous",
"ils" )
```




renvoie "jetuilnousvousils".

FormatString()

Formate une chaîne de caractères en fonction de l'expression contenue dans le paramètre *strFormat*.

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Exemple**

L'exemple de code suivant montre comment formater une chaîne de caractères :

```
Dim MyString
  MyString="2000/03/14"
  RetVal=FormatString(MyString, "dddd d mmmm yyyy")
: 'Renvoie "Tuesday 14 March 2000"
```

FV()

Cette fonction renvoie le futur montant d'une annuité basée sur des versements constants et périodiques, et sur un taux d'intérêt fixe.

Syntaxe BASIC interne

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

Champ d'application**Version : 3.00****Utilisable****AssetCenter APIs****Script de configuration d'un champ ou d'un lien****Action de type "Script"****Script d'un assistant****Script FINISH.DO d'un assistant**

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

$0,06/12=0,005$ soit 0,5%

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
 - 0 si les paiements sont dus à terme échu (c'est à dire en fin de période)
 - 1 si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités. Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

GetListItem()

Renvoie la *lN*bième portion d'une chaîne délimitée par des séparateurs.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.

- *INb* : Position de la chaîne à récupérer.
- *strEscChar* : Caractère d'échappement. Si ce caractère préfixe un séparateur, ce dernier sera ignoré.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant :

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

renvoie "est".

```
GetListItem("ceci%_est_un_test", "_", 2, "%")
```

renvoie "un".

Hex()

Revoie la valeur hexadécimale d'un nombre.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la valeur hexadécimale.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Hour()

Renvoie la valeur de l'heure contenue dans le paramètre *tmTime*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmTime* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

InStr()

Renvoie la position de la première occurrence d'une chaîne de caractères à l'intérieur d'une autre chaîne de caractères.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *iPosition* : Position de départ de la recherche. Ce paramètre ne peut être négatif et ne doit pas dépasser 65.535.
- *strSource* : Chaîne dans laquelle s'effectue la recherche.
- *strPattern* : Chaîne de caractères à rechercher.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

Int()

Renvoie la partie entière (premier nombre entier inférieur dans le cas d'un nombre négatif) d'un nombre à virgule.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la partie entière.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

IPMT()



Cette fonction renvoie le montant des intérêts pour une échéance donnée d'une annuité.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

$$0,06/12=0,005 \text{ soit } 0,5\%$$

- *iPer* : Ce paramètre indique la période concernée par le calcul, comprise entre 1 et la valeur du paramètre *Nper*.
- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
 - 0 si les paiements sont dus à terme échu (c'est à dire en fin de période)
 - 1 si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités. Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

IsNumeric()

Cette fonction permet de déterminer si une chaîne de caractères contient une valeur numérique.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strString* : Ce paramètre contient la chaîne de caractères à analyser.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Kill()

Efface un fichier.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strKilledFile* : Chemin complet du fichier concerné par l'opération.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

LCase()





Passes tous les caractères d'une chaîne en minuscules.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strString* : Chaîne de caractères à passer en minuscules.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
' 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()

Renvoie les *iNumber* premiers caractères d'une chaîne en partant de la gauche.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strString* : Chaîne de caractères à traiter.
- *iNumber* : Nombre de caractères à renvoyer.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
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()

Extrait la portion d'une chaîne de caractères située à gauche du séparateur précisé dans le paramètre *strSep*.

La recherche du séparateur s'effectue de la gauche vers la droite.





La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test" :

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

Renvoie la chaîne "Ceci".

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

Renvoie la chaîne "Ceci_est_un".

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

Renvoie la chaîne "test".

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

Renvoie la chaîne "est_un_test".

LeftPartFromRight()

Extrait la portion d'une chaîne de caractères située à gauche du séparateur précisé dans le paramètre *strSep*.

La recherche du séparateur s'effectue de la droite vers la gauche.

La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test":

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

Revoie la chaîne "Ceci".

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

Revoie la chaîne "Ceci_est_un".

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

Revoie la chaîne "test".

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

Revoie la chaîne "est_un_test".

Len()

Revoie le nombre de caractères d'une chaîne ou d'un variant.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *vValue* : Variant concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim strTest as String
Dim iLength as Integer
  strTest = "Peregrine Systems"
  iLength = Len(strTest) : 'The value of iLength
is 17
  RetVal=iLength
```

LocalToBasicDate()

Cette fonction convertit une date au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows) en une date au format Basic.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strDateLocal* : Date au format chaîne à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

LocalToBasicTime()

Cette fonction convertit une heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows) en une heure au format Basic.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strTimeLocal* : Heure au format chaîne à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

LocalToBasicTimeStamp()

Cette fonction convertit un ensemble Date+Heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows) en un ensemble Date+Heure au format Basic.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strTSLocal* : Date+Heure au format chaîne à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

LocalToUTCDate()

Cette fonction convertit une date au format "Date+Heure" en une date au format UTC (indépendante d'un quelconque fuseau horaire).

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmLocal* : Date au format "Date+Heure".

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Log()





Renvoie le logarithme népérien d'un nombre.

Syntaxe BASIC interne

Function Log(dValue As Double) As Double

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le logarithme.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

LTrim()

Supprime tous les espaces précédant le premier caractère (différent d'un espace) d'une chaîne.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strString* : Chaîne de caractères à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
' 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()

Cette fonction renvoie un booléen inversé (0 devient 1, tout autre nombre devient 0).

Syntaxe BASIC interne

Function MakeInvertBool(*lValue* As Long) As Long

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *lValue* : Nombre concerné par l'opération.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyValue
MyValue=MakeInvertBool(0) : 'Renvoie la valeur 1
MyValue=MakeInvertBool(1) : 'Renvoie la valeur 0
MyValue=MakeInvertBool(254) : 'Renvoie la valeur
0
```

Mid()





Extrait une chaîne de caractères contenue dans une autre chaîne.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strString* : Chaîne de caractères concernée par l'opération.
- *iStart* : Position de départ de la chaîne à extraire à l'intérieur de *strString*.
- *iLen* : longueur de la chaîne à extraire.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim strTest as String
  strTest="One Two Three" : ' Defines the test
string
  strTest=Mid(strTest,5,3) : ' strTest="Two"
  RetVal=strTest
```

Minute()

Renvoie le nombre de minutes contenues l'heure exprimée par le paramètre *tmTime*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmTime* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim strMinute
    strMinute=Minute(Date())
    RetVal=strMinute : 'Renvoie le nombre de minutes
écoulées dans l'heure courante par exemple "45"
s'il est actuellement 15:45:30
```

MkDir()




Crée un répertoire.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	

Utilisable**Script FINISH.DO d'un assistant****Entrée**

- *strMkDirectory* : Chemin complet du répertoire à créer.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Month()

Renvoie le mois contenu dans la date exprimée par le paramètre *tmDate*.

Syntaxe BASIC interne

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

Champ d'application**Version : 3.00****Utilisable**

AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim strMonth
  strMonth=Month(Date())
RetVal=strMonth : 'Renvoie le mois courant
```

Name()





Renomme un fichier.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strSource* : Chemin complet du fichier à renommer.
- *strDest* : Nouveau nom du fichier.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Now()





Renvoie la date et l'heure courantes.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

NPER()





Cette fonction renvoie le nombre d'échéances d'une annuité basée sur des versements constants et périodiques, et sur un taux d'intérêt fixe.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

$0,06/12=0,005$ soit 0,5%

- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
 - 0 si les paiements sont dus à terme échu (c'est à dire en fin de période)
 - 1 si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les sommes versées (exprimées notamment par le paramètre *Pmt*) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

Oct()

Revoie la valeur octale d'un nombre.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la valeur octale.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

ParseDate()





Cette fonction convertit une date exprimée sous la forme d'une chaîne de caractères en un objet date au sens Basic du term.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.60

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strDate* : Date au format chaîne de caractères.
- *strFormat* : Ce paramètre contient le format de la date contenue dans la chaîne de caractères. Les valeurs possibles sont les suivantes :
 - DD/MM/YY
 - DD/MM/YYYY
 - MM/DD/YY
 - MM/DD/YYYY
 - YYYY/MM/DD
 - Date : date exprimée suivant les paramètres de date du poste client.
 - DateInter : date exprimée au format international
- *strStep* : Ce paramètre optionnel contient le séparateur de date utilisé dans la chaîne de caractères. Les séparateurs autorisés sont "\" et "-".

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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


ParseDMYDate()

Cette fonction renvoie un objet Date (au sens Basic) à partir d'une date formatée comme suit :

```
dd/mm/yyyy
```

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strDate* : Date stockée sous la forme d'une chaîne.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

ParseMDYDate()

Cette fonction renvoie un objet Date (au sens Basic) à partir d'une date formatée comme suit :





mm/dd/yyyy

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strDate* : Date stockée sous la forme d'une chaîne.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

ParseYMDDate()

Cette fonction convertit une chaîne de caractères représentant une date au format yyyy/mm/dd en une variable Basic de type Date.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strDate* : Date stockée sous la forme d'une chaîne.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

PMT()

Cette fonction renvoie le montant d'une annuité basée sur des versements constants et périodiques, et sur un taux d'intérêt fixe.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

$0,06/12=0,005$ soit 0,5%

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle

générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.

- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
 - 0 si les paiements sont dus à terme échu (c'est à dire en fin de période)
 - 1 si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités. Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

PPMT()

Cette fonction renvoie le montant du remboursement du capital, pour une échéance donnée, d'une annuité basée sur des versements constants et périodiques et sur un taux d'intérêt fixe.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

$0,06/12=0,005$ soit 0,5%

- *iPer* : Ce paramètre indique la période concernée par le calcul, comprise entre 1 et la valeur du paramètre *Nper*.
- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière

- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
 - 0 si les paiements sont dus à terme échu (c'est à dire en fin de période)
 - 1 si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les paramètres *Rate* et *Nper* doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités. Les sommes versées (exprimées notamment par le paramètre *Pmt*) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

PV()

Cette fonction renvoie le montant actuel d'une annuité basée sur des échéances futures constantes et périodiques, et sur un taux d'intérêt fixe.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

$0,06/12=0,005$ soit 0,5%

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.

- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
 - 0 si les paiements sont dus à terme échu (c'est à dire en fin de période)
 - 1 si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les paramètres *Rate* et *Nper* doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités. Les sommes versées (exprimées notamment par le paramètre *Pmt*) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

Randomize()





Initialise le générateur de nombres aléatoires.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *lValue* : Paramètre optionnel utilisé pour initialiser le générateur de nombres aléatoires de la fonction Rnd en lui donnant une nouvelle valeur initiale. Si ce paramètre est omis, la valeur renvoyée par l'horloge système est utilisée comme valeur initiale.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyNumber
  Randomize
  MyNumber= Int((10*Rnd)+1) : 'Renvoie une valeur
aléatoire comprise entre 1 et 10.
  RetVal=MyNumber
```

RATE()

Cette fonction renvoie le taux d'intérêt par échéance pour une annuité.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.

- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
 - 0 si les paiements sont dus à terme échu (c'est à dire en fin de période)
 - 1 si les paiements sont dus à terme à échoir (c'est à dire en début de période)
- *dblGuess* : Ce paramètre contient la valeur estimée du taux d'intérêt par échéance.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Les sommes versées (exprimées notamment par le paramètre *Pmt*) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs. Cette fonction effectue les calculs par itération, en commençant par la valeur attribuée au paramètre *Guess*. Si aucun résultat n'est trouvé au bout de vingt itérations, la fonction échoue.

RemoveRows()

Supprime dans une liste les lignes identifiées par le paramètre *strRowNames*.

Cette fonction est utile lors du traitement des valeurs d'un contrôle de type "ListBox". Les valeurs d'un tel contrôle sont représentées par des chaînes bi-dimensionnelles dont les caractéristiques sont les suivantes :

- Le caractère "|" est utilisé comme séparateur de colonnes.
- Le caractère "," est utilisé comme séparateur de lignes.
- Chaque ligne est terminée par un identifiant unique situé à droite du signe "="

Syntaxe BASIC interne

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



Champ d'application

Version : 3.5

Utilisable

AssetCenter APIs

Script de configuration d'un champ ou d'un lien 

	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strList* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strRowNames* : Identifiants des lignes à supprimer. Les identifiants sont séparés par des virgules.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
  MyStr=RemoveRows("a1|a2=a0,b1|b2=b0", "a0,c0")
:'Renvoie "b1|b2=b0"
  RetVal=MyStr
```

Replace()

Remplace toutes les occurrences du paramètre *strOldPattern* par la valeur du paramètre *strNewPattern* au sein de la chaîne de caractères contenue dans *strData*. La recherche de *strOldPattern*





peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strData* : Chaîne de caractères contenant les occurrences à remplacer.
- *strOldPattern* : Occurrence à recherche dans la chaîne de caractères contenue dans *strData*.
- *strNewPattern* : Texte remplaçant toute occurrence trouvée.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
MyStr=Replace("toimoitoimoitoi", "toi", "moi",0)
:'Renvoie "moimoimoimoimoi"
MyStr=Replace("toimoitoimoitoi", "Toi", "moi",1)
:'Renvoie "toimoitoimoitoi"
MyStr=Replace("toimoiToimoitoi", "Toi", "moi",1)
:'Renvoie "toimoimoimoitoi"
RetVal=" "
```

Right()





Renvoie iNumber caractères d'une chaîne en partant de la droite.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strString* : Chaîne de caractères à traiter.
- *iNumber* : Nombre de caractères à renvoyer.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
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()

Extrait la portion d'une chaîne de caractères située à droite du séparateur précisé dans le paramètre *strSep*.

La recherche du séparateur s'effectue de la droite vers la gauche.

La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test" :

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

Revoie la chaîne "Ceci".

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

Revoie la chaîne "Ceci_est_un".

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

Revoie la chaîne "test".

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

Revoie la chaîne "est_un_test".

RightPartFromLeft()

Extrait la portion d'une chaîne de caractères située à droite du séparateur précisé dans le paramètre *strSep*.

La recherche du séparateur s'effectue de la gauche vers la droite.

La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

Utilisable

AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test" :

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

Renvoie la chaîne "Ceci".

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

Renvoie la chaîne "Ceci_est_un".

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

Renvoie la chaîne "test".

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

Renvoie la chaîne "est_un_test".

Rmdir()

Détruit un répertoire.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strRmDirectory* : Chemin complet du répertoire à détruire.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Rnd()





Renvoie une valeur contenant un nombre aléatoire.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Paramètre optionnel dont la valeur définit le mode de génération adopté par la fonction :
 - Inférieur à zéro : Le même nombre est généré à chaque fois.
 - Supérieur à zéro : Nombre aléatoire suivant dans la série.
 - Egal à zéro : Dernier nombre aléatoire généré.
 - Omis : Nombre aléatoire suivant dans la série.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Avant d'appeler cette fonction, vous devez utiliser la fonction `Randomize`, sans aucun paramètre, pour initialiser le générateur de nombres aléatoires.

Exemple

```
Dim MyNumber
  Randomize
  MyNumber= Int((10*Rnd)+1) : 'Renvoie une valeur
aléatoire comprise entre 1 et 10.
  RetVal=MyNumber
```

RTrim()

Supprime tous les espaces suivant le dernier caractère (qui n'est pas un espace) d'une chaîne.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs

	Utilisable
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strString* : Chaîne de caractères à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
' 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()

Renvoie le nombre de secondes contenu dans la l'heure exprimée par le paramètre *tmTime*.

Syntaxe BASIC interne

Function Second(tmTime As Date) As Long

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmTime* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim strSecond
  strSecond=Second(Date())
  RetVal=strSecond : 'Renvoie le nombre de secondes
écoulées dans l'heure courante par exemple "30"
s'il est actuellement 15:45:30
```

SetSubList()



Définit les valeurs d'une sous-liste pour un contrôle "ListBox".

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	

	Utilisable
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strValues* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strRows* : Liste de valeurs à ajouter ou à substituer à celles de la chaîne contenue dans le paramètre *strValues*. Les valeurs sont séparées par le caractère "|". Les lignes traitées sont identifiées par leur identifiant, situé à droite du signe "=". Les lignes inconnues de sont pas traitées.
- *strRowFormat* : Instructions de formatage de la sous-liste. Les instructions sont séparées par le caractère "|". Ce paramètre possède les caractéristiques suivantes :
 - "1" représente les informations contenues dans la première colonne de la sous-liste.
 - "i-j" peut être utilisé pour définir un ensemble de colonnes.
 - "-" prend en compte toutes les colonnes.
 - Une colonne inconnue ne renvoie aucune valeur.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```

Dim MyStr

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
"A2|A1=a0, B2|B1=b0", "2|1") :'Renvoie
"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") :'Renvoie
"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") :'Renvoie
"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", "-") :'Renvoie
"a1|a2|a3=a0,B1|B2|B3|B4=b0,c1|c2|c3=c0"
  MyStr=SubList("A|B|C,D|E|F", "X=*", "2")
:'Renvoie "A|X|C,D|X|F"
  RetVal=""

```

Sgn()

Renvoie une valeur indiquant le signe d'un nombre.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le signe.

Sortie

La fonction peut renvoyer une des valeurs suivante :

- 1 : Le nombre est supérieur à zéro.
- 0 : Le nombre est égal à zéro.
- -1 : Le nombre est inférieur à zéro.

Exemple

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

Shell()

Lance un programme exécutable.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strExec* : Chemin complet de l'exécutable à lancer.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

Sin()





Renvoie le sinus d'un nombre, exprimé en radians.

Syntaxe BASIC interne

Function Sin(*dValue* As Double) As Double

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le sinus.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

Space()

Crée une chaînes de caractères comprenant le nombre d'espaces indiqué par le paramètre *iSpace*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *iSpace* : Nombre d'espaces à insérer dans la chaîne.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Remarques



Note : Cette fonction peut servir à formater des chaînes ou à effacer des données dans des chaînes de longueur fixe.

Exemple

```
Dim MyString
' Renvoie une chaîne de 10 espaces.
MyString = Space(10)
:' Insère 10 espaces entre deux chaînes.
MyString = "Espace" & Space(10) & "inséré"
RetVal=MyString
```

Sqr()

Renvoie la racine carrée d'un nombre.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la racine carrée.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

Str()

Convertit un nombre en une chaîne de caractères.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

Utilisable

AssetCenter APIs

Script de configuration d'un champ ou d'un
lien



	Utilisable
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strValue* : nombre à convertir en chaîne.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

StrComp()





Effectue la comparaison entre deux chaînes de caractères.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strString1* : Première chaîne de caractères.
- *strString2* : Deuxième chaîne de caractères.
- *iOptionCompare* : type de comparaison. Ce paramètre peut prendre la valeur "0" pour une comparaison binaire, ou "1" pour une comparaison du texte des deux chaînes.

Sortie

- -1 : *strString1* est supérieure à *strString2*.
- 0 : *strString1* est égale à *strString2*.
- 1 : *strString1* est inférieure à *strString2*.

String()

Renvoie une chaîne composée de *iCount* fois le caractère *strString*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *iCount* : Nombre d'occurrences du caractère *strString*.
- *strString* : caractère utilisé pour la composition de la chaîne.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

SubList()





Renvoie une sous-liste d'une liste de valeurs contenue dans une chaîne de caractères représentant les valeurs d'un contrôle "ListBox".

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strValues* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strRows* : Identifiants des lignes à inclure dans la sous-liste. Les identifiants sont séparés par une virgule. Certains jokers sont acceptés :
 - "*" inclut tous les identifiants dans la sous-liste.
 - Un identifiant inconnu renvoie une valeur vide pour la sous-liste.
- *strRowFormat* : Instructions de formatage de la sous-liste. Les instructions sont séparées par le caractère "|". Ce paramètre possède les caractéristiques suivantes :

- "1" représente les informations contenues dans la première colonne de la liste dont on extrait une sous-liste.
- "0" représente l'identifiant de la ligne de la liste dont on extrait une sous-liste.
- "*" représente les informations contenues dans toutes les colonnes (à l'exception de l'identifiant de la ligne).
- Une colonne inconnue ne renvoie aucune valeur.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
"a0,b0,a0", "3|2|3") :'Renvoie
"a3|a2|a3,b3|b2|b3,a3|a2|a3"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
"*", "*|0") :'Renvoie
"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") :'Renvoie
"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") : 'Renvoie "=a0,=b0,=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
"z0", "*=0") : 'Renvoie " "

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
"*", "=1") : 'Renvoie "=a1,=b1,=c1"
  MyStr=SubList("A|B|C,D|E|F", "*", "2=0")
: 'Renvoie "B,E"
  RetVal=" "

```

Tan()





Renvoie la tangente d'un nombre, exprimé en radians.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la tangente.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Exemple

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

Time()

Renvoie l'heure courante.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la fonction AmLastErrorMsg()) pour savoir si une erreur s'est produite (et son message associé).

Timer()





Revoie le nombre de secondes écoulées depuis 12:00 AM.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.

- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

TimeSerial()

Cette fonction renvoie une heure formatée en fonction des paramètres *iHour*, *iMinute* et *iSecond*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *iHour* : Heure.
- *iMinute* : Minutes.
- *iSecond* : Secondes.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

Chacun de ces paramètres peut prendre pour valeur une expression numérique représentant un nombre d'heures, de minutes ou de secondes. Ainsi l'exemple suivant :

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

renvoie la valeur :

```
3:50:00
```

Lorsque la valeur d'un paramètre est en dehors de l'intervalle de valeurs généralement admis (c'est à dire 0-59 pour les minutes et les secondes et 0-23 pour les heures), elle est convertie vers le paramètre immédiatement supérieur. Ainsi, si vous entrez "75" comme valeur pour le paramètre *iMinute*, ce dernier sera interprété comme 1 heure et 15 minutes.

L'exemple suivant :

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

renvoie la valeur :

```
16:50:45
```

TimeValue()

Cette fonction renvoie la partie heure d'une valeur "Date+Heure"

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmTime* : Date au format "Date+Heure".

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

L'exemple suivant :

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

renvoie la valeur :

```
15:00:00
```

ToSmart()

Cette fonction reformate un chaîne source en mettant des majuscules au début de chaque mot.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strString* : Chaîne source à reformater.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Trim()

Supprime tous les espaces précédant le premier caractère (qui n'est pas un espace) d'une chaîne et tous les espaces suivant le dernier caractère (qui n'est pas un espace) d'une chaîne.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strString* : Chaîne de caractères à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
' 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()

Passer tous les caractères d'une chaîne en majuscules.

Syntaxe BASIC interne

Function UCase(strString As String) As String

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strString* : Chaîne de caractères à passer en majuscules.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
' 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()

Supprime tous les caractères d'échappement d'une chaîne de caractères.

Syntaxe BASIC interne





```

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

```

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strSource* : Chaîne de caractères à traiter.
- *strEscChar* : Caractère d'échappement à supprimer.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
MyStr=UnEscapeSeparators("toi\|moi\|toi\|", "\")
:'Renvoie la valeur "toi|moi|toi|"
RetVal=""
```

Union()

Rassemble deux chaînes de caractères délimitées par des séparateurs. Les doublons sont supprimés.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *strListOne* : Première chaîne de caractères.
- *strListTwo* : Deuxième chaîne de caractères.
- *strSeparator* : Séparateur utilisé pour délimiter les éléments contenus dans les chaînes.
- *strEscChar* : Caractère d'échappement. Si ce caractère préfixe le séparateur, ce dernier est ignoré.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim MyStr
MyStr=Union("a1|a2,b1|b2", "a1|a3,b1|b2", ",", "\", "
") : 'Renvoie la valeur "a1|a2,b1|b2,a1|a3"
MyStr=Union("a1|a2,b1|b2", "a1|a3\",b1|b2", ",", " ,
```

```
"\" ) : 'Renvoie la valeur
"a1|a2,b1|b2,a1|a3\",b1|b2"
RetVal=" "
```

UTCToLocalDate()





Cette fonction convertit une date au format UTC (indépendante d'un quelconque fuseau horaire) en une date au format "Date+Heure".

Syntaxe BASIC interne

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

Champ d'application

Version : 3.5

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *tmUTC* : Date au format UTC.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction AmLastError() (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

Val()





Convertit une chaîne de caractères représentant un nombre en un nombre de type "Double".

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	
Action de type "Script"	
Script d'un assistant	
Script FINISH.DO d'un assistant	

Entrée

- *strString* : Chaîne à convertir.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la

fonction `AmLastErrorMsg()` pour savoir si une erreur s'est produite (et son message associé).

Exemple

```
Dim strYear
Dim dYear as Double
  strYear=Year(Date())
  dYear=Val(strYear)
  RetVal=dYear : 'Renvoie l'année en cours
```

WeekDay()

Renvoie le jour de la semaine contenu dans la date exprimée par le paramètre *tmDate*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

Le nombre retourné correspond à un jour de la semaine, le "1" représentant le dimanche, le "2" le lundi, ..., le "7" le samedi.

Exemple

```
Dim strWeekDay
    strWeekDay=WeekDay(Date())
RetVal=strWeekDay : 'Renvoie le jour de la semaine'
```

Year()

Renvoie l'année contenue dans la date exprimée par le paramètre *tmDate*.

Syntaxe BASIC interne

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

Champ d'application

Version : 3.00

	Utilisable
AssetCenter APIs	
Script de configuration d'un champ ou d'un lien	✓
Action de type "Script"	✓
Script d'un assistant	✓
Script FINISH.DO d'un assistant	✓

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, deux cas de figure se présentent :

- Sous AssetCenter, l'exécution du script contenant la fonction est interrompue et un message d'erreur est envoyé à l'utilisateur.
- Dans le cas d'un appel par le biais d'un programme externe, vous devez appeler la fonction `AmLastError()` (et éventuellement la fonction `AmLastErrorMsg()`) pour savoir si une erreur s'est produite (et son message associé).

IV. Index

9 Fonctions disponibles -

Domaine : Tous

CHAPITRE

- Abs
- AmActionDde
- AmActionExec
- AmActionMail
- AmActionPrint
- AmActionPrintPreview
- AmActionPrintTo
- AmAddAllPOLinesToInv
- AmAddCatRefAndCompositionToPOrder
- AmAddCatRefToPOrder
- AmAddEstimLinesToPO
- AmAddEstimLineToPO
- AmAddPOLineToInv
- AmAddPOrderLineToReceipt
- AmAddReceiptLineToInvoice
- AmAddReqLinesToEstim

- AmAddReqLinesToPO
- AmAddReqLineToEstim
- AmAddReqLineToPO
- AmAddRequestLineToPOOrder
- AmAddTemplateLineToPOOrder
- AmAddTemplateToPOOrder
- 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
- AmEvalScript
- AmExecTransition
- AmExecuteActionById
- AmExecuteActionByName
- AmExportDocument
- AmFindCable
- AmFindDevice

- AmFindRootLink
- AmFindTermDevice
- AmFindTermField
- AmGenSqlName
- 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
- AmGetNextAssetPin
- AmGetNextAssetPort
- AmGetNextCableBundle
- AmGetNextCablePair
- AmGetNTDomains
- AmGetNTMachinesInDomain
- AmGetNTUsersInDomain
- AmGetPOLinePrice
- AmGetPOLinePriceCur
- AmGetPOLinePricing
- 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
- AmUpdateLoginSlot
- AmUpdateRecord
- AmValueOf
- AmWizChain
- AmWorkTimeSpanBetween
- AppendOperand
- ApplyNewVals
- Asc
- Atn
- BasicToLocalDate
- BasicToLocalTime
- BasicToLocalTimeStamp
- Beep
- CDb1

- ChDir
- ChDrive
- Chr
- CInt
- CLng
- Cos
- CountOccurrences
- CountValues
- CSng
- CStr
- CurDir
- CVar
- Date
- DateSerial
- DateValue
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileLen
- Fix
- FormatDate
- FormatResString
- FormatString
- 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

10 Fonctions disponibles - Domaine : Technique

CHAPITRE

- AmCalcConsolidatedFeature
- AmCleanup
- AmClearLastError
- AmCloseAllChildren
- AmCloseConnection
- AmCommit
- AmConvertCurrency
- AmConvertDateBasicToUnix
- AmConvertDateIntlToUnix
- AmConvertDateStringToUnix
- AmConvertDateUnixToBasic
- AmConvertDateUnixToIntl
- AmConvertDateUnixToString
- AmConvertDoubleToString
- AmConvertMonetaryToString
- AmConvertStringToDouble

- AmConvertStringToMonetary
- AmCounter
- AmCreateLink
- AmCreateRecord
- AmCryptPassword
- AmCurrentDate
- AmCurrentIsoLang
- AmCurrentLanguage
- AmCurrentServerDate
- AmDateAdd
- AmDateAddLogical
- AmDateDiff
- AmDbGetDate
- AmDbGetDouble
- AmDbGetList
- AmDbGetListEx
- AmDbGetLong
- AmDbGetPk
- AmDbGetString
- AmDbGetStringEx
- AmDefaultCurrency
- AmDeleteLink
- AmDeleteRecord
- AmDuplicateRecord
- AmEvalScript
- AmExportDocument
- AmGenSqlName
- AmGetComputeString
- AmGetCurrentNTDomain
- AmGetCurrentNTUser
- AmGetFeat

- AmGetFeatCount
- AmGetField
- AmGetFieldCount
- AmGetFieldDateValue
- AmGetFieldDescription
- AmGetFieldDoubleValue
- AmGetFieldFormat
- AmGetFieldFormatFromName
- AmGetFieldFromName
- AmGetFieldLabel
- AmGetFieldLabelFromName
- AmGetFieldLongValue
- AmGetFieldName
- AmGetFieldRights
- AmGetFieldSize
- AmGetFieldSqlName
- AmGetFieldStringValue
- AmGetFieldType
- AmGetFieldUserType
- AmGetForeignKey
- AmGetIndex
- AmGetIndexCount
- AmGetIndexField
- AmGetIndexFieldCount
- AmGetIndexFlags
- AmGetIndexName
- AmGetLink
- AmGetLinkCardinality
- AmGetLinkCount
- AmGetLinkDstField
- AmGetLinkFeatureValue

- AmGetLinkFromName
- AmGetLinkType
- AmGetMainField
- AmGetNTDomains
- AmGetNTMachinesInDomain
- AmGetNTUsersInDomain
- AmGetRecordFromMainId
- AmGetRecordHandle
- AmGetRecordId
- AmGetRelDstField
- AmGetRelSrcField
- AmGetRelTable
- AmGetReverseLink
- AmGetSelfFromMainId
- AmGetSourceTable
- AmGetTable
- AmGetTableCount
- AmGetTableDescription
- AmGetTableFromName
- AmGetTableLabel
- AmGetTableName
- AmGetTableRights
- AmGetTableSqlName
- AmGetTargetTable
- AmGetTypedLinkField
- AmGetVersion
- AmHasAdminPrivilege
- AmHasRelTable
- AmImportDocument
- AmInsertRecord
- AmIsConnected

- AmIsFieldForeignKey
- AmIsFieldIndexed
- AmIsFieldPrimaryKey
- AmIsLink
- AmIsTypedLink
- AmLastError
- AmLastErrorMsg
- AmListToString
- AmLoginId
- AmLoginName
- AmMsgBox
- AmOpenConnection
- AmQueryCreate
- AmQueryExec
- AmQueryGet
- AmQueryNext
- AmQuerySetAddMainField
- AmQuerySetFullMemo
- AmQueryStartTable
- AmQueryStop
- AmRefreshAllCaches
- AmReleaseHandle
- AmRevCryptPassword
- AmRgbColor
- AmRollback
- AmSetFieldDateValue
- AmSetFieldDoubleValue
- AmSetFieldLongValue
- AmSetFieldStrValue
- AmSetLinkFeatureValue
- AmSqlTextConst

- AmStartTransaction
- AmStartup
- AmTableDesc
- AmUpdateLoginSlot
- AmUpdateRecord

11 | Fonctions disponibles - Domaine : Achats

CHAPITRE

- AmAddAllPOLinesToInv
- AmAddCatRefAndCompositionToPOrder
- AmAddCatRefToPOrder
- AmAddEstimLinesToPO
- AmAddEstimLineToPO
- AmAddPOLineToInv
- AmAddPOrderLineToReceipt
- AmAddReceiptLineToInvoice
- AmAddReqLinesToEstim
- AmAddReqLinesToPO
- AmAddReqLineToEstim
- AmAddReqLineToPO
- AmAddRequestLineToPOrder
- AmAddTemplateLineToPOrder
- AmAddTemplateToPOrder
- AmAddTemplateToRequest

- AmCreateAssetsAwaitingDelivery
- AmCreateDelivFromPO
- AmCreateEstimFromReq
- AmCreateEstimsFromAllReqLines
- AmCreateInvFromPO
- AmCreatePOFromEstim
- AmCreatePOFromReq
- AmCreatePOrderFromRequest
- AmCreatePOrdersFromRequest
- AmCreatePOsFromAllReqLines
- AmCreateReceiptFromPOrder
- AmCreateRequestToInvoice
- AmCreateRequestToPOrder
- AmCreateRequestToReceipt
- AmCreateReturnFromReceipt
- AmGetPOLinePrice
- AmGetPOLinePriceCur
- AmGetPOLinePricing
- AmGetPOLineReference
- AmInstantiateReqLine
- AmInstantiateRequest
- AmMapSubReqLineAgent
- AmReceiveAllPOLines
- AmReceivePOLine
- AmReturnAsset
- AmReturnContract
- AmReturnPortfolioItem
- AmReturnTraining
- AmReturnWorkOrder

12 Fonctions disponibles -

Domaine : Fonctionnel

CHAPITRE

- AmBusinessSecondsInDay
- AmCalcDepr
- AmComputeAllLicAndInstallCounts
- AmComputeLicAndInstallCounts
- AmDeadLine
- AmEndOfNthBusinessDay
- AmTaxRate
- AmWorkTimeSpanBetween

13 | Fonctions disponibles - Domaine : HelpDesk

CHAPITRE

- AmDefAssignee
- AmDefEscalationScheme
- AmDefGroup

14 | Fonctions disponibles - Domaine : Refacturation

CHAPITRE

- AmCbkReplayEvent

15 | Fonctions disponibles -

Domaine : Câble

CHAPITRE

- 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

16 Fonctions disponibles -

Domaine : Actions

CHAPITRE

- `AmActionDde`
- `AmActionExec`
- `AmActionMail`
- `AmActionPrint`
- `AmActionPrintPreview`
- `AmActionPrintTo`
- `AmExecuteActionById`
- `AmExecuteActionByName`



17 | Fonctions disponibles -

Domaine : Interface Utilisateur

CHAPITRE

- `AmOpenScreen`

18 Fonctions disponibles -

Domaine : Builtin

CHAPITRE

- Abs
- AppendOperand
- ApplyNewVals
- Asc
- Atn
- BasicToLocalDate
- BasicToLocalTime
- BasicToLocalTimeStamp
- Beep
- CDb1
- ChDir
- ChDrive
- Chr
- CInt
- CLng
- Cos

- CountOccurrences
- CountValues
- CSng
- CStr
- CurDir
- CVar
- Date
- DateSerial
- DateValue
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileLen
- Fix
- FormatDate
- FormatResString
- FormatString
- 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

19 Fonctions disponibles -

Domaine : Assistants

CHAPITRE

- `AmDecrementLogLevel`
- `AmExecTransition`
- `AmIncrementLogLevel`
- `AmLog`
- `AmPagePath`
- `AmProgress`
- `AmRefreshProperty`
- `AmSetProperty`
- `AmUpdateDetail`
- `AmValueOf`
- `AmWizChain`

