

VFP2C32 帮助文件

Ver: 2.0.0.15

翻译: xinjie

QQ:411618689

2020-09-17

目 录

VFP2C32	1
错误信息	1
目录结构	1
软件包.....	1
如何使用.....	2
库的初始化.....	2
部署方式.....	2
Intellisense 和上下文帮助.....	2
安装 Intellisense.....	2
特殊脚本.....	3
特例.....	3
上下文帮助.....	3
安装上下文帮助	3
版本历史 (机翻)	4
开发人员和合作者	14
VFP2C32Front.....	14
转换选项	16
VFP baseclass (VFP 基类)	16
Character set (字符集设置)	16
Pragma pack (对齐方式)	16
Buffered (缓冲)	16
Array indexing (数组索引)	16
Treat type BYTE as (将 BYTE 类型视为)	17
ReadOnly (只读)	17
Assertions (断点)	17
Memory allocator (内存分配器)	17

Memory management (内存管理)	17
常见错误	17
示例.....	18
GlobalMemoryStatusEx.....	18
RegisterPowerSettingNotification	20
参考.....	26
按字母排序	26
A	26
B	28
C.....	28
D	29
E	29
F	29
G	30
I	31
L	31
M	31
N	32
O	32
P	32
R.....	32
S.....	34
T	35
U	35
V.....	36
W	36
按功能.....	37
数组	38
AAverage (数组元素的算术平均值)	38

AMax (数组元素中的最大值)	39
AMin (数组元素中的最小值)	40
ASum (数组元素的合计)	41
C 数组仿真.....	41
MarshalCArray2Cursor (C 数组转换为 VFP Cursor)	42
MarshalCArray2FoxArray (C 数组转换为 VFP 数组)	43
MarshalCursor2CArray (VFP Cursor 转换为 C 数组)	44
MarshalFoxArray2CArray (VFP 数组转换为 C 数组)	46
C 回调和事件.....	47
AsyncWaitForObject (异步等待)	47
BindEventsEx (高级绑定)	51
CancelWaitForObject (取消异步等待)	54
CreateCallbackFunc (创建回调函数)	55
CreatePublicShadowObjReference (创建公共影子对象引用)	57
DestroyCallbackFunc (取消回调函数)	58
ReleasePublicShadowObjReference (释放公共影子对象引用)	59
UnbindEventsEx (取消高级绑定)	59
C 结构仿真.....	60
ReadBytes	63
ReadChar.....	63
ReadCharArray.....	63
ReadCString	63
ReadDouble	63
ReadFloat.....	63
ReadInt	63
ReadInt64.....	63
ReadInt8	63
ReadLogical.....	63
ReadPChar	63

ReadPCString.....	63
ReadPDouble.....	63
ReadPFloat	63
ReadPInt.....	63
ReadPInt64.....	64
ReadPInt8	64
ReadPLogical	64
ReadPointer.....	64
ReadPPointer	64
ReadProcessMemoryEx	64
ReadPShort	64
ReadPUInt.....	64
ReadPUInt64.....	64
ReadPUInt8	64
ReadPUShort	64
ReadPWString	64
ReadShort.....	64
ReadUInt	64
ReadUInt64	64
ReadUInt8.....	65
ReadUShort.....	65
ReadWCharArray.....	65
ReadWString.....	65
WriteBytes	65
WriteChar.....	65
WriteCharArray	65
WriteCString	65
WriteDouble	65
WriteFloat	65

WriteGPCString.....	65
WriteInt.....	65
WriteInt64.....	65
WriteInt8	65
WriteLogical.....	65
WritePChar.....	66
WritePCString.....	66
WritePDouble.....	66
WritePFloat.....	66
WritePInt.....	66
WritePInt64.....	66
WritePInt8.....	66
WritePLogical	66
WritePointer.....	66
WritePPointer	66
WritePShort	66
WritePUInt.....	66
WritePUInt64.....	66
WritePUInt8	66
WritePUShort	66
WritePWChar.....	67
WritePWString	67
WriteShort.....	67
WriteUInt	67
WriteUInt64.....	67
WriteUInt8	67
WriteUShort.....	67
WriteWChar	67
WriteWCharArray.....	67

WriteWString.....	67
COM	67
CLSIDFromProgID (从 ProgID 返回 CLSID)	68
CLSIDFromString(将可识别 CLSID 转换为二进制格式)	69
CreateGuid (创建 GUID)	70
CreateThreadObject (创建线程对象)	71
GetIUnknown (获取 IUnknown 接口的指针)	76
IsEqualGuid (比较 GUID)	77
ProgIDFromCLSID (从 CLSID 返回 ProgID)	78
RegisterActiveObject (注册活动对象)	79
RegisterObjectAsFileMoniker (注册对象为文件名字对象)	80
RevokeActiveObject (撤销活动对象)	82
StringFromCLSID (从 CLSID 返回可识别字符串)	82
公共对话框.....	83
GetOpenFileName (打开文件对话框)	84
GetSaveFileName (保存文件对话框)	87
MessageBoxEx (高级消息框)	90
SHBrowseFolder	94
转换.....	96
Colors2RGB	97
Double2Str	99
Float2Str	100
Int642Str.....	101
Long2Str.....	102
Num2Binary.....	103
PG_ByteA2Str.....	105
PG_Str2ByteA.....	106
RGB2Colors	107
Short2Str.....	108

Str2Double	109
Str2Float	110
Str2Int64	112
Str2Long	113
Str2Short	114
Str2UInt64	115
Str2ULong	117
Str2UShort	118
UInt642Str	119
ULong2Str	121
UShort2Str	122
Value2Variant	123
Variant2Value	124
日期和时间	125
ATimeZones	126
Double2DT	128
DT2Double	128
DT2FT	129
DT2ST	130
DT2Timet	131
DT2UTC	132
FT2DT	133
GetSystemTime	134
SetSystemTime	135
ST2DT	136
Timet2DT	137
UTC2DT	138
文件系统	139
ADirectoryInfo	139

ADirEx	141
ADriveInfo	143
AFileAttributes	145
AFileAttributesEx.....	146
CancelFileChange	148
CompareFileTimes.....	149
CopyFileEx.....	150
DeleteDirectory	152
DeleteFileEx.....	153
FindFileChange	154
GetFileAttributes	157
GetFileOwner.....	158
GetFileSize.....	159
GetFileTimes	161
GetLongPathName	162
GetShortPathName	164
MoveFileEx	165
SetFileAttributes.....	167
SetFileTimes.....	168
通用库.....	170
AErrorEx.....	170
FormatMessageEx	171
InitVFP2C32.....	172
VFP2CSys	174
低级文件处理	175
AFHandlesEx.....	175
FChSizeEx.....	177
FCloseEx	178
FCreateEx	179

FEoFEx.....	182
FFlushEx.....	183
FGetsEx	184
FLockFile.....	185
FLockFileEx.....	186
FOpenEx.....	188
FPutsEx	190
FReadEx.....	192
FSeekEx.....	193
FUnlockFile.....	194
FUnlockFileEx	195
FWriteEx	196
内存管理.....	198
AllocHGlobal.....	198
AllocMem	199
AllocMemTo.....	200
AMemBlocks.....	202
CompactMem	203
FreeHGlobal.....	203
FreeMem.....	204
FreePMem	205
FreeRefArray.....	206
LockHGlobal	207
ReAllocHGlobal.....	208
ReAllocMem.....	210
SizeOfMem.....	211
UnlockHGlobal.....	212
ValidateMem	213
杂项	214

AFontInfo.....	214
ASplitStr.....	215
Decimals.....	216
GetCursorPosEx.....	217
Int64_Add	218
Int64_Div	220
Int64_Mod	222
Int64_Mul.....	223
Int64_Sub.....	225
网络.....	226
AbortUrlDownloadToFileEx	227
AlpAddresses.....	227
ANetFiles.....	228
ANetServers.....	230
GetServerTime	231
IcmpPing	232
Ip2MacAddress.....	234
ResolveHostTolp.....	235
SyncToSNTPServer.....	236
UrlDownloadToFileEx.....	237
ODBC	239
ASQLDatasources.....	240
ASQLDrivers.....	241
ChangeSQLDataSource.....	242
CreateSQLDataSource.....	243
DeleteSQLDataSource.....	244
SQLCancelEx	245
SQLExecEx.....	245
SQLGetPropEx.....	250

SQLPrepareEx	251
SQLSetPropEx	255
打印机信息	256
APaperSizes	256
APrinterForms	257
APrintersEx	258
APrinterTrays	262
APrintJobs	263
进程信息	265
AHeapBlocks	265
AProcesses	266
AProcessHeaps	267
AProcessModules	268
AProcessThreads	269
RAS(远程访问)	269
AbortRasConnectionNotification	270
ARasConnections	271
ARasDevices	272
ARasPhonebookEntries	273
RasClearConnectionStatistics	274
RasConnectionNotificationEx	275
RasDialDlgEx	277
RasDialEx	278
RasGetConnectStatusEx	281
RasHangUpEx	282
RasPhonebookDlgEx	283
注册表	285
ARegistryKeys	286
ARegistryValues	289

CancelRegistryChange	292
CloseRegistryKey	293
CreateRegistryKey	293
DeleteRegistryKey	298
FindRegistryChange	301
OpenRegistryKey	304
ReadRegistryKey	308
RegistryHiveToObject	311
RegistryValuesToObject	314
WriteRegistryKey	317
资源信息	321
AResourceLanguages	321
AResourceNames	322
AResourceTypes	323
服务管理	324
ADependentServices	324
AServiceConfig	326
AServices	327
AServiceStatus	329
CloseServiceHandle	331
ContinueService	332
ControlService	333
CreateService	334
OpenService	337
PauseService	339
StartService	340
StopService	341
WaitForServiceStatus	343
Shell	345

SHCopyFiles.....	345
SHDeleteFiles	347
SHMoveFiles.....	348
SHRenameFiles	350
SHSpecialFolder	351
系统信息.....	355
ADesktopArea	356
ADesktops	357
ADisplayDevices	358
AResolutions.....	359
AWindowStations	360
ExpandEnvironmentStrings	361
GetLocaleInfoEx.....	362
GetSystemDirectory	363
GetWindowsDirectory.....	363
OsEx.....	364
磁盘卷信息.....	365
AVolumeInformation	366
AVolumeMountPoints	367
AVolumePaths	367
AVolumes	368
窗体信息.....	369
AWindowProps	369
AWindows	370
AWindowsEx	371
CenterWindowEx.....	373
GetWindowRectEx	374
GetWindowTextEx.....	375
附录.....	376

VFP2C32 Ver:2.0.0.16 新增函数.....	376
AMonitors.....	376
FindFileChangeEx.....	376
译者写在最后.....	376

VFP2C32

VFP2C32 (“VFP TO C 32 bit”的缩写) 是 Visual Foxpro 和 C 编程语言之间的桥梁。此外，它还封装了许多 Windows API 函数。

使用此库，您可以在 VFP 中创建任何 C 数据类型：结构、union 和数组。在 [VFP2C32Front](#) 一节中解释了如何为 C 结构或 union 类型创建 VFP 封装类的过程。你还可以使用 vfp2carray.prg 中的类创建 C 数组。

Windows API 的封装允许您使用 Windows 内置的各种功能。例如注册表 api，用于查询打印机信息和控制 Windows 服务。要了解此方法内容，请查看 “[按功能](#)” 一节。。

错误信息

错误编号	描述	原因
9	数据类型不匹配。	参数类型不正确。
11	函数参数的值、类型或数量无效。	参数无效或者数量不正确。
43	无足够内存完成此操作。	内存分配失败。
1754	找不到入口点。	所使用功能未使用 InitVFP2C32 初始化, 或功能在当前操作系统不可用。
1098	包含系统或自定义错误消息。	调用 AErrorEX 可获得更详细的错误信息。

目录结构

软件包

该库发行版目录结构如下：

vfp2c32	包含 FLL 和所有支持文件。
vfp2c32examples	示例项目
vfp2c32front	VFP2C32 Front 是一个支持程序，它可将 C 的 struct、union、enum 类型转换为 VFP 代码。

如何使用

将 VFP2C32.dll 和 vfp2c.h 添加至项目中——这两个文件都可以从应用程序/可执行文件中排除(译者注：可在项目中将它们设置为“排除”)。

Vfp2c32t.dll 是该库的线程安全版本，您可以在多线程 VFP COM DLL 中使用它。

库的初始化

```
&& 在程序启动代码的某个位置
#include vfp2c.h
Set Library To vfp2c32.dll Additive

If !InitVFP2C32(VFP2C_INIT_ALL) && 您可以根据需要更改初始化参数
Local laError[1], InCount, xj, lcError

InCount = AERROREX('laError')
lcError = 'VFP2C32 初始化失败: ' + Chr(13)

For xj = 1 To InCount
lcError = lcError + ;
'错误编号: ' + Transform(laError[1]) + Chr(13) + ;
'功能: ' + laError[2] + Chr(13) + ;
'消息: ' + laError[3] + ""

Endfor

&& 记录或显示错误并中止程序初始化
Endif
```

部署方式

仅需将 vfp2c32.dll/vfp2c32t.dll 和 msocr71.dll 的副本放在应用程序目录中。

注意，不要放置在 Windows\System32 中。

Intellisense 和上下文帮助

安装 Intellisense

要为 VFP2C32 中的功能安装 Intellisense 支持，请从子目录 “vfp2c32 \ help” 中执行程序

“vfp2cinstallintelli.prg”。

特殊脚本

编写代码时，如果要列出所有函数，请键入“vfp2c”，然后按空格键

特例

以下函数名超出了 Intellisense 26 个字符的限制：

- AbortRasConnectionNotification
- CreatePublicShadowObjReference
- RasClearConnectionStatistics
- RasConnectionNotificationEx
- RegisterObjectAsFileMoniker
- ReleasePublicShadowObjReference

当弹出这些功能的提示时，仅显示前 26 个字符，此时无需介意，当输入完参数后，键入“，”而不是“)” ，脚本即可将函数名自动更正。

上下文帮助

Vfp2c32 中的功能具有相关的上下文帮助，该脚本与 VFP 帮助系统无缝集成。

仅需在编辑器窗口中将所需要帮助的函数高亮显示（译者注：使用鼠标将函数名全选），然后按 F1 即可直接进入该 CHM 文件（译者注：英文版帮助）中的帮助主题。

安装上下文帮助

上下文帮助脚本取决于 Intellisense 脚本，因此请先安装脚本。然后将 vfp2c32_help.prg 和 vfp2c32.chm 复制到您的 VFP 安装目录中。

该脚本需要在 VFP 每次启动时执行。要实现此目的，请将下面的代码添加到 _STARTUP 脚本中：

如果您还没有 _STARTUP 脚本，则只需将文件 _startup.prg 复制到 VFP 安装目录中，然后单击菜单“工具-选项”，在打开的对话框中，选择“文件位置”选项卡，在“启动程序”中予以指定。

版本历史 (机翻)

版本 2.0.0.15-2017-03-18

- 增强 CopyFileEx 和 MoveFileEx-新的可选参数 nShareMode, 用于指定打开复制/移动文件的共享模式。

版本 2.0.0.14-2016-01-04

- 增强 DT2Timet
- 新的子项目 vfpsrvhost-一个小型 C ++ 主机可执行文件, 可通过 VFP COM 对象创建真实的 Windows 服务。它并没有 100% 完成, 但是无论如何我会在它在 HDD 上腐烂之前将其发布

版本 2.0.0.13-2013-08-05

- 增强 SQLExecEx

SQL_VARCHAR 和 SQL_WVARCHAR 列现在已映射到 C (x) 数据类型, 就像在 SQLExec 中一样, 使用 CURSORSETPROP ('MapVarchar', .T., 0) 更改到 V (x) 数据类型的映射; 使用 SQL Native Client ODBC 驱动程序时, 可以将 SQL Server 的 varchar (max) / varbinary (max) 字段正确地作为 Memo / Blob 字段检索

- 添加了 SQLPrepareEx 和 SQLCancelEx
- 在 vfp2c.h 中添加了 CreateGUID 的常量
- 修复 MemberData 在 VFP2CFront 中的生成

版本 2.0.0.11-2012-02-04

- 增强的 CreateThreadObject: 返回的代理对象实现 2 个新方法: GetCallQueueSize 和 GetCallRunTime
- 增强的 GetSaveFileName: 如果通过了文件扩展名过滤器, 该函数会将所选扩展名附加到输入的文件名上

版本 2.0.0.10-2011-08-11

- 修复了 CreateThreadObject 中的错误, 该错误可能导致传递的回调对象上的对象悬空

版本 2.0.0.9-2011-05-24

- 删除了对 msdp71.dll 的依赖
- 修复了 FGetsEx 中的错误: 检索文件中的行时文件指针设置错误, 并且当前文件指针位置大于

4GB。现在，就换行符回车处理而言，FGetsEx 的行为与 FGets 相同。

- 修复了 ANetFiles, ANetServers 中的错误：报告的文件/服务器名称被截断了。
- 将许多功能更改为直接链接而不是动态运行时链接，这有两个含义：
 1. VFP2C32 现在支持的最小操作系统是 Windows 2000
 2. 大多数函数不再需要通过调用 InitVFP2C32 来初始化。
- 创建了该库的线程安全版本 (vfp2c32t.dll)，您可以在 Visual FoxPro COM 服务器内部使用该版本或将其用于多线程开发。
- 添加的功能：CreateThreadObject 允许您在其自己的线程上创建 COM 对象，并异步触发这些对象上的函数调用-或简单地说：创建多线程 FoxPro 程序。

版本 2.0.0.8-2011-01-27

- 修复了 BindEventsEx 中的错误-如果在 “cParmDefinition” 参数中传递了空字符串，则将使用默认参数而不是没有参数来调用回调函数。
- 修复了 AProcessModules 中的潜在错误-该函数有时可能会失败。
- 修复了 AAverage 中的错误-如果所有值的总和超过最大货币或两倍限制，则结果是错误的。
- 现在，ASum 会引发错误 1988 “货币值超出范围”。如果结果溢出。
- 修复了 SHCopyFiles, SHMoveFiles, SHRenameFiles 和 SHDeleteFiles 中的潜在错误，即使操作中止，该函数也可以返回 1 表示成功。
- 修复了如果存在 32 个以上的活动 RAS 连接，则 ARasConnections 中可能存在的错误
- 扩展了 AllocHGlobal 和 ReAllocHGlobal，使用秒参数可以更改分配选项。
- 扩展的 CreateCallbackFunc：为 STRING, INT64 和 UINT64 数据类型添加了更多封送处理修改器。
- 扩展的 Colors2RGB 和 RGB2Colors：现在可以选择设置或检索颜色的 Alpha 通道。
- 扩展的 FCreateEx, FOpenEx, DeleteFileEx：现在甚至支持名称超过 MAX_PATH 的文件。
- 扩展的 Int64_Add, Int64_Sub, Int64_Mul, Int64_Div 和 Int64_Mul：这些函数现在在数字溢出或除零时抛出错误，并且额外的参数允许指定返回值的格式。
- 扩展的 RasDialDlgEx：两个新的可选参数允许指定要拨打的子条目和所有者窗口。
- 扩展了 ReadInt64, ReadPInt64, ReadUInt64 和 ReadPUInt64：一个附加参数允许以不同格式返回 64 位整数。
- 扩展的 WriteInt64, WritePInt64, WriteUInt64 和 WritePUInt64：现在的功能除外，更多格

式的 64 位值除外。

- 重写了所有必须通过 DECLARE 才能使用 FLL 导出机制的封送处理功能，此外，如果传递了 0 指针，则所有封送处理功能现在都会引发错误“无效参数”，并且所有 ReadPxxx 函数现在都返回 NULL。如果传递的地址包含 0 指针。
- 新增功能：
 - OsEx
 - AVolumes
 - AVolumeMountPoints
 - AVolumePaths
 - CreateService
 - DeleteService
 - MessageBoxEx
 - Int64ToStr
 - Str2Int64
 - UInt64ToStr
 - Str2UInt64
- 内部代码清理

重大变化！

- 更改了函数 AHeapBlocks：从结果数组中删除了第一列，因为它没有价值。
- 更改了函数 ReadRegistryValue：参数 4 的类型和含义已更改-请查看新文档。
- 更改了函数 GetServerTime：参数 2 的类型和含义已更改。
- 删 除 了 功 能 ReadInt64AsDouble , ReadPInt64AsDouble , ReadUInt64AsDouble 和 ReadPUInt64AsDouble：请改用 ReadInt64 , ReadPInt64...。
- 删 除 了 功 能 SetSystemTimeEx：改用 SetSystemTime。
- 删 除 了 功 能 SyncToServerTime：改用 SyncToSNTPServer。
- 删 除 了 功 能 MarshalArrayXXX , UnMarshalArrayXXX , MarshalCursorXXX 和 UnMarshalCursorXXX：请改用 MarshalCArray2Cursor , MarshalCArray2FoxArray , MarshalCursor2CArray 或 MarshalFoxArray2CArray。

版本 2.0.0.7-2010-11-16

- 修复了 AAverage 中的错误-数组中的 NULL 值损坏了结果
- 删除了 ASum, AMin, AMax, AAverage 中的限制-函数限制为 65000 个元素, 现在限制为 65000 行
- 现在, 当传递空字符串时, ASplitStr 返回一个带有空字符串的 1 元素数组(行为类似于 ALINES)
- 内部代码清理和优化:
 - 优化的 FoxArray 类-许多函数将速度提高 10-20%, 这些函数将结果返回到数组中-例如 AProcesses, AWindows, AdirEx 等。
 - 通过 LCK 优化的 C ++ API
 - 用 C ++ 强制转换替换了 C 样式强制转换
 - 删除了几乎所有预处理器宏

版本 2.0.0.6-2010-11-08

- 更改了 FCreateEx, FOpenEx 和所有其他 F ... Ex 函数, 以直接使用 Windows api 文件句柄而不是内部伪句柄。
- 如果您已经使用这些函数编写了代码, 它们仍然可以像旧版本一样工作, 则仅有的两个区别是, 由于不再需要以前返回 api 文件句柄的函数 “FHandleEx” , 现在已删除, 而 “AFHandlesEx” 现在返回一个 1 列的 api 句柄数组, 而不是 2 列的内部句柄到 api 句柄的映射数组。此更改的优势在于, 现在可以将任何文件/管道/任何 api 句柄 (不一定使用 FCreateEx 或 FOpenEx 创建) 都可以与 FPutsEx, FWriteEx, FReadEx ... 函数一起使用。
- 更改了 FLL 资源中的版权声明 (AGETFILEVERSION ())
- 现在, CLSIDFromProgID 返回一个二进制字符串 (如使用 CREATEBINARY 创建的字符串), 而不是普通字符串-不应影响已经使用它的任何代码-只是外观, 因为返回的 CLSID 实际上是二进制数据
- 现在, CreateGUID 还返回一个二进制字符串 (仅当将 2 作为参数传递时)
- 修复了 CLSIDFromString 中的错误-函数始终返回空字符串
- 修复了 ARegistryKeys 中的错误-函数引发错误 “访问被拒绝” , 原因是未以足够的权限打开注册表项
- 内部代码清除:
 - 在 LCK 上修改了 C ++ API

- 用 C ++ 强制替换了许多 C 风格的强制转换
- 删除了更多的宏

版本 2.0.0.5-2010-11-05

- 新增功能:

- APaperSizes
- AServiceConfig
- ADependentServices
- WaitForServiceStatus
- AFontInfo
- ARasDevices
- ARasPhonebookEntries
- RasPhonebookDlgEx
- RasDialEx
- RasHangUpEx
- RasGetConnectStatusEx
- RasDialDlgEx
- RasConnectionNotificationEx
- AbortRasConnectionNotificationEx
- RasClearConnectionStatisticsEx
- CreatePublicShadowObjReference
- ReleasePublicShadowObjReference
- GetLocaleInfoEx
- AsyncWaitForObject
- CancelWaitForObject

- 重命名函数:

- "Invoke" to "IDispatch_Invoke"
- "AsyncInvoke" to "IDispatch_AsyncInvoke"

■ "ADialUpConnections" to "ARasConnections"

- 修复了在错误情况下发生的各种功能中的某些内存泄漏
- 内部代码清除-用 C ++ 内联函数替换了许多预处理器 macros
- 删除了 FindFileChange 和 FindRegistryChange 中的硬编码线程限制

版本 2.0.0.3

- 修复了 AbortUrlDownloadToFileEx 函数中的错误,
- 具有进度监视功能的 Internet 资源可中止的异步下载现在可以正常工作, 请参阅 UrlDownloadToFileEx / AbortUrlDownloadToFileEx 的更新示例更新的 Intellisense 表

版本 2.0.0.2

- SQLExecEx 中的固定参数验证-回调功能不可用

版本 2.0.0.1

- 固定的 ClsIdFromProgID-函数返回的是空字符串而不是 CLSID
- 固定的 CreateCallbackFunc-最后一个参数需要数字标志, 但仅接受逻辑值
- 增强的 ADirEx, AFileAttributes (Ex) 和 Get (Set) FileTimes, 现在可以将文件时间检索/设置为 UTC 时间

版本 2.0.0.0

- 修复了 vfp2c.h 中的 BINDSTATUS #DEFINE
- 扩展的 UrlDownloadToFile, 现在可以在其他线程上异步执行该函数
- 添加了功能 AbortUrlDownloadToFileEx 来中止以 UrlDownloadToFileEx 开始的异步下载
- 添加了函数 Decimals, 该函数返回数字值的小数位数
- 添加了功能 MoveFileEx 和 CopyFileEx (通过进度回调移动/复制文件)
- 添加了封送处理功能: ReadPInt64AsDouble, ReadPUInt64AsDouble, WriteInt64, WritePInt64, WriteUInt64, WritePUInt64
- 新增功能 CreateGuid
- 添加了功能 CenterWindowEx
- 增强功能 CreateCallbackFunc-现在支持从其他线程调用的 C 回调函数
- 新增功能 ControlService
- 新增功能 APrintersEx

- 新增功能 IcmpPing
- 用一组新的 C ++ 类重写了 FLL 的大部分内容，这些类包装了 Fox 库构造工具包 (vfp2ccppapi.cpp / h)

版本 1.0.3.5

- 添加了功能 Value2Variant 和 Variant2Value
- 添加了 VFP 之类的扩展文件功能：
FCreateEx, FOpenEx, FCloseEx, FReadEx, FWriteEx, FGetsEx, FPutsEx, FSeekEx, FEoFEx, FChSizeEx, FFlushEx, FLockFile, FUnlockFile, FLockFileEx, FUnlockFileEx, FHandleEx, AFHandlesEx

版本 1.0.3.3

- 增强的 ANetFiles，现在也可以在 Win95 / 98 / Me 上运行
- 增强的 AProcesses，现在也可以在 WinNT 上运行（如果存在 psapi.dll）
- 增强了将文件名作为参数的所有功能：
- GetFileSize, AFileAttributes (Ex), Get (Set) FileAttributes, Get (Set) FileTimes, CompareFileTimes , GetLongPathName , GetShortPathName , GetFileOwner , DeleteFileEx-不再需要指定文件的全路径名了，现在内部是 VFP 函数 FULLPATH 用于检索文件的完整路径。
- 增强的 GetOpenFileName 和 GetSaveFileName：
- 现在，这两个函数可以选择在对话框运行时调用 VFP 回调过程（用于对话框事件，自定义对话框外观）。GetOpenFileName 作为一个额外的参数使用一个存储多个文件选择的数组名（请参见 dialogs.prg 示例）
- 添加了函数 Int64_Add, Int64_Sub, Int64_Mul, Int64_Div, Int64_Mod 以对 Int64 文字进行简单算术
- 添加的功能 AIPAddresses-将本地计算机的 IP 地址存储到阵列中
- 新增功能 ResolveHostToIP- 从您的域或顶级域中获取服务器的 IP 地址，例如 www.google.com
- 添加的函数 GetWindowRectEx-将窗口的屏幕坐标存储到数组中

版本 1.0.3.2

- SQLExecEx 增强/修复：

- 现在默认情况下将 SQL_BIGINT 转换为字符 (C (20)) 列
- 现在， 默认情况下， SQL_GUID 已转换为字符 (C (36)) 列
- 删除 “常规字段” 支持后， 内容已按原样成功存储到常规字段中，但是常规字段在 VFP 中无效。
- 默认情况下， 根据大小， 未知的 SQL 类型现在默认转换为字符/备注字段 (例如， MS SQL Server 中的 xml 数据类型)
- 修复了小数位数比 VFP 支持的精度或小数位数大的错误。
- 添加了标志 SQL_EXCEX_STORE_INFO， 可以将 PRINT 语句或其他命令的输出存储在数组中
- 添加了新的可选参数 nCallbackInterval-指定应调用回调进度函数的记录间隔 (默认为 100)
- 固定的回调功能-将超过 7 个参数传递给 SQL_EXCEX 时出现 “无效参数” 错误
- 修复了一些无效的数据类型转换
- 。空值。值现在已正确检索到备忘/ blob 字段中
- SQL_DEST_VARIABLE 现在支持所有数据类型，但不存储大型类型 (SQL_LONGVARCHAR, SQL_LONGBINARY, SQL_WLONGVARCHAR ..) 。
- 现在默认情况下， SQL_FLOAT, SQL_REAL 和 SQL_DOUBLE 转换为 B (16) 数据类型，而不是 N (20,16) 。
- 现在已正确发送在 SQL 语句中指定为参数的备忘/ blob 字段
- 修复了 SQL_CHAR 或 SQL_BINARY 列大于 254 个字节时的错误，现在已将其转换为备注/ blob 字段，而不会出现错误 “无效字段长度”
- 修复了使用 SQLGetData 存储货币数据类型的错误-出现错误 “数据类型不匹配”
- 修复了解析游标模式的错误，丢失了 N (X, X) 的小数精度，从而导致 N (X, 0)
- 如果数据应该转换为货币，则由于大多数 ODBC 驱动程序不支持以前使用的 SQL_C_NUMERIC 类型，因此现在将其检索为 SQL_CHAR

- 新增功能 ASum, AAverage, AMax 和 Amin

版本 1.0.3.0

- 添加函数 GetWindowsDirectory, GetSystemDirectory 和 ExpandEnvironmentStrings
- 添加功能 GetLongPathName, GetShortPathName
- 添加了功能 RegisterObjectAsFileMoniker

版本 1.0.2.9

- 添加功能 GetWindowTextEx
- 更改了 BindEventsEx 的标志-请参阅 Intellisense 帮助。
- 修复了子类化多个窗口时 UnBindEventsEx 中的错误
- 修复了 BindEventsEx 中的另一个错误，该错误是在您重新绑定以前挂接到另一个对象的消息时发生的，第一个对象已从内存中释放。

版本 1.0.2.8

- 修复了 BINDEVENTSEX 中针对对象方法回调的严重错误

版本 1.0.2.7

- 添加了功能 BindEventsEx 和 UnbindEventsEx
- 新增功能 GetCursorPosEx
- 新增功能 StartService 和 StopService
- 重写 InitVFPC32 ()，如果发生错误，不会停止初始化库的独立部分
- 更正了 vfp2c.h 中的初始化#defineINES

版本 1.0.2.6

- 如果您尝试将 NULL 或空 date (time) s 作为参数传递，则更正了 SQLExecEx 中的错误。
- 向 SQLExecEx 函数添加了标志 SQL_EXCEX_REUSE_CURSOR
- 新增功能 RGB2Colors 和 Colors2RGB

版本 1.0.2.5

- 完成的 SQLExecEx
扩展了 SQLExec，具有存储更新/删除/插入的行，进度回调和自定义 cursorschema 支持的功能。

几年前我们需要吗？:-)

由于此函数相当复杂（本身约有 1300 行代码以及 30 个辅助函数），因此我首先将其放入此开发版本中。

- 修复了 ODBC 错误处理程序中的错误-AErrorEx 返回了错误数量的错误
- 添加的功能 ATimeZones-枚举系统上所有可用的时区
- 添加了功能 DT2UTC 和 UTC2DT，以将日期时间从 UTC / GMT 转换为本地时间，反之亦然
- 添加了功能 SyncToSNTPServer-同步到 Internet 或本地网络上的 SNTP (时间) 服务器 (适用于所有 OS)
- 更改了 GetSystemTime 和 GetServerTime-现在都返回时间，而不是将其存储到通过引用传递的变量中

如果您正在使用这些功能，则必须调整代码-向后兼容性不是我的首要任务。

如果我在功能界面中出错（例如在这种情况下），我将尽快予以纠正。

版本 1.0.2.0

- 固定的 SetFileTimes-完全搞砸了..
- 添加了将数字类型转换为二进制字符串的功能，反之亦然：
- Str2Short, Short2Str, Str2Long, Long2Str, Str2ULong, ULong2Str, Str2Double, Double2Str, Str2Float, Float2Str ..
- 添加了函数 RegisterActiveObject, RevokeActiveObject-将 COM 对象注册到正在运行的对象表中
- 修复了内部帮助器函数 DateTimeToFileTimeEx 中的错误（适用于使用源代码或感兴趣的人）
- 添加的函数 RegistryHiveToObject-在一次调用中使用注册表项的值和子项构建对象层次结构
- 修复了 SHFileOperation 包装器（SHCopyFiles, SHDeleteFiles, SHMoveFiles, SHRenameFiles）中的一个小错误。

如果您将对话框的标题传递给这些函数之一，但未指定 FOF_SIMPLEPROGRESS 标志，则标题将被忽略，

现在，如果您通过标题，则会添加 FOF_SIMPLEPROGRESS 标志

- 添加的功能 ASplitStr-将字符串拆分为 N 个固定长度字符串的数组

版本 1.0.1.0

- 添加了功能 SyncToServerTime 和 RegistryValuesToObject
- 增强了 ADIREX 并修复了其中的一个小错误

开发人员和合作者

- 开发者: Christian Ehlscheid - c.ehlscheid@gmx.de
- 合作者: Eric Selje

VFP2C32Front

● 介绍

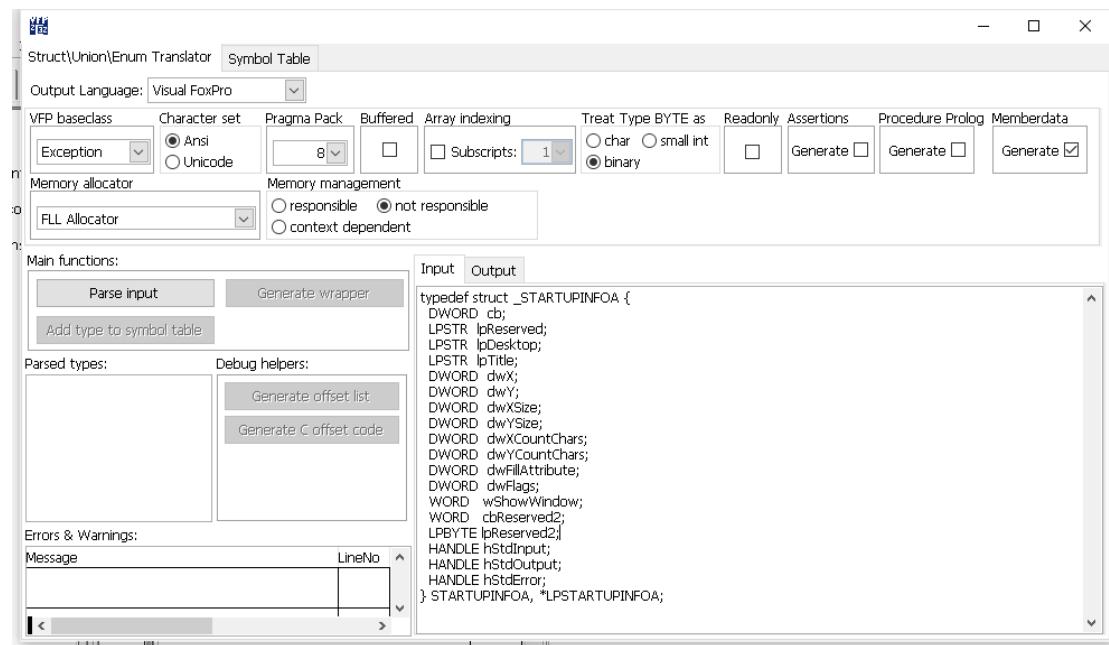
VFP2C32Front 应用程序分析 C 代码段，并创建 VFP 类，以模拟提供的类型定义。

生成的类可以像普通的 VFP 类一样使用，这使您可以以干净的面向对象方式进行编程，并轻松地将 C 移植到 VFP 代码。

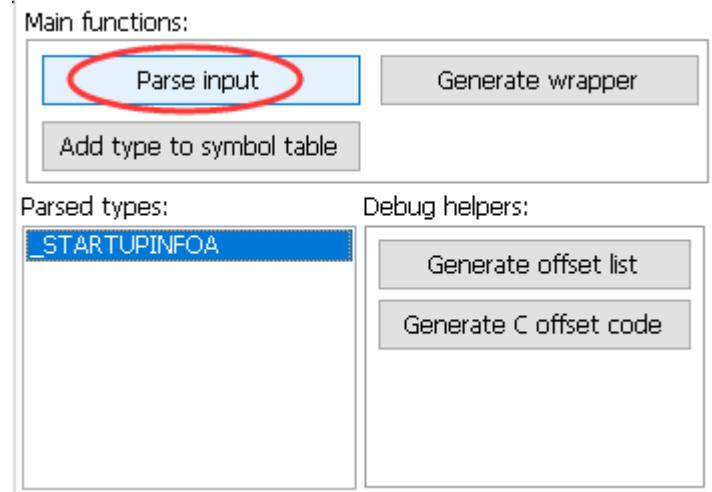
● 一般用法

要将 C 类型转换为 VFP，请将这些类型的源代码复制到“Input”页面的编辑框中。

图 1 显示了具有 [STARTUPINFO](#) 结构的示例。



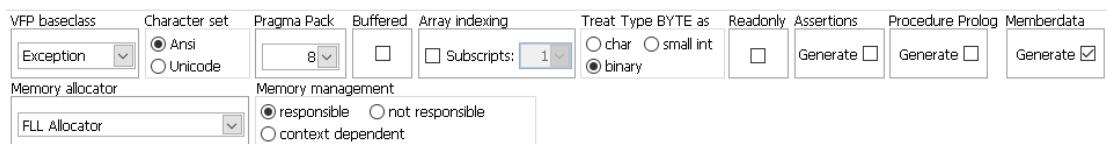
在下一步中，单击“Parse Input”按钮，如果一切顺利，您应该在“解析的类型”列表框中看到解析的类型，如图 2 所示。



现在，您必须配置应如何转换类型。

控制类型转换方式的选项位于顶部，如图 3 所示。

请查看主题翻译选项，以获取有关这些选项如何影响生成的代码的解释。



最后，单击 Generate wrapper 按钮，并在 Output 页面中显示生成的 VFP 代码，如图 4 所示。

```

DEFINE CLASS STARTUPINFOA AS Exception
Address = 0
SizeOf = 68
Name = "STARTUPINFOA"
&& structure fields


```

转换选项

VFP baseclass (VFP 基类)

指定创建的封装类的基类是 Exception 还是 Relation。 VFP 9 中开销最小的类是 Exception。

Character set (字符集设置)

对于许多 Windows api 函数，实际上有 2 种实现，一种使用 **Ansi** 字符集，另一种使用 **Unicode** 字符集。

有几种通用的字符串类型，例如 LPTSTR 映射到 Ansi 或 Unicode 字符串，此设置指定如何解释这些类型。

Pragma pack (对齐方式)

Pragma pack 选项控制 结构 和 unions 的对齐方式。

对于 Windows API 中的 结构 和 unions，很少需要调整此选项。

只需先尝试默认值，如果在将 struct 传递给 C 函数时遇到问题，则可以通过将 **Generate offset list** 的输出与 **Generate C offset code** 生成的已编译 C 程序的输出计算单个成员的正确偏移量。

Buffered (缓冲)

此选项是结构的内存管理的特殊情况。

许多 Windows api 函数期望您传递一个大内存块，然后该函数将一个特定结构的数组存储到其中，包括这些结构引用的所有数据。

如果您打算使用此功能，请选中此选项。例如：[EnumServicesStatusEx](#), [EnumDependentServices](#)。

此选项还可以使结构只读。

Array indexing (数组索引)

此选项添加了特殊代码，使您可以轻松循环遍历结构数组。

如果您设置了 Buffered，则很有可能还需要设置该选项。

Treat type BYTE as (将 BYTE 类型视为)

指定如何在结构中读取和写入 BYTE 类型的任何成员。

ReadOnly (只读)

仅为结构或 union 的成员生成 Access 方法。

Assertions (断点)

如果选中，则在每个 ASSIGN 方法中生成 ASSERT 语句，以验证分配值的数据类型和范围。

Memory allocator (内存分配器)

指定应使用哪些内存分配功能。

FLL Allocator 选项将在生成的代码中使用自定义 FLL 函数的代码：AllocMem, ReAllocMem, FreeMem。

Global Allocator 选项将在生成的代码中使用 FLL 全局函数的代码：AllocHGlobal, ReAllocHGlobal, FreeHGlobal，它们封装了 [GlobalAlloc](#), [GlobalReAlloc](#) 和 [GlobalFree](#) Windows api 函数。

Memory management (内存管理)

此选项指定生成的封装类如何处理内存。

Responsible 在生成的类中生成代码以分配和释放所需的内存。

Not responsible 不产生任何内存管理代码，其用于仅用作其他结构一部分的结构。

Context dependet 为上述两种情况生成代码。此选项的一个主要示例是 [RECT](#) 结构，该结构通常在其他结构中使用，但还需要在许多 api 函数中进行不独立的传递。

常见错误

这是在点击 Parse input 时可能发生的常见错误的列表。

错误	解决方案
----	------

syntax error, unexpected \$end, expecting ';'!	每个类型定义都必须以' ;' 符合结尾。 在行号末尾添加一个' ;' 字符。
Preprocessor token 'SOMENAME' detected. Replace it with its actual value.	解析器无法解析预定义常量。 在 .h 文件中查找定义，并将其替换为实际值。 例如：将 MAX_PATH 替换为 261。
Unknown type specifier 'SOMENAME'!	输入的结构类型不在 Symbol table 中。 你需先在其中进行添加。如果转换包含子结构的结构，则此错误常见。 首先转换子结构，然后将其添加到 Symbol table 中。

示例

GlobalMemoryStatusEx

[GlobalMemoryStatusEx](#) 函数需要一个指向 [MEMORYSTATUSEX](#) 结构的指针。

程序使用默认[转换选项](#)为结构分配和释放所需的内存。

对创建的代码进行的一些修改请参见下面示例中的注释。

```

Local loMemStatus
m.loMemStatus = Createobject('MEMORYSTATUSEX')
? GlobalMemoryStatusEx(m.loMemStatus.Address)
? '内存使用', m.loMemStatus.dwMemoryLoad
? '物理内存', m.loMemStatus.ullTotalPhys

Function GlobalMemoryStatusEx(lpBuffer)
    && BOOL WINAPI GlobalMemoryStatusEx( _inout  LPMEMORYSTATUSEX lpBuffer)
    Declare Integer GlobalMemoryStatusEx In WIN32API Integer
    Return GlobalMemoryStatusEx(m.lpBuffer)
Endfunc

Define Class MEMORYSTATUSEX As Exception
    Address = 0
    SizeOf = 64
    Name = "MEMORYSTATUSEX"
    && 结构字段
    dwLength = .F.
    dwMemoryLoad = .F.
    ullTotalPhys = .F.
    ullAvailPhys = .F.
    ullTotalPageFile = .F.
    ullAvailPageFile = .F.
    ullTotalVirtual = .F.

```

```

ullAvailVirtual = .F.
ullAvailExtendedVirtual = .F.

Procedure Init()
    This.Address = AllocMem(This.SizeOf)
    This.dwLength = This.SizeOf && 手动添加，此成员需要设置为结构的大小。
Endproc

Procedure Destroy()
    FreeMem(This.Address)
Endproc

Procedure dwLength_Access()
    Return ReadUInt(This.Address)
Endproc

Procedure dwLength_Assign(InNewVal)
    WriteUInt(This.Address, InNewVal)
Endproc

Procedure dwMemoryLoad_Access()
    Return ReadUInt(This.Address+4)
Endproc

Procedure ullTotalPhys_Access()
    Return ReadUInt64(This.Address+8, 4) && 将返回值更改为数值
Endproc

Procedure ullAvailPhys_Access()
    Return ReadUInt64(This.Address+16, 4)
Endproc

Procedure ullTotalPageFile_Access()
    Return ReadUInt64(This.Address+24, 4)
Endproc

Procedure ullAvailPageFile_Access()
    Return ReadUInt64(This.Address+32, 4)
Endproc

Procedure ullTotalVirtual_Access()
    Return ReadUInt64(This.Address+40, 4)
Endproc

```

```

Procedure ullAvailVirtual_Access()
    Return ReadUInt64(This.Address+48, 4)
Endproc

Procedure ullAvailExtendedVirtual_Access()
    Return ReadUInt64(This.Address+56, 4)
Endproc
Enddefine

```

RegisterPowerSettingNotification

以下示例监视系统的电源状态更改。

使用 [RegisterPowerSettingNotification](#) 函数，我们注册了一个窗口句柄，当系统的电源状态更改时，该窗口句柄将接收通知。

然后，我们使用 [BindEventsEx](#) 函数绑定窗口过程。

[POWERBROADCAST_SETTING](#) 结构的长度是可变的，指向该结构的指针将传递到已注册的回调函数 [PowerBroadCastEvent](#)。

该结构的转换选项是 **Memory management-not responsible**，因为该结构是从 OS 传递给我们的，因此我们不需要分配或释放任何内存。

手动添加了 **DataDword** 和 **DataGuid** 成员以访问结构中的各种数据。

```

#define WM_POWERBROADCAST          0x0218
#define PBT_POWERSETTINGCHANGE     0x8013

#define GUID_POWERSCHEME_PERSONALITY      0h41855D2443392244B02513A784F679B7
                                         && {245D8541-3943-4422-B025-13A784F679B7}
#define GUID_MIN_POWER_SAVINGS          0hDA7F5E8CBFE8964A9A85A6E23A8C635C &&
                                         {8C5E7FDA-E8BF-4A96-9A85-A6E23A8C635C}
#define GUID_MAX_POWER_SAVINGS          0h081384A14135AB4FBC81F71556F20B4A &&
                                         {A1841308-3541-4FAB-BC81-F71556F20B4A}
#define GUID_TYPICAL_POWER_SAVINGS       0h22421B3894F6F0419685FF5BB260DF2E &&
                                         {381B4222-F694-41F0-9685-FF5BB260DF2E}
#define GUID_ACDC_POWER_SOURCE          0h599A3E5DD5E9004BA6BDFF34FF516548 &&
                                         {5D3E9A59-E9D5-4B00-A6BD-FF34FF516548}
#define GUID_BATTERY_PERCENTAGE_REMAINING
                                         0h4180ADA75AB4AE4C87A3EECBB468A9E1 && {A7AD8041-B45A-4CAE-87A3-
                                         EECBB468A9E1}
#define GUID_IDLE_BACKGROUND_TASK         0hD8315C5134F73D16A0FD11A08C91E8F1 &&
                                         {515C31D8-F734-163D-A0FD-11A08C91E8F1}
#define GUID_SYSTEM_AWAYMODE            0h80F5A798F701AA489C0F44352C29E5C0 &&

```

```

{98A7F580-01F7-48AA-9C0F-44352C29E5C0}
#define GUID_MONITOR_POWER_ON          0h151073021045264599E6E5A17EBD1AEA &&
{02731015-4510-4526-99E6-E5A17EBD1AEA}

Public goPowerEvents
m.goPowerEvents = Createobject('cPowerEvents')
m.goPowerEvents.RegisterEvent(GUID_POWERSCHEME_PERSONALITY)
m.goPowerEvents.RegisterEvent(GUID_ACDC_POWER_SOURCE)
m.goPowerEvents.RegisterEvent(GUID_BATTERY_PERCENTAGE_REMAINING)
m.goPowerEvents.RegisterEvent(GUID_MONITOR_POWER_ON)

Define Class cPowerEvents As Custom
Protected oRegisteredEvents
oRegisteredEvents = .Null.

Function Init
This.oRegisteredEvents = Createobject('Collection')
BINDEVENTSEX(_vfp.HWnd, WM_POWERBROADCAST, This, 'PowerBroadCastEvent',
'wParam, lParam')
Endfunc

Function Destroy
This.UnregisterEvent()
UNBINDEVENTSEX(_vfp.HWnd, WM_POWERBROADCAST)
Endfunc

Function RegisterEvent(lcEvent)
Local lnIndex, lnHandle
m.lnIndex = This.oRegisteredEvents.GetKey(m.lcEvent)
If m.lnIndex = 0
    m.lnHandle = RegisterPowerSettingNotification(_vfp.HWnd, m.lcEvent, 0)
    This.oRegisteredEvents.Add(m.lnHandle, m.lcEvent)
Endif
Endfunc

Function UnregisterEvent(lcEvent)
Local lnIndex, lnHandle
If Vartype(m.lcEvent) = 'C'
    m.lnIndex = This.oRegisteredEvents.GetKey(m.lcEvent)
    If m.lnIndex != 0
        m.lnHandle = This.oRegisteredEvents.Item(m.lnIndex)
        UnregisterPowerSettingNotification(m.lnHandle)
        This.oRegisteredEvents.Remove(m.lnIndex)
    Endif
Endfunc

```

```

Else && 移除所有事件
For Each m.InHandle In This.oRegisteredEvents
    UnregisterPowerSettingNotification(m.InHandle)
Endfor
This.oRegisteredEvents.Remove(-1)
Endif
Endfunc

Function PowerBroadCastEvent
Lparameters dwEvent, InData

Do Case

Case m.dwEvent = PBT_POWERSETTINGCHANGE

Local loPBS, lcPowerSetting
m.loPBS = Createobject('POWERBROADCAST_SETTING', m.InData) && pass
pointer to structure!
    m.lcPowerSetting = m.loPBS.PowerSetting.Guid
Do Case
    Case m.lcPowerSetting == GUID_POWERSHEME_PERSONALITY
        Local lcPSGuid, lcPSName
        m.lcPSGuid = m.loPBS.DataGuid.Guid
        m.lcPSName = Icase(m.lcPSGuid == GUID_MIN_POWER_SAVINGS,
'高性能';
        m.lcPSGuid == GUID_MAX_POWER_SAVINGS, '省电', '平衡')
        ? '活动电源方案更改为: ' + m.lcPSName

    Case m.lcPowerSetting == GUID_ACDC_POWER_SOURCE
        Local InPowerSource
        m.InPowerSource = m.loPBS.DataDword
Do Case
    Case m.InPowerSource = 0
        ? '计算机由交流电源供电。'
    Case m.InPowerSource = 1
        ? '计算机由电池电源供电。'
    Case m.InPowerSource = 2
        ? '计算机由短期电源 (如 UPS 设备) 供电。'
Endcase

Case m.lcPowerSetting == GUID_BATTERY_PERCENTAGE_REMAINING
    ? '当前电池容量: ' + Alltrim(Str(m.loPBS.DataDword)) + '%'

Case m.lcPowerSetting == GUID_MONITOR_POWER_ON

```

```

? '监视器状态更改为: ' + Iif(m.loPBS.DataDword = 0, '关', '开')
Endcase
Endcase
Endfunc
Enddefine

Function RegisterPowerSettingNotification(hRecipient, PowerSettingGuid, Flags)
    Declare Integer RegisterPowerSettingNotification In WIN32API Integer hRecipient, String
PowerSettingGuid, Integer Flags
    Return RegisterPowerSettingNotification(m.hRecipient, m.PowerSettingGuid, m.Flags)
Endfunc

Function UnregisterPowerSettingNotification(Handle)
    Declare Integer UnregisterPowerSettingNotification In WIN32API Integer Handle
    Return UnregisterPowerSettingNotification(m.Handle)
Endfunc

Define Class POWERBROADCAST_SETTING As Exception
    Address = 0
    Name = "POWERBROADCAST_SETTING"
    && structure fields
    PowerSetting = .Null.
    DataLength = .F.
    DataGuid = .F. && 手动添加以在数据成员的偏移量处访问GUID
    DataDword = .F. && 手动添加以在Data成员的偏移量处访问DWORD

    Procedure Init(InAddress)
        Assert Type('InAddress') = 'N' And InAddress != 0 Message 'Invalid structure address!'
        This.Address = InAddress
        This.PowerSetting = Createobject('GUID', This.Address)
        This.DataGuid = Createobject('GUID', This.Address+20) && 手动添加
    Endproc

    Procedure Address_Assign(InAddress)
        Do Case
            Case This.Address = 0
                This.Address = InAddress
            Case This.Address = InAddress
            Otherwise
                This.Address = InAddress
                This.PowerSetting.Address = InAddress
                This.DataGuid.Address = InAddress + 20 && 手动添加
        Endcase
    Endproc

```

```

Procedure DataLength_Access()
    Return ReadUInt(This.Address+16)
Endproc

Procedure DataDword_Access()
    Return ReadUInt(This.Address+20)
Endproc
Enddefine

Define Class GUID As Exception
    Address = 0
    SizeOf = 16
    Protected Embedded
    Embedded = .F.
    Name = "GUID"
    && structure fields
    Data1 = .F.
    Data2 = .F.
    Data3 = .F.
    Data4 = .F.
    Guid = .F. && custom member to assign and access GUID as a binary string
    GuidString = .F. && custom member to assign and access GUID as a readable string

Procedure Init(InAddress)
    If Pcount() = 0
        This.Address = AllocMem(This.SizeOf)
    Else
        This.Address = InAddress
        This.Embedded = .T.
    Endif
Endproc

Procedure Destroy()
    If !This.Embedded
        FreeMem(This.Address)
    Endif
Endproc

Procedure Data1_Access()
    Return ReadUInt(This.Address)
Endproc

Procedure Data1_Assign(InNewVal)

```

```

    WriteUInt(This.Address,InNewVal)
Endproc

Procedure Data2_Access()
    Return ReadUShort(This.Address+4)
Endproc

Procedure Data2_Assign(InNewVal)
    WriteUShort(This.Address+4,InNewVal)
Endproc

Procedure Data3_Access()
    Return ReadUShort(This.Address+6)
Endproc

Procedure Data3_Assign(InNewVal)
    WriteUShort(This.Address+6,InNewVal)
Endproc

Procedure Data4_Access()
    Return ReadCharArray(This.Address+8,8)
Endproc

Procedure Data4_Assign(InNewVal)
    WriteCharArray(This.Address+8,InNewVal,8)
Endproc

Procedure Guid_Access()
    Return ReadBytes(This.Address, 16)
Endproc

Procedure Guid_Assign(InNewVal)
    WriteBytes(This.Address, m.InNewVal, 16)
Endproc

Procedure GuidString_Access()
    Return STRINGFROMCLSID(This.Address)
Endproc

Procedure GuidString_Assign(InNewVal)
    Local IcGuid
    m.IcGuid = CLSIDFROMSTRING(m.InNewVal)
    WriteBytes(This.Address, m.IcGuid)
Endproc

```

Enddefine

参考

按字母排序

A

AAverage	计算数组中值的算术平均值
AbortRasConnectionNotification	终止监视由 RasConnectionNotificationEx 启动的 RAS 连接的线程。
AbortUrlDownloadToFileEx	中止以 UrlDownloadToFileEx 开始的异步下载。
ADependentServices	将所有依赖于传入服务的服务存储到一个数组中。
ADesktopArea	将可见桌面 (不包括系统任务栏或任何应用程序桌面工具栏) 区域的尺寸存储到数组中。
ADesktops	将与调用过程的指定窗口站关联的所有桌面存储到一个数组中。
ADirectoryInfo	将有关目录的信息存储到数组中。
ADirEx	ADIR 函数的扩展。将有关文件的信息存储到数组，游标或为每个文件调用回调函数。
ADisplayDevices	将有关当前会话中的显示设备的信息存储到数组中。
ADriveInfo	将有关当前可用磁盘驱动器的信息存储到数组中。
AErrorEx	将有关库中最后发生的错误信息存储到数组中。
AFHandlesEx	将使用 FCreateEx 或 FOpenEx 创建的所有打开文件句柄存储到数组中。
AFileAttributes	将指定文件或目录的属性存储到数组中。
AFileAttributesEx	将指定文件或目录的扩展属性存储到数组中。
AFontInfo	从 Ture Type 字体文件中检索有关包含字体的信息到对象中。
AHeapBlocks	将有关进程已分配的堆块的信息存储到数组中。
AlpAddresses	将机器的所有 IP 地址存储到一个数组中。
AllocHGlobal	从全局堆分配指定数量的字节。
AllocMem	从自定义库堆分配指定数量的字节。
AllocMemTo	从自定义库堆中分配指定数量的字节，并在传递的地址处存储指向分配的内存的指针。
AMax	获取数组中的最大值

AMemBlocks	将有关从库内部堆分配的所有块的信息存储到数组中。
AMin	获取数组中的最小值
ANetFiles	将有关服务器上已打开文件的信息存储到数组中。
ANetServers	将在域中可见的所有指定类型的服务器存储到数组中。
APaperSizes	将有关支持的纸张尺寸的信息存储到数组中。
APrinterForms	将有关指定打印机支持的窗体的信息存储到数组中。
APrintersEx	将有关可用打印机, 打印服务器, 域或打印提供程序的信息存储到数组中。
APrinterTrays	将有关打印机可用纸盒的信息存储到数组中。
APrintJobs	将有关指定打印机的一组指定打印作业的信息存储到数组中。
AProcesses	将有关系统当前正在运行的进程的信息存储到数组中。
AProcessHeaps	将有关进程堆的信息存储到数组中。
AProcessModules	将有关进程加载的模块 (DLL) 的信息存储到数组中。
AProcessThreads	将有关进程的线程信息存储到数组中。
ARasConnections	将有关所有活动 RAS 连接的信息存储到数组中。
ARasDevices	将有关所有可用的支持 RAS 的设备的信息存储到数组中。
ARasPhonebookEntries	将有关远程访问电话簿中所有条目名称的信息存储到数组中。
ARegistryKeys	将有关指定注册表项的所有子项的信息存储到数组中。
ARegistryValues	将有关指定注册表项的所有值的信息存储到数组中。
AResolutions	将有关显示设备的所有图形模式 (分辨率) 的信息存储到数组中。
AResourceLanguages	将与二进制模块关联的指定类型和名称的特定于语言的资源的信息存储到数组中。
AResourceNames	将有关二进制模块中指定类型的资源的信息存储到数组中。
AResourceTypes	将有关二进制模块内资源类型的信息存储到数组中。
AServiceConfig	将指定的 Windows 服务的配置参数存储到数组中。
AServices	将有关 Windows 服务的信息存储到数组中。
AServiceStatus	将有关指定服务的当前状态的信息存储到数组中。
ASplitStr	根据提供的长度将字符串拆分为数组。
ASQLDatasources	将有关已配置的 ODBC 数据源的信息存储到数组中。
ASQLDrivers	将有关已安装的 ODBC 驱动程序的信息存储到数组中。
ASum	对数组中的值求和
AsyncWaitForObject	在单独的线程上等待 API 句柄而不阻塞, 当句柄发出信号时, 将执行提供的回调函数
ATimeZones	获取有关时区的信息。
AVolumeInformation	将有关与指定根目录关联的文件系统和卷的信息存储到阵列

	中。
AVolumeMountPoints	将指定卷上已安装文件夹的名称存储到阵列中。
AVolumePaths	将指定卷的驱动器号和卷 GUID 路径存储到阵列中。
AVolumes	将计算机上的卷名称存储到阵列中。
AWindowProps	将有关窗口的属性列表中条目的信息存储到数组中。
AWindows	将窗口句柄 (HWND) 存储到数组中。
AWindowsEx	将有关窗口的信息存储到数组中。
AWindowStations	将当前会话中所有窗口站的名称存储到一个数组中。

B

BindEventsEx	提供在 API 窗口收到指定的窗口消息时执行函数或对象方法的功能。
------------------------------	-----------------------------------

C

CancelFileChange	停止监视指定目录的文件更改的线程。
CancelRegistryChange	停止监视指定注册表项更改的线程。
CancelWaitForObject	停止等待句柄发出信号的线程
CenterWindowEx	在特定窗口，父窗口或桌面上将窗口居中。
ChangeSQLDataSource	修改 ODBC 数据源。
CloseRegistryKey	关闭提供的注册表项句柄 (HKEY)。
CloseServiceHandle	关闭提供的服务句柄 (SC_HANDLE)。
CLSIDFromProgID	返回给定 ProgID (COM 类名) 的二进制 CLSID。 例如：“VisualFoxPro.Application”。
CLSIDFromString	将可识别的 CLSID 转换为二进制 CLSID。这是 StringFromCLSID 函数的反函数。
Colors2RGB	将提供的红色，绿色和蓝色值转换为 32 位整数。
CompactMem	返回库特定堆中最大已提交自由块的大小。
CompareFileTimes	比较两个文件的最后写入时间。
ContinueService	将继续请求发送到指定的 Windows 服务。
ControlService	将自定义控制请求发送到指定的 Windows 服务。
CopyFileEx	复制文件时，可以传递一个可选的回调函数，该函数在复制操作进行时会接收状态。
CreateCallbackFunc	创建一个模拟 C 回调函数的程序集 thunk
CreateGuid	创建一个新的 GUID (全局唯一标识符)。

CreatePublicShadowObjReference	创建一个新的公共变量，引用提供的对象，而不增加对象引用计数。
CreateRegistryKey	创建或打开注册表项。
CreateService	安装 Windows 服务。
CreateSQLDataSource	创建 ODBC 数据源
CreateThreadObject	在单独的线程上创建一个 COM 对象。

D

Decimals	检索数字的小数位数。
DeleteDirectory	删除包括所有文件和子目录的目录。
DeleteFileEx	删除文件。
DeleteRegistryKey	删除指定的注册表项。
DeleteSQLDataSource	删除 ODBC 数据源
DestroyCallbackFunc	释放传入的 C 回调函数
Double2DT	将双精度（数字）值转换为日期时间。
Double2Str	将双精度（64 位数值）值转换为二进制字符串。
DT2Double	将日期时间值转换为双精度（数字）值。
DT2FT	将日期时间值转换为 FILETIME 结构。
DT2ST	将日期时间值转换为 SYSTEMTIME 结构。
DT2Timet	将日期时间值转换为 Time_t (Unix) 时间戳。
DT2UTC	将当前活动时区的日期时间转换为 UTC 日期时间。

E

ExpandEnvironmentStrings	在传入的字符串中扩展环境变量。
--	-----------------

F

FChSizeEx	更改使用 FOpenEx 或 FCreateEx 函数打开的文件的大小。
FCloseEx	刷新并关闭使用 FCreateEx 或 FOpenEx 函数打开的文件或通信端口。
FCreateEx	创建并打开一个文件。
FEoFEx	确定文件指针是否位于文件末尾。
FFlushEx	将使用 FCreateEx 或 FOpenEx 打开的文件刷新到磁盘。

FGetsEx	从指定的文件或使用 FOpenEx 或 FCreateEx 打开的通信端口返回一系列字节，直到遇到回车符为止。
FindFileChange	监视目录以在单独的线程中更改文件。
FindRegistryChange	监视注册表项中单独线程中的更改。
Float2Str	将浮点（32 位数值）值转换为二进制字符串。
FLockFile	在使用 FCreateEx 或 FOpenEx 打开的文件中锁定字节区域。
FLockFileEx	在使用 FCreateEx 或 FOpenEx 打开的文件中锁定字节区域。
FOpenEx	打开文件。
FormatMessageEx	检索指定错误号的错误消息。
FPutsEx	将字符串、回车符和换行符写入使用 FCreateEx 或 FOpenEx 打开的文件中。
FReadEx	从使用 FCreateEx 或 FOpenEx 打开的文件中返回指定的字节数。
FreeHGlobal	释放分配给 AllocHGlobal 的内存。
FreeMem	释放由 AllocMem 或 ReAllocMem 函数从库内部堆分配的内存块。
FreePMem	从库内部堆释放传入的指针指向的内存块。
FreeRefArray	释放分配给以 C 样式数组传递的所有内存。
FSeekEx	在使用 FCreateEx 或 FOpenEx 打开的文件中移动文件指针。
FT2DT	将 FILETIME 结构转换为日期时间值。
FUnlockFile	解锁使用 FCreateEx 或 FOpenEx 打开的文件中的字节区域。
FUnlockFileEx	解锁使用 FCreateEx 或 FOpenEx 打开的文件中的字节区域。
FWriteEx	将字符串写入使用 FCreateEx 或 FOpenEx 打开的文件中。

G

GetCursorPosEx	检索鼠标光标的位置。
GetFileAttributes	获取文件或目录的属性。
GetFileOwner	获取文件的所有者。
GetFileSize	获取文件的大小。
GetFileTimes	获取文件的创建时间、最后访问时间和最后写入时间。
GetIUnknown	返回指向 COM 对象的 IUnknown 接口的指针。
GetLocaleInfoEx	检索有关语言环境的信息。
GetLongPathName	将指定的路径转换为其长格式。
GetOpenFileName	创建一个“打开”对话框，该对话框允许用户指定驱动器、目录以及要打开的文件（集）的名称。

GetSaveFileName	创建一个“保存”对话框，该对话框允许用户指定要保存的文件的驱动器，目录和名称。
GetServerTime	从 Windows 服务器检索当前时间。
GetShortPathName	获取指定路径的短路径形式。
GetSystemDirectory	检索 Windows 系统目录的路径。
GetSystemTime	以协调世界时 (UTC) 检索当前系统日期和时间或当地时间。
GetWindowRectEx	检索指定窗口的边界矩形的尺寸。
GetWindowsDirectory	检索 Windows 目录的路径。
GetWindowTextEx	检索与窗口相对应的文本。

IcmpPing	发送 IPv4 ICMP 回显请求（也称为 ping），并返回所有回显响应答复。
InitVFP2C32	初始化库。
Int64_Add	64-bit integers 相加。
Int64_Div	除以 64-bit integers。
Int64_Mod	将一个 64-bit integer 除以另一个，然后返回余数。
Int64_Mul	乘以 64-bit integers.
Int64_Sub	减去 64-bit integers.
Int64ToStr	将_int64 (64 位数值) 值转换为请求的格式。
Ip2MacAddress	返回给定 IP 地址的 MAC 地址。
IsEqualGuid	比较两个 GUID 是否相等。

LockHGlobal	锁定全局内存对象，并返回一个指向该对象内存块第一个字节的指针。
Long2Str	将带符号的 long (32 位数值) 值转换为二进制字符串。

M

MarshalCArray2Cursor	将 C 数组转换为 VFP Cursor
MarshalCArray2FoxArray	将 C 数组转换为 VFP 数组
MarshalCursor2CArray	将 VFP Cursor 转换为 C 数组
MarshalFoxArray2CArray	将 VFP 数组转换为 C 数组

MessageBoxEx	创建、显示和操作消息框。 消息框包含应用程序定义的消息文本和标题，任何图标以及预定义按钮的任意组合。
MoveFileEx	移动文件时，可以传递可选的回调函数，该函数在进行移动操作时接收状态。

N

Num2Binary	将 32 位整数转换为可识别的二进制字符串。
----------------------------	------------------------

O

OpenRegistryKey	打开指定的注册表项。
OpenService	打开现有服务。
OsEx	检索当前操作系统。

P

PauseService	将暂停请求发送到指定的服务。
PG_BytA2Str	将 PostgreSQL 的 ByteA 值转换为字符串。
PG_Str2ByteA	将字符串转换为转义的 PostgreSQL ByteA 值。
ProgIDFromCLSID	返回给定 CLSID 的 ProgID。

R

RasClearConnectionStatistics	RasClearConnectionStatistics 函数清除指定 RAS 连接的所有累积统计信息。
RasConnectionNotificationEx	每当创建或终止连接时，都会启动一个新线程来监视系统的 RAS 连接，该线程将调用指定的回调过程。
RasDialDlgEx	使用指定的电话簿条目和登录用户的凭据建立 RAS 连接。
RasDialEx	RasDial 函数在 RAS 客户端和 RAS 服务器之间建立 RAS 连接。
RasGetConnectStatusEx	将有关指定远程访问连接的当前状态的信息检索到数组中。
RasHangUpEx	终止远程访问连接。
RasPhonebookDlgEx	显示主“拨号网络”对话框。 在此模式对话框中，用户可以拨打，编辑或删除所选的电话簿条目，创建新的电话簿条目或指定用户首选项。 对话框关闭时，该函数返回。

ReadBytes	从指定地址返回一个字节范围。
ReadChar	从指定地址返回一个 C 字符 (单个字符)。
ReadCharArray	从 C 样式字符数组中检索字符串。
ReadCString	从指定地址返回一个 C 字符串。
ReadDouble	从指定地址返回一个双精度 (64-bit floating point value)
ReadFloat	从指定地址返回一个浮点数 (32-bit floating point value)
ReadInt	从指定地址返回一个 32 位整数 (32-bit integer)
ReadInt64	从指定地址返回一个 64 位带符号整数(64-bit signed integer)
ReadInt8	从指定地址返回一个 8 位整数 (8-bit integer)
ReadLogical	从指定地址返回一个逻辑值
ReadPChar	从指定的间接地址返回 C 字符 (单个字符)。
ReadPCString	从指定的间接地址返回 C 字符串。
ReadPDouble	从指定的间接地址返回一个双精度 (64-bit floating point value)
ReadPFloat	从指定的间接地址返回一个浮点数 (32-bit floating point value)
ReadPlnt	从指定的间接地址返回一个 32 位整数 (32-bit integer)
ReadPlnt64	从指定的间接地址返回一个 64 位带符号整数 (64-bit signed integer)
ReadPlnt8	从指定的间接地址返回一个 8 位整数 (8-bit integer)
ReadPLogical	从指定的间接地址返回一个逻辑值
ReadPointer	从指定地址返回指针。
ReadPPointer	从指定的间接地址返回指针。
ReadProcessMemoryEx	从另一个进程的内存空间中返回一定范围的字节。
ReadPShort	从指定的间接地址中返回一个 16 位整数 (16-bit integer)
ReadPUInt	从指定的间接地址中返回 32 位无符号整数 (32-bit unsigned integer)
ReadPUInt64	从指定的间接地址中返回 64 位无符号整数 (64-bit unsigned integer)
ReadPUInt8	从指定的间接地址中返回一个 8 位无符号整数(8-bit unsigned integer)
ReadPUSHort	从指定的间接地址中返回一个 16 位无符号整数 (16-bit unsigned integer)
ReadPWString	从指定的间接地址返回转换为 Ansi 的 Unicode 字符串。
ReadRegistryKey	检索指定注册表值的数据。
ReadShort	从指定地址返回一个 16 位整数 (16-bit integer)

ReadUInt	从指定的地址返回一个 32 位无符号整数 (32-bit unsigned integer)
ReadUInt64	从指定地址返回一个 64 位无符号整数 (64-bit unsigned integer)
ReadUInt8	从指定地址返回一个 8 位无符号整数(8-bit unsigned integer)
ReadUShort	从指定地址返回一个 16 位无符号整数 (16-bit unsigned integer)
ReadWCharArray	从 C 风格的 unicode 字符数组中返回一个字符串。
ReadWString	从指定地址返回转换为 Ansi 的 Unicode 字符串。
ReAllocHGlobal	更改指定的全局内存对象的大小。
ReAllocMem	更改以前由 AllocMem 分配的内存块的大小。
RegisterActiveObject	将传递的对象注册为 cProgID 中指定的类的活动对象。 注册会导致该对象在 OLE 的运行对象表 (ROT) 中列出，该对象是全局可访问的查找表，用于跟踪计算机上当前正在运行的对象。
RegisterObjectAsFileMoniker	为传递的对象创建文件名字对象。
RegistryHiveToObject	将包括所有子项的注册表项存储到对象中。
RegistryValuesToObject	将注册表项的所有值存储到一个对象中。
ReleasePublicShadowObjReference	释放引用使用 CreatePublicShadowObjReference 创建的对象的公共变量。
ResolveHostTolp	解析指定主机名的 IP 地址。
RevokeActiveObject	从运行对象表 (ROT) 中注销对象。
RGB2Colors	将 RGB 拆分为其组成部分。

S

SetFileAttributes	设置文件或目录的属性。
SetFileTimes	设置创建、上次访问或上次修改指定文件或目录的日期和时间。
SetSystemTime	设置当前系统时间和日期。
SHBrowseFolder	显示一个对话框，允许用户选择一个 Shell 文件夹。
SHCopyFiles	复制一个或几个文件。
SHDeleteFiles	删除一个或几个文件。
SHMoveFiles	移动一个或几个文件。
Short2Str	将带符号的 short (16 位数值) 值转换为二进制字符串。
SHRenameFiles	重命名文件。
SHSpecialFolder	检索特殊文件夹的路径。

SizeOfMem	检索由 AllocMem 分配的内存块的大小。
SQLCancelEx	取消准备好的 SQL 语句。
SQLExecEx	扩展的 SQLEXEC, 将 SQL 语句发送到数据源, 在该数据源中处理该语句。
SQLGetPropEx	扩展的 SQLGETPROP, 检索 ODBC 连接属性。
SQLPrepareEx	扩展的 SQLPREPARE。 准备一个 SQL 语句供 SQLExecEx () 远程执行。
SQLSetPropEx	扩展的 SQLSETPROP, 设置 ODBC 连接属性。
ST2DT	将 SYSTEMTIME 结构转换为日期时间值。
StartService	启动 Windows 服务。
StopService	将停止请求发送到指定的服务。
Str2Double	将二进制字符串转换为双精度 (64 位数值)。
Str2Float	将二进制字符串转换为浮点数 (32 位数字值)。
Str2Int64	将二进制字符串转换为 <code>_int64</code> (64 位数值)。
Str2Long	将二进制字符串转换为带符号的 <code>long</code> (32 位数字值)。
Str2Short	将二进制字符串转换为带符号的 <code>short</code> (16 位数值)。
Str2UInt64	将二进制字符串转换为无符号 <code>_int64</code> (64 位数值)。
Str2ULong	将二进制字符串转换为 <code>unsigned long</code> (32 位数值)。
Str2UShort	将二进制字符串转换为无符号 <code>short</code> (16 位数值)。
StringFromCLSID	将全局唯一标识符 (GUID) 转换为可识别字符的字符串。
SyncToSNTPServer	与 SNTP 服务器同步系统时间和日期。

T

Timet2DT	将 <code>Time_t</code> (Unix) 时间戳转换为日期时间值。
--------------------------	---

U

UInt64ToStr	将无符号的 <code>_int64</code> (64 位数值) 值转换为请求的格式。
ULong2Str	将无符号 <code>long</code> (32 位数字值) 值转换为二进制字符串。
UnbindEventsEx	取消绑定先前与 <code>BindEventsEx</code> 绑定的窗口的事件。
UnlockHGlobal	递减与用 AllocHGlobal 分配一个内存对象关联的锁计数。
UrlDownloadToFileEx	从 Internet 下载资源并将其保存到文件中。
UShort2Str	将无符号的 <code>short</code> (16 位数值) 值转换为二进制字符串。
UTC2DT	将日期时间值从 UTC 转换为本地时区。

V

ValidateMem	验证库内部堆。 该函数扫描堆中的所有内存块，并验证堆管理器维护的堆控制结构是否处于一致状态。 您也可以使用 ValidateMem 函数来验证单个内存块，而无需检查整个堆的有效性。
Value2Variant	将任何类型的变量或字段转换为二进制字符串。
Variant2Value	将 Value2Variant 创建的二进制字符串转换为原始值。
VFP2CSys	提供对库内部资源的访问或更改全局库行为。

W

WaitForServiceStatus	监视 Windows 服务以达到指定状态。
WriteBytes	将二进制数据写入指定的地址
WriteChar	在指定地址写入一个字符
WriteCharArray	在指定地址写入一个字符串
WriteCString	分配或重新分配 C 样式字符串
WriteDouble	在指定地址写入一个双精度值(64-bit floating point)
WriteFloat	在指定地址写入一个浮点数(32-bit floating point)
WriteGPCString	为 C 风格的字符串分配或重新分配内存，并在指定地址将指针写入该字符串
WriteInt	在指定地址写入 32 位整数 (32-bit integer)
WriteInt64	在指定地址写入一个 64 位带符号整数(64-bit signed integer)
WriteInt8	在指定地址写入 8 位整数(8-bit integer)
WriteLogical	将逻辑值写入指定的地址
WritePChar	在指定的间接地址处写入单个字符
WritePCString	为 C 风格的字符串分配或重新分配内存，并在指定地址将指针写入该字符串。
WritePDouble	在指定的间接地址处写入一个双精度(64-bit floating point)
WritePFloat	在指定的间接地址处写入浮点数(32-bit floating point)
WritePlnt	在指定的间接地址处写入 32 位整数(32-bit integer)
WritePlnt64	在指定的间接地址处写入 64 位带符号整数(64-bit signed integer)
WritePlnt8	在指定的间接地址处写入一个 8 位整数(8-bit integer)
WritePLogical	在指定的间接地址处写入逻辑值
WritePointer	将指针写入指定的地址

WritePPointer	将指针写入指定的间接地址
WritePShort	在指定的间接地址处写入 16 位整数(16-bit integer)
WritePUInt	在指定的间接地址处写入 32 位无符号整数(32-bit unsigned integer)
WritePUInt64	在指定的间接地址处写入 64 位无符号整数(64-bit unsigned integer)
WritePUInt8	在指定的间接地址写入 8 位无符号整数(8-bit unsigned integer)
WriteP USHORT	在指定的间接地址处写入 16 位无符号整数 (16-bit unsigned integer)
WritePWChar	在指定的间接地址处写入单个 Unicode 字符
WritePWString	为 C 风格的 unicode 字符串分配或重新分配内存，并在指定地址将指针写入该字符串。
WriteRegistryKey	在注册表项下设置数据和指定值的类型。
WriteShort	在指定的地址写入 16 位整数 (16-bit integer)
WriteUInt	在指定地址写入 32 位无符号整数 (32-bit unsigned integer)
WriteUInt64	在指定地址写入一个 64 位无符号整数 (64-bit unsigned integer)
WriteUInt8	在指定地址写入一个 8 位无符号整数(8-bit unsigned integer)
WriteUShort	在指定地址写入一个 16 位无符号整数 (16-bit unsigned integer)
WriteWChar	在指定地址写入一个 Unicode 字符
WriteWCharArray	在指定地址写入 Unicode 字符串
WriteWString	分配或重新分配 C 样式 Unicode 字符串

按功能

数组	用于 VFP 数组的函数
C 数组仿真	VFP Cursor 和 数组与 C 数组相互转换的函数
C 回调和事件	C 回调函数仿真和异步事件
C 结构仿真	可模拟 C 结构的内存操作功能
COM	COM / OLE 相关函数
公共对话框	一些公共对话框
转换	各种转换功能
日期和时间	提供有关时区的信息，将日期时间与常见的 C 类型进行相互转换
文件系统	检索信息或处理文件和目录

通用库	通用函数
低级文件处理	扩展的低级文件处理功能
内存分配	内存分配
杂项	各种有用的功能
网络	网络相关函数
ODBC	扩展的 ODBC 函数
打印机信息	提供有关打印机的信息
进程信息	提供有关系统运行的进程的信息
RAS(远程访问)	管理 RAS (远程访问) 连接
注册表	用于管理注册表的 API
资源信息	检索有关模块 (dll 和可执行文件) 中资源的信息
服务管理	检索信息并控制 Windows 服务
Shell	封装 Windows shell 的一些功能。
系统信息	检索有关系统的信息。
磁盘卷信息	检索有关磁盘卷的信息。
窗体信息	检索有关窗体的信息

数组

用于 VFP 数组的函数

AAverage	计算数组中值的算术平均值
AMax	获取数组中的最大值
AMin	获取数组中的最小值
ASum	对数组中的值求和

AAverage (数组元素的算术平均值)

计算数组中值的算术平均值。

AAverage(@aArray [, nDimension])

参数

@aArray

按引用传递的需要计算的数组。

nDimension (可选)

默认值：1

应在其上执行计算的数组的维数。

如果此参数为 0，则对数组的所有元素执行计算。

返回值

传入数组所有元素值的平均值。如果其中有元素为 .NULL.，则结果也为.NULL.。

备注

数组中值的有效类型：数字 (N) 或货币 (Y)

进行计算的数组（维度）中的所有值都必须是同一类型，否则会出现“无效参数”错误。

示例

```
LOCAL laTest[20], xj
FOR xj = 1 TO 20
laTest[xj] = xj
ENDFOR

? AAverage(@laTest) && 返回 10.5
```

参考

[AMax](#)

[AMin](#)

[ASum](#)

AMax (数组元素中的最大值)

获取数组中的最大值。

AMax (@aArray [, nDimension])

参数

@aArray

按引用方式传递的需要计算的数组。

nDimension (可选)

默认值： 1

应在其上执行计算的数组的维数。

如果此参数为 0，则对数组的所有元素执行计算。

返回值

传入数组所有元素值的最大值。如果其中有元素为 .NULL., 则结果也为.NULL..

备注

数组中值的有效类型：数字（N），货币（Y），日期（D）或日期时间（T）。

进行计算的数组（维度）中的所有值都必须是同一类型，否则会出现“无效参数”错误。

参考

[AAverage](#)

[AMin](#)

[ASum](#)

AMin (数组元素中的最小值)

获取数组中的最小值。

AMin (@aArray [, nDimension])

参数

@aArray

按引用方式传递的需要计算的数组。

nDimension (可选)

默认值： 1

应在其上执行计算的数组的维数。

如果此参数为 0，则对数组的所有元素执行计算。

返回值

传入数组所有元素值的最大值。如果其中有元素为 .NULL., 则结果也为.NULL..

备注

数组中值的有效类型：数字（N），货币（Y），日期（D）或日期时间（T）。

进行计算的数组（维度）中的所有值都必须是同一类型，否则会出现“无效参数”错误。

参考

[AAverage](#)

[AMax](#)[ASum](#)

ASum (数组元素的合计)

对数组中的元素求和。。

ASum (@aArray [, nDimension])

参数

@aArray

按引用方式传递的需要计算的数组。

nDimension (可选)

默认值：1

应在其上执行计算的数组的维数。

如果此参数为 0，则对数组的所有元素执行计算。

返回值

传入数组所有元素值的合计。如果其中有元素为 .NULL.，则结果也为.NULL.。

备注

数组中值的有效类型：数字 (N)，货币 (Y)。

进行计算的数组（维度）中的所有值都必须是同一类型，否则会出现“无效参数”错误。

如果值的类型为 Y 并且所有值的总和不在货币数据类型的范围内，则会抛出错误 1988 “货币值超出范围”。

参考

[AAverage](#)

[AMax](#)

[AMin](#)

C 数组仿真

函数将 VFP Cursor 和数组转换为 C 样式数组，反之亦然。

MarshalCArray2Cursor	将 C 数组转换为 VFP Cursor
MarshalCArray2FoxArray	将 C 数组转换为 VFP 数组
MarshalCursor2CArray	将 VFP Cursor 转换为 C 数组
MarshalFoxArray2CArray	将 VFP 数组转换为 C 数组

MarshalCArray2Cursor (C 数组转换为 VFP Cursor)

将 C 数组转换为 VFP Cursor。

MarshalCArray2Cursor (nAddress, cCursorAndFieldNames, nType, nRows [, nLength | nCodePage [, nCodePage]])

参数

nAddress

C 数组的基本地址(base address)。

cCursorAndFieldNames

Cursor 和字段名，例如 “ cursorname.fieldname, fieldname2”。

nType

C 数组的数据类型。它们可以取下表中的值

类型	C 数组声明
CTYPE_SHORT	short array[]
CTYPE USHORT	unsigned short array[]
CTYPE_INT	int array[]
CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[] []
CTYPE_WCHARARRAY	wchar_t array[] []
CTYPE_INT64	_int64 array[]
CTYPE_UINT64	unsigned _int64 array[]

nRows

要封送的行数。

nLength | nCodePage (可选)

字符数组的长度或用于 unicode 到 ansi 转换的代码页。

nCodePage (可选)

用于 unicode 到 ansi 转换的代码页。

返回值

.T.

备注

对于字符类型指针数组，CTYPE_CSTRING 和 CTYPE_WSTRING 将 0 指针转换为 NULL 值。

参考

[MarshalCArray2FoxArray](#)

[MarshalCursor2CArray](#)

[MarshalFoxArray2CArray](#)

MarshalCArray2FoxArray (C 数组转换为 VFP 数组)

将 C 数组转换为 VFP Cursor。

MarshalCArray2FoxArray(nAddress, @aArray, nType [, nLength | nCodePage [, nCodePage]])

参数**nAddress**

C 数组的基本地址(base address)。

@aArray

按引用传递的目标 VFP 数组。数组的大小取决于 C 数组的大小。

注意：行和列的最大值为 65000。

nType

C 数组的数据类型。它们可以取下表中的值

类型	C 数组声明
CTYPE_SHORT	short array[]
CTYPE USHORT	unsigned short array[]
CTYPE_INT	int array[]

CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[] []
CTYPE_WCHARARRAY	wchar_t array[] []
CTYPE_INT64	__int64 array[]
CTYPE_UINT64	unsigned __int64 array[]

nLength | nCodePage (可选)

字符数组的长度或用于 unicode 到 ansi 转换的代码页。

nCodePage (可选)

用于 unicode 到 ansi 转换的代码页。

返回值

.T.

备注

对于字符类型指针数组，CTYPE_CSTRING 和 CTYPE_WSTRING 将 0 指针转换为 NULL 值。

参考

[MarshalCArray2Cursor](#)

[MarshalCursor2CArray](#)

[MarshalFoxArray2CArray](#)

MarshalCursor2CArray (VFP Cursor 转换为 C 数组)

将 VFP Cursor 转换为 C 数组。

MarshalCursor2CArray(nAddress, cCursorAndFieldNames, nType [, nLength | nCodePage [, nCodePage]])

参数**nAddress**

C 数组的基本地址(base address)。

cCursorAndFieldNames

Cursor 和字段名，例如 “ cursorname.fieldname, fieldname2” 。

nType

C 数组的数据类型。它们可以取下表中的值：

类型	C 数组声明
CTYPE_SHORT	short array[]
CTYPE USHORT	unsigned short array[]
CTYPE_INT	int array[]
CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[] []
CTYPE_WCHARARRAY	wchar_t array[] []
CTYPE_INT64	_int64 array[]
CTYPE_UINT64	unsigned _int64 array[]

nLength | nCodePage (可选)

字符数组的长度或用于 unicode 到 ansi 转换的代码页。

nCodePage (可选)

用于 unicode 到 ansi 转换的代码页。

返回值

.T.

备注

Cursor 的字段必须为正确的类型或 NULL，否则会出现 “无效参数” 错误。

NULL 值将被忽略，并且当前数组索引处的值不会更改。

对于字符类型的指针数组，CTYPE_CSTRING 和 CTYPE_WSTRING 的 NULL 值导致指针为 0。

参考

[MarshalCArray2Cursor](#)

[MarshalCArray2FoxArray](#)

[MarshalFoxArray2CArray](#)

MarshalFoxArray2CArray (VFP 数组转换为 C 数组)

将 VFP 数组转换为 C 数组。

MarshalFoxArray2CArray(nAddress, @aArray, nType [, nLength | nCodePage [, nCodePage]])

参数

nAddress

C 数组的基本地址(base address)。

@aArray

按引用传递的目标 VFP 数组。

nType

C 数组的数据类型。它们可以取下表中的值

类型	C 数组声明
CTYPE_SHORT	short array[]
CTYPE USHORT	unsigned short array[]
CTYPE_INT	int array[]
CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[] []
CTYPE_WCHARARRAY	wchar_t array[] []
CTYPE_INT64	_int64 array[]
CTYPE_UINT64	unsigned _int64 array[]

nLength | nCodePage (可选)

字符串数组的长度或用于 unicode 到 ansi 转换的代码页。

nCodePage (可选)

用于 unicode 到 ansi 转换的代码页。

返回值

.T.

备注

数组的元素必须为正确的类型或 NULL，否则将引发“无效参数”错误。

NULL 值将被忽略，并且当前数组索引处的值不会更改。

对于字符类型的指针数组，CTYPE_CSTRING 和 CTYPE_WSTRING 的 NULL 值导致指针为 0。

参考

[MarshalCArray2Cursor](#)

[MarshalCArray2FoxArray](#)

[MarshalCursor2CArray](#)

C 回调和事件

C 回调函数仿真和异步事件

AsyncWaitForObject	在单独的线程上等待 API 句柄而不阻塞，当句柄发出信号时，将执行提供的回调函数
BindEventsEx	提供在 API 窗口收到指定的窗口消息时执行函数或对象方法的功能
CancelWaitForObject	停止等待句柄发出信号的线程
CreateCallbackFunc	创建一个模拟 C 回调函数的程序集 thunk
CreatePublicShadowObjReference	创建一个新的公共变量，引用提供的对象，而不增加对象引用计数。
DestroyCallbackFunc	释放传入的 C 回调函数
ReleasePublicShadowObjReference	释放引用使用 CreatePublicShadowObjReference 创建的对象的公共变量。
UnbindEventsEx	取消绑定先前与 BindEventsEx 绑定的窗口的事件。

AsyncWaitForObject (异步等待)

在单独的线程上等待 API 句柄而不阻塞，当句柄发出信号时，将执行提供的回调函数。

AsyncWaitForObject (nHandle, cCallback)

参数

nHandle

要监视的句柄。

cCallback

发出信号时调用的函数。该函数应具有以下原型：

```
Function ObjectSignaled
    Lparameters nError
    If nError = 0
        ? 'Object signaled'
    Else
        ? 'Function: WaitForMultipleObjects failed with error', nError
    Endif
Endfunc
```

返回值

创建的线程句柄（数值）。

示例

译者注：本示例为 vfp2c32 发行版中的 [Demo](#) (async.prg)。

```
#INCLUDE "vfp2c.h"

CD (FULLPATH(JUSTPATH(SYS(16))))
SET LIBRARY TO vfp2c32.f11 ADDITIVE

&& 监视目录中的更改
LOCAL lcDir
m.lcDir = GETDIR(' ', ' ', '选择目录以查看更改')
IF !EMPTY(m.lcDir)
    PUBLIC loFileMonitor
    loFileMonitor = CREATEOBJECT('FileSystemWatcherEx')
    loFileMonitor.Watch(m.lcDir, .T., FILE_NOTIFY_CHANGE_FILE_NAME +
FILE_NOTIFY_CHANGE_DIR_NAME)
ENDIF

DEFINE CLASS AsyncWatcher AS Custom
    PROTECTED hHandle
    PROTECTED cPublicName
    hHandle = 0
    cPublicName = ''

FUNCTION Init
    THIS.cPublicName = THIS.Class + SYS(2015)
    CreatePublicShadowObjReference(THIS.cPublicName, THIS)
ENDFUNC
```

```

FUNCTION Destroy
    THIS.Stop()
    ReleasePublicShadowObjReference(THIS.cPublicName)
ENDFUNC

FUNCTION Stop
ENDFUNC

FUNCTION Watch
ENDFUNC

ENDDEFINE

DEFINE CLASS FileSystemWatcherEx AS AsyncWatcher
    PROTECTED oWatchedDirs
    oWatchedDirs = .NULL.

    FUNCTION Init
        DODEFAULT()
        THIS.oWatchedDirs = CREATEOBJECT('Collection')
    ENDFUNC

    FUNCTION Stop
        LPARAMETERS cPath
        LOCAL lnHandle
        DO CASE
            CASE VARTYPE(m.cPath) = 'L'
                CancelFileChangeEx(0)
                THIS.oWatchedDirs = CREATEOBJECT('Collection')
            CASE VARTYPE(m.cPath) = 'C'
                m.cPath = ADDBS(LOWER(m.cPath))
                m.lnHandle = THIS.oWatchedDirs.Item(m.cPath)
                THIS.oWatchedDirs.Remove(m.cPath)
                CancelFileChangeEx(m.lnHandle)
        ENDCASE
    ENDFUNC

    FUNCTION Watch
        LPARAMETERS cPath, bWatchSubtree, nFilter
        LOCAL lnHandle
        m.lnHandle = FindFileChangeEx(m.cPath, m.bWatchSubtree, m.nFilter,
THIS.cPublicName + '.FolderChanged')
        IF m.lnHandle > 0
            THIS.oWatchedDirs.Add(m.lnHandle, ADDBS(LOWER(m.cPath)))
        ENDIF
    ENDFUNC

```

```
ENDFUNC
```

&& 异步模式下调用的回调函数，你可以以你认为合理的方式进行定义

```
FUNCTION FolderChanged
    LPARAMETERS hHandle, nReason, cPath, cPath2
    DO CASE
        CASE m.nReason = 0
            ? '函数 ' + m.cPath + " 失败。错误编号: ", m.cPath2
        CASE m.nReason = 1
            ? '文件增加: ' + m.cPath
        CASE m.nReason = 2
            ? '文件移除: ' + m.cPath
        CASE m.nReason = 3
            ? '文件编辑: ' + m.cPath
        CASE m.nReason = 4
            ? '文件名从: ' + m.cPath + ' 更改到: ' + m.cPath2
    ENDCASE
ENDFUNC
```

```
ENDDEFINE
```

参考

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[CreatePublicShadowObjReference](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

使用的 WinApi

[WaitForMultipleObjects](#)

[CreateEvent](#)

[CreateThread](#)

[CloseHandle](#)

[PostMessage](#)

BindEventsEx (高级绑定)

提供在 API 窗口收到指定的窗口消息时执行函数或对象方法的功能。

BindEventsEx(nHwnd, nMsg, oObject, cFunction [, cParmDefinition [, nFlags]])

参数

nHwnd

应当拦截消息的窗口的句柄。

nMsg

应当触发回调函数的窗口消息，例如 WM_MOUSEMOVE, WM_ACTIVATE

oObject

如果 cFunction 是公共的 FUNCTION / PROCEDURE，则应调用在 cFunction 中传递函数的对象，或者 NULL。

cFunction

窗口收到消息时应调用的函数或方法的名称。

cParmDefinition (可选)

传递给函数的以逗号分隔的参数列表。

如果省略 cParmDefinition 或传递 NULL，则默认值类似于 BINDEVENTSEX，即将 4 个参数传递给回调函数：

FUNCTION YourFunc(Hwnd, uMsg, wParam, lParam)

wParam 和 lParam 被限制为 signed integer。您还可以指定自己的函数定义，例如：

&& 你仅需要 wParam & lParam 作为参数

BINDEVENTSEX(yourHwnd, WM_SOMEMESSAGE, NULL, 'SomeFunc', 'wParam,lParam')

&& 如果你不需要参数则可以传递一个空串

BINDEVENTSEX(yourHwnd, WM_SOMEMESSAGE, NULL, 'SomeFunc', '')

下表为可以使用的值：

值	描述
hwnd, uMsg, wParam or lParam	参数为 signed integer
UNSIGNED(wParam or lParam)	wParam 或 lParam 为 unsigned integer
HIWORD(wParam or lParam)	高位 (高 16 位) 为 signed short
LOWORD(wParam or lParam)	低位 (低 16 位) 为 signed shorg
UNSIGNED(HIWORD(wParam or lParam))	高位 (高 16 位) 为 unsigned short
UNSIGNED(LOWORD(wParam or lParam))	低位 (低 16 位) 为 unsigned short

BOOL(wParam or lParam)	参数转换为逻辑值。除 0 为.F.外，其他为.T.
------------------------	---------------------------

nFlags (可选, 附加)

如果省略 nFlags 参数或传递 0，则默认值为 BINDEVENTSEX_CALL_BEFORE。

标记常量	描述
BINDEVENTSEX_CALL_BEFORE	在调用 CallWindowProc 之前调用你的函数
BINDEVENTSEX_CALL_AFTER	在调用 CallWindowProc 之后调用你的函数。 必须从回调函数返回一个数值，该值用作底层窗口过程的返回值。
BINDEVENTSEX_RETURN_VALUE	CallWindowProc 不会被自动调用，你可以在回调函数中自行调用或者忽略它。 Y 必须从回调函数返回一个数值，该值用作底层窗口过程的返回值。
BINDEVENTSEX_NO_RECURSION	与 BINDEVENTS 中的相同。 在调用函数前 _VFP.AutoYield 设置为 .F.，然后将其重置为原始值以防止递归。
BINDEVENTSEX_CLASSPROC	传递的 <i>nHwnd</i> 的窗口类不是实际的窗口，而是子类的。 从 windowclass 创建的所有后续窗口都将被自动子类化。 !! 您必须将第三个参数设置为 .T. 来调用 UnbindEventsEx 。 如果您从 windowclass 取消绑定消息！

注意：BINDEVENTSEX_CALL_BEFORE, BINDEVENTSEX_CALL_AFTER 和 BINDEVENTSEX_RETURN_VALUE 是互斥的，这意味着您只能指定一个值，否则会出现错误“无效参数”。

返回值

从 [GetWindowLong](#) (*nHwnd*, *GWL_WNDPROC*) 返回的指向原始窗口过程的指针（数字）。

备注

BindEventsEx 的行为几乎与 BINDEVENT 完全相同，区别在于：

1. 它返回指向原始窗口过程的指针，而不是没有任何有意义的值，因此您不必调用 [GetWindowLong](#) (*nHwnd*, *GWL_WNDPROC*)。
2. 您不必将消息绑定到对象方法，它也可以是公共函数/过程
3. 默认情况下，您不必调用 [CallWindowProc](#)，在调用函数之前/之后，将使用原始参数自动调用 [CallWindowProc](#)，此行为可通过 nFlags 参数控制
4. 您可以确切指定应将什么参数发送到回调函数/方法，以及如何将其编组。

5.您不能像 BINDEVENT 中那样将 0 作为 nHwnd 参数传递来绑定到所有窗口

为了在对象上模拟回调，库必须将对象的副本复制到公共变量中。

变量的名称是通过以下方案自动生成的：

```
lcVarName = "__VFP2C_WCBO" + Iif(Bitand(nFlags, BINDEVENTSEX_CLASSPROC) > 0, "1", "0")
+ "__" + Alltrim(Str(nHwnd)) + "__" + Alltrim(Str(nMsg))
```

由于 FoxPro LCK 不提供 API 函数来调用对象的方法，因此必须采取这种解决方法。

取消绑定消息时，公共变量会自动释放。

尽管已创建对象的副本，但内部对象引用计数不会增加，但公共副本不会影响对象的生命周期（作用域）。

您唯一需要考虑的是，您自己的变量不会与上述命名方案冲突，这种情况基本不可能发生。

示例

使用适当的参数绑定窗口的 mousemove 事件。

```
BINDEVENTSEX(yourHwnd, WM_MOUSEMOVE, Null, 'MouseMoveCallback', 'wParam,
LOWORD(lParam), HIWORD(lParam)')
Function MouseMoveCallback(nKeyState, nXCoord, nYCoord)
&& 这里可以控制鼠标.....
Endfunc
```

绑定应用程序窗口的 Activate/Deactivate。

```
BINDEVENTSEX(_Screen.Hwnd, WM_ACTIVATE, 'AppFocusStateChanged', 'BOOL(wParam)')
Function AppFocusStateChanged(bState)
? "应用 " + Iif(m.bState, '获得焦点', '失去焦点')
Endfunc
```

参考

[AsyncWaitForObject](#)
[CancelWaitForObject](#)
[CreateCallbackFunc](#)
[CreatePublicShadowObjReference](#)
[DestroyCallbackFunc](#)
[ReleasePublicShadowObjReference](#)
[UnbindEventsEx](#)

使用的 WinAPI

[SetWindowLong](#)

[GetWindowLong](#)[SetClassLong](#)[GetClassLong](#)[CallWindowProc](#)[HeapAlloc](#)[VirtualProtect](#)

CancelWaitForObject (取消异步等待)

停止等待信号通知 HANDLE 的线程。

CancelWaitForObject (nThreadHandle)

参数

nThreadHandle

从 [AsyncWaitForObject](#) 返回的线程句柄。

返回值

如果监视该句柄的线程终止则返回 .T., 否则返回.F.。

参考

[AsyncWaitForObject](#)[BindEventsEx](#)[CreateCallbackFunc](#)[CreatePublicShadowObjReference](#)[DestroyCallbackFunc](#)[ReleasePublicShadowObjReference](#)[UnbindEventsEx](#)

使用的 WinApi

[SetEvent](#)

CreateCallbackFunc (创建回调函数)

创建一个模拟 C 回调函数的程序集 thunk。

CreateCallbackFunc (cCallback, cReturnType, cParamTypes [, oObject [, nFlags]])

参数

cCallback

回调函数的名称。

cReturnType

C 回调函数的返回值的数据类型。其取值为以下列表之一：

数据类型	描述
VOID 或空字符串	无返回值
INTEGER or LONG	signed 32bit integer
UINTEGER or ULONG	unsigned 32bit integer
SHORT	signed 16bit integer
USHORT	unsigned 16bit integer
BOOL	boolean 32bit 值 - 0 映射为 .F., 其他映射为 .T.
SINGLE	32bit floating point 值
DOUBLE	64bit floating point 值

cParamTypes

用逗号分隔的 C 函数期望的参数类型列表。有效类型：

数据类型	描述
INTEGER or LONG	signed 32bit integer
UINTEGER or ULONG	unsigned 32bit integer
SHORT	signed 16bit integer
USHORT	unsigned 16bit integer
BOOL	boolean 32bit value - 0 映射为 .F., 其他映射为 .T.
SINGLE (0-6)	32bit floating point value (1)
DOUBLE (0-16)	64bit floating point value (1)
STRING (ANSI UNICODE)	C style string (2)
INT64 (BINARY LITERAL CURRENCY)	signed 64 bit interger value (3)
UINT64 (BINARY LITERAL CURRENCY)	unsigned 64 bit integer value (3)

注意：

- 对于 SINGLE 和 DOUBLE 数据类型，可以指定一个可选的精度值，例如

InCallback = CreateCallbackFunc ('Foo', 'DOUBLE 15', 'INTEGER, INTEGER')

如果省略精度值，则使用默认值 6。

2. 默认情况下，STRING 被封装为数字指针值。

您可以通过将值传递给 ReadCString (Ansi) 或 ReadWString (Unicode) 函数来获取实际的字符串。

如果另外传递 ANSI 或 UNICODE，则这些函数会自动将字符串编组。

3. 默认情况下，带符号和无符号的 64 位整数被封装为数字值。回调将截断值。

如果参数超过 VFP numeric 限制。如果您不是 100% 确定从不传递如此大的数字，请指定 BINARY, LITERAL 或 CURRENCY 修饰符。BINARY 将参数编组为 8 字节的 varbinary 值，LITERAL 将参数编组为数字字符串，而 CURRENCY 将参数编组为货币文字。

4. 参数的数量限制为 27 个，你可以不考虑 VFP 函数的参数个数限制。

oObjectRef (可选)

如果应对对象引用上调用 FoxPro 方法，则可以在此参数中传递对象引用。

如果要传递 nFlags 参数，但又不想传递对象，则传递 NULL。

nFlags (可选，附加)

默认值： CALLBACK_SYNCRONOUS

有效值：

Flag	描述
CALLBACK_SYNCRONOUS	在 VFP 主线程上调用 C 回调函数。对所有 WINAPI 有效，例如： EnumWindows 。
CALLBACK_ASYNCROUS_POST	对于引发事件的第三方 C DLL，在单独的线程上调用 C 回调函数（调用 C 函数）。回调本身是异步发送到 Visual FoxPro 主线程的，它不会阻止第三方线程，当 FoxPro 处于 READ EVENTS 状态时，将处理该事件。
CALLBACK_ASYNCROUS_SEND	对于引发事件的第三方 C DLL，在单独的线程上调用 C 回调函数（调用 C 函数）。回调被同步发送到 Visual FoxPro 主线程，该线程在 FoxPro 函数返回后恢复。
CALLBACK_CDECL	C 函数使用 cdecl 调用约定，默认情况下，回调函数是使用 stdcall 调用约定创建的。

注意

如果指定 CALLBACK_ASYNCROUS_POST，则多个回调可以相互插入，将 _VFP.AutoYield 设置为 .F. 可避免这种情况的发生 - 适用于整个应用程序或 FoxPro 函数内部的回调函数，例如：

```
Function YourCallbackFunction(lnParameter1, lnParameter2)
```

```

_vfp.AutoYield = .F.
&& 你自己的代码 ...
_vfp.AutoYield = .T.
Endfunc

```

返回值

指向创建的回调函数的指针（数字）。

备注

为了在对象上模拟回调，库必须将对象的副本复制到公共变量中。

变量的名称是通过以下方案自动生成的：

```
lcVarName = "__VFP2C_CBO_" + Alltrim(Str(returnvalue))
```

由于 FoxPro LCK 不提供 API 函数来调用对象的方法，因此必须采取这种解决方法。

取消绑定消息时，公共变量会自动释放。

尽管已创建对象的副本，但内部对象引用计数不会增加，但公共副本不会影响对象的生命周期（作用域）。

您唯一需要考虑的是，您自己的变量不会与上述命名方案冲突，这种情况基本不可能发生。

参考

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreatePublicShadowObjReference](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

CreatePublicShadowObjReference (创建公共影子对象引用)

创建一个新的公共变量，引用提供的对象，而不增加对象引用计数。

CreatePublicShadowObjReference (cVariableName, oObject)

参数

cVariableName

被创建的公共变量的名称。

oObject

公共变量应引用的对象。

返回值

.T.

参考

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

DestroyCallbackFunc (取消回调函数)

释放传入的 C 回调函数。

DestroyCallbackFunc (nCallbackAddress)

参数

nCallbackAddress

从 [CreateCallbackFunc](#) 函数返回的指向 C 回调函数的指针。

返回值

如果成功释放返回.T.; 否则返回.F.。

参考

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[CreatePublicShadowObjReference](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

ReleasePublicShadowObjReference (释放公共影子对象引用)

释放使用 [CreatePublicShadowObjReference](#) 创建的公共变量对象的引用。

ReleasePublicShadowObjReference (cVariableName)

参数

cVariableName

要释放的变量的名称。

返回值

.T.

参考

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[CreatePublicShadowObjReference](#)

[DestroyCallbackFunc](#)

[UnbindEventsEx](#)

UnbindEventsEx (取消高级绑定)

取消先前与 BindEventsEx 绑定的窗口事件的绑定。

UnbindEventsEx (nHwnd [, nMsg [, bClass]])

参数

nHwnd

指定解绑事件的窗口的句柄。

nMsg (可选)

该消息将被解绑。

如果省略 nMsg 参数或传递 0，则所有消息均解绑，并且窗口将取消类化。

bClass (可选)

默认值：.F.

如果使用 BINDEVENTSEX_CLASSPROC 标志将 windowclass 子类化，则必须传递.T.。

返回值

.T.

参考

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[CreatePublicShadowObjReference](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

使用的 WinApi

[SetWindowLong](#)

[SetClassLong](#)

C 结构仿真

可模拟 C 结构的内存操作功能

ReadBytes	从指定地址返回一个字节范围。
ReadChar	从指定地址返回一个 C 字符 (单个字符)。
ReadCharArray	从 C 样式字符数组中检索字符串。
ReadCString	从指定地址返回一个 C 字符串。
ReadDouble	从指定地址返回一个双精度 (64-bit floating point value)
ReadFloat	从指定地址返回一个浮点数 (32-bit floating point value)
ReadInt	从指定地址返回一个 32 位整数 (32-bit integer)
ReadInt64	从指定地址返回一个 64 位带符号整数 (64-bit signed integer)
ReadInt8	从指定地址返回一个 8 位整数 (8-bit integer)

ReadLogical	从指定地址返回一个逻辑值
ReadPChar	从指定的间接地址返回 C 字符 (单个字符)。
ReadPCString	从指定的间接地址返回 C 字符串。
ReadPDouble	从指定的间接地址返回一个双精度 (64-bit floating point value)
ReadPFloat	从指定的间接地址返回一个浮点数 (32-bit floating point value)
ReadPInt	从指定的间接地址返回一个 32 位整数 (32-bit integer)
ReadPInt64	从指定的间接地址返回一个 64 位带符号整数 (64-bit signed integer)
ReadPInt8	从指定的间接地址返回一个 8 位整数 (8-bit integer)
ReadPLogical	从指定的间接地址返回一个逻辑值
ReadPointer	从指定地址返回指针。
ReadPPointer	从指定的间接地址返回指针。
ReadProcessMemoryEx	从另一个进程的内存空间中返回一定范围的字节。
ReadPShort	从指定的间接地址中返回一个 16 位整数 (16-bit integer)
ReadPUInt	从指定的间接地址中返回 32 位无符号整数 (32-bit unsigned integer)
ReadPUInt64	从指定的间接地址中返回 64 位无符号整数 (64-bit unsigned integer)
ReadPUInt8	从指定的间接地址中返回一个 8 位无符号整数 (8-bit unsigned integer)
ReadPUSHort	从指定的间接地址中返回一个 16 位无符号整数 (16-bit unsigned integer)
ReadPWString	从指定的间接地址返回转换为 Ansi 的 Unicode 字符串。
ReadShort	从指定地址返回一个 16 位整数 (16-bit integer)
ReadUInt	从指定的地址返回一个 32 位无符号整数 (32-bit unsigned integer)
ReadUInt64	从指定地址返回一个 64 位无符号整数 (64-bit unsigned integer)
ReadUInt8	从指定地址返回一个 8 位无符号整数 (8-bit unsigned integer)
ReadUShort	从指定地址返回一个 16 位无符号整数 (16-bit unsigned integer)
ReadWCharArray	从 C 风格的 unicode 字符数组中返回一个字符串。
ReadWString	从指定地址返回转换为 Ansi 的 Unicode 字符串。
WriteBytes	将二进制数据写入指定的地址
WriteChar	在指定地址写入一个字符
WriteCharArray	在指定地址写入一个字符串
WriteCString	分配或重新分配 C 样式字符串
WriteDouble	在指定地址写入一个双精度值(64-bit floating point)
WriteFloat	在指定地址写入一个浮点数(32-bit floating point)
WriteGPCString	为 C 风格的字符串分配或重新分配内存，并在指定地址将指针写入该字符串
WriteInt	在指定地址写入 32 位整数 (32-bit integer)
WriteInt64	在指定地址写入一个 64 位带符号整数(64-bit signed integer)
WriteInt8	在指定地址写入 8 位整数(8-bit integer)

WriteLogical	将逻辑值写入指定的地址
WritePChar	在指定的间接地址处写入单个字符
WritePCString	为 C 风格的字符串分配或重新分配内存，并在指定地址将指针写入该字符串。
WritePDouble	在指定的间接地址处写入一个双精度(64-bit floating point)
WritePFloat	在指定的间接地址处写入浮点数(32-bit floating point)
WritePInt	在指定的间接地址处写入 32 位整数(32-bit integer)
WritePInt64	在指定的间接地址处写入 64 位带符号整数(64-bit signed integer)
WritePInt8	在指定的间接地址处写入一个 8 位整数(8-bit integer)
WritePLogical	在指定的间接地址处写入逻辑值
WritePointer	将指针写入指定的地址
WritePPointer	将指针写入指定的间接地址
WritePShort	在指定的间接地址处写入 16 位整数(16-bit integer)
WritePUInt	在指定的间接地址处写入 32 位无符号整数(32-bit unsigned integer)
WritePUInt64	在指定的间接地址处写入 64 位无符号整数(64-bit unsigned integer)
WritePUInt8	在指定的间接地址写入 8 位无符号整数(8-bit unsigned integer)
WritePUShort	在指定的间接地址处写入 16 位无符号整数 (16-bit unsigned integer)
WritePWChar	在指定的间接地址处写入单个 Unicode 字符
WritePWString	为 C 风格的 unicode 字符串分配或重新分配内存，并在指定地址将指针写入该字符串。
WriteShort	在指定的地址写入 16 位整数 (16-bit integer)
WriteUInt	在指定地址写入 32 位无符号整数 (32-bit unsigned integer)
WriteUInt64	在指定地址写入一个 64 位无符号整数 (64-bit unsigned integer)
WriteUInt8	在指定地址写入一个 8 位无符号整数 (8-bit unsigned integer)
WriteUShort	在指定地址写入一个 16 位无符号整数 (16-bit unsigned integer)
WriteWChar	在指定地址写入一个 Unicode 字符
WriteWCharArray	在指定地址写入 Unicode 字符串
WriteWString	分配或重新分配 C 样式 Unicode 字符串

ReadBytes

ReadChar

ReadCharArray

ReadCString

ReadDouble

ReadFloat

ReadInt

ReadInt64

ReadInt8

ReadLogical

ReadPChar

ReadPCString

ReadPDouble

ReadPFloat

ReadPInt

ReadPInt64

ReadPInt8

ReadPLogical

ReadPointer

ReadPPointer

ReadProcessMemoryEx

ReadPShort

ReadPUInt

ReadPUInt64

ReadPUInt8

ReadPUSHort

ReadPWString

ReadShort

ReadUInt

ReadUInt64

ReadUInt8

ReadUShort

ReadWCharArray

ReadWString

WriteBytes

WriteChar

WriteCharArray

WriteCString

WriteDouble

WriteFloat

WriteGPCString

WriteInt

WriteInt64

WriteInt8

WriteLogical

WritePChar

WritePCString

WritePDouble

WritePFloat

WritePInt

WritePInt64

WritePInt8

WritePLogical

WritePointer

WritePPointer

WritePShort

WritePUInt

WritePUInt64

WritePUInt8

WritePUSHort

WritePWChar**WritePWString****WriteShort****WriteUInt****WriteUInt64****WriteUInt8****WriteUShort****WriteWChar****WriteWCharArray****WriteWString****COM**

COM/OLE 相关函数

CLSIDFromProgID	返回给定 ProgID (COM 类名) 的二进制 CLSID。 例如：“VisualFoxPro.Application”。
CLSIDFromString	将可识别的 CLSID 转换为二进制 CLSID。这是 StringFromCLSID 函数的反函数。
CreateGuid	创建一个新的 GUID (全局唯一标识符)。
CreateThreadObject	在单独的线程上创建一个 COM 对象。
GetIUnknown	返回指向 COM 对象的 IUnknown 接口的指针。
IsEqualGuid	比较两个 GUID 是否相等。
ProgIDFromCLSID	返回给定 CLSID 的 ProgID。

RegisterActiveObject	将传递的对象注册为 cProgID 中指定的类的活动对象。 注册会导致该对象在 OLE 的运行对象表 (ROT) 中列出，该对象是全局可访问的查找表，用于跟踪计算机上当前正在运行的对象。
RegisterObjectAsFileMoniker	为传递的对象创建文件名字对象。
RevokeActiveObject	从运行对象表 (ROT) 中注销对象。
StringFromCLSID	将全局唯一标识符 (GUID) 转换为可识别字符的字符串。

CLSIDFromProgID (从 ProgID 返回 CLSID)

返回给定 ProgID (COM 类名) 的二进制 CLSID。例如：“VisualFoxPro.Application”。

CLSIDFromProgID(cProgID)

参数

cProgID

要返回 CLSID 的 ProgID。

返回值

CLSID (二进制字符)。

备注

程序标识符 (ProgID) 是可以与 CLSID 关联的注册表项。像 CLSID 一样，ProgID 标识一个类，但精度较低，因为不能保证它是全局唯一的。ProgID 的格式为<Program>.<Component>.<Version>，用句点分隔并且没有空格，例如：Word.Document.6

示例

```
1cCLSID = CLSIDFromProgID('VisualFoxPro.Application')
&& 1cCLSID 此时是二进制格式
? StringFromCLSID(1cCLSID) && 转换成可识别的字符
```

参考

[CLSIDFromString](#)

[CreateGuid](#)

[CreateThreadObject](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)[RegisterActiveObject](#)[RegisterObjectAsFileMoniker](#)[RevokeActiveObject](#)[StringFromCLSID](#)

使用的 Winapi

[CLSIDFromProgID](#)

CLSIDFromString(将可识别 CLSID 转换为二进制格式)

将可识别的 CLSID 转换为二进制 CLSID。 这是 [StringFromCLSID](#) 函数的反函数。

CLSIDFromString (cClsID)

参数

cClsId

可读的 CLSID，例如 “{002D2B10-C1FA-4193-B134-D86EAEC5250}”。

返回值

CLSID (二进制字符)。

备注

CLSID 是标识 COM 类对象的全局唯一标识符。 如果服务器或容器允许链接到其嵌入式对象，则需要为每种受支持的对象类注册一个 CLSID。

示例

```
lcGuidString = StringFromCLSID(CLSIDFROMPROGID("Word.Application"))
lcClsId = CLSIDFromString(lcGuidString)
? lcClsId
```

参考

[CLSIDFromProgID](#)[CreateGuid](#)[CreateThreadObject](#)[GetIUnknown](#)[IsEqualGuid](#)

[ProgIDFromCLSID](#)[RegisterActiveObject](#)[RegisterObjectAsFileMoniker](#)[RevokeActiveObject](#)[StringFromCLSID](#)

使用的 WinApi

[CLSIDFromString](#)

CreateGuid (创建 GUID)

创建一个新的 GUID (全局唯一标识符)。

CreateGuid([nOutputFormat])

参数

nOutputFormat (可选)

默认值: CREATE_GUID_ANSI。 可选值:

OutputFormat	返回值
CREATE_GUID_ANSI	Guid 作为 ansi 字符串返回
CREATE_GUID_UNICODE	Guid 作为 unicode 字符串返回
CREATE_GUID_BINARY	Guid 以二进制格式返回

返回值

GUID, 其格式取决于您发送的参数。

备注

尽管一般认为 GUID 是全局唯一的, 但不能保证它是全局唯一的。如果打算将应用程序部署到其他行星上, 则应谨慎使用。

如果你在 VFP 表中为主键设置的默认值使用这个函数创建的 GUID, 那么, 在你添加新记录时, 此函数必须保证可用, 否则会出现错误。您可能需要在打开数据库时使用数据库触发器来加载 VFP2C32, 以确保此函数可用。

示例

```
cId = CreateGuid()
? cId
```

* 返回可识别的由36个字符组成的GUID

参考

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateThreadObject](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)

[RegisterActiveObject](#)

[RegisterObjectAsFileMoniker](#)

[RevokeActiveObject](#)

[StringFromCLSID](#)

使用的 WinApi

[CoCreateGuid](#)

[StringFromGUID2](#)

CreateThreadObject (创建线程对象)

在单独的线程上创建一个 COM 对象。

CreateThreadObject(cClassName [, oCallback [, bSynchronousAccess [, nContext [, nStackSize]]]])

参数

cClassName

指定从中创建新对象的 COM 类。

oCallback

默认值 .NULL.

当创建的线程对象上的方法完成或发生错误时，此对象引用的方法将被触发。

如果传递 NULL，则不会在方法完成或发生错误时收到任何信息。

如果传递对象，则此对象必须至少实现以下方法之一：

Function OnCallComplete(callid, result, callcontext)

```

    && callid = 分配给每个调用的唯一调用ID (integer)
    && result = 方法的返回值
    && callcontext = 与调用相关联的 callcontext 值
Function OnError(callid, callcontext, errornumber, errorsource, errordescription)
    && callid = 分配给每个调用的唯一调用ID (integer)
    && callcontext = 与调用相关联的 callcontext 值
    && errornumber = 错误编号
    && errorsource = 发生错误的原因
    && errordescription = 发生的错误的描述

```

注意:

您应该通过在 COM 类内使用 COMRETURNERROR 函数来实现正确的 COM 错误处理，以使 OnError 回调方法返回正确的信息。

bSynchronousAccess

默认值: .F.

如果设置为 .T.，所创建的代理对象将支持 “Object” 属性，以同步访问创建的对象。

nContext

默认值: CLSCTX_INPROC_SERVER

创建的 COM 对象的执行上下文。

此参数可以是以下值之一：

执行上下文	描述
CLSCTX_INPROC_SERVER	创建和管理此类对象的代码是一个 DLL，它与指定类上下文的函数的调用方在同一进程中运行。
CLSCTX_LOCAL_SERVER	创建和管理此类的对象的 EXE 代码在同一台计算机上运行，但在单独的进程空间中加载。

nStackSize

默认值: 64 KB

指定创建的线程的堆栈大小 (以字节为单位)。

返回值

创建的对象的代理对象，允许您异步调用函数。

备注

此功能使您可以创建真正的多线程 FoxPro 程序。

实际上可以在 FoxPro 中创建多线程代码，因为您可以将 FoxPro 代码编译为 COM 类。问题是您只能在其他多线程感知的环境/语言 (例如 IIS, C ++或 C #) 中以多线程方式运行此代码。

CreateThreadObject 函数提供了难题中缺少的部分-从 Visual FoxPro 中的自己的线程上创建具有多线程功能的 COM 对象。

您进行的每个方法调用或属性访问/分配都在所创建的对象“所在”的线程上执行，该调用在后台执行该方法时立即返回。返回值始终是分配给该呼叫的唯一标识符值 (callid)。

所有呼叫都放入 FIFO (先进先出) 队列中，并按照发出的顺序进行处理。队列的大小仅受可用内存的限制，因此您可以根据需要在对象上排队尽可能多的调用。

线程的生存期绑定到返回的代理对象。如果代理对象被破坏，线程也将被破坏。

当对象被销毁时，线程仍在执行方法时，销毁将阻塞，直到方法完成执行为止。仍在队列中的所有其他呼叫将被丢弃。

返回的代理对象还实现了一些用于管理方法中止和其他任务的功能。

类型	名称	描述
方法	AbortCall(callid)	如果传递了 0，则中止所有呼叫，否则中止给定 callid 的呼叫。
方法	GetCallQueueSize	返回队列中未决呼叫的数量。当前处理的呼叫也被计数。
方法	GetCallRunTime(callid)	返回调用运行的时间（以毫秒为单位）。如果呼叫已完成或仍在队列中，则返回 0。如果传递 0，则返回当前处理的调用的运行时间。
属性	Object	访问 Object 属性时，所有调用都是同步进行的。此属性仅当你在 bSynchronousAccess 参数传递 .T. 时可用。
属性	ThreadId	只读属性，该属性返回在其上运行对象的线程 ID。
属性/方法	CallContext	CallContext 属性/方法允许您将值（例如，表中某记录的主键）与下一个随后的异步方法调用相关联。方法完成后，此值将传递到完成回调。

如果在用 CreateThreadObject 创建的 COM 类中实现名为 CallInfo 的属性，则可以在方法执行期间访问此属性以检查方法中止和其他信息。该属性将返回一个实现以下接口的对象。

类型	名称	描述
属性	AbortEvent	返回 winapi 事件对象 (CreateEvent)，该对象在内部用于中止调用。
方法	Aborted()	如果调用被终止，返回 .T.，否则返回 .F.。
属性	CallContext	返回与方法调用关联的值
属性	CallId	返回分配给此调用的唯一 ID。

示例

TestFunc 阻止传入的超时

TestFunc2 循环，使用 Sleep api 调用模拟一些工作，并检查方法中止

&& 将该类编译为“多线程COM服务器(dll)”

```

Define Class ExampleObject As Session OlePublic
    DataSession = 2 && 你应该始终在各自的数据工作期中运行每个对象
    CallInfo = .Null. && "魔法"属性

    Function TestFunc(lnTimeOut As Long) As Long
        Declare Sleep In WIN32API Integer
        Sleep(m.lnTimeOut * 1000)
        Return m.lnTimeOut
    Endfunc

    Function TestFunc2() As Long
        Local lnRetVal, xj
        Declare Sleep In WIN32API Integer
        m.lnRetVal = 1
        For m.xj = 1 To 30
            Sleep(500)
            If This.CallInfo.Aborted()
                m.lnRetVal = -1
                Exit
            Endif
        Endfor
        Return m.lnRetVal
    Endfunc
Enddefine

&& 使用上面代码编译好的类
Public loObj, loCallback, xj, lnCallId
m.loCallback = Createobject('ExampleCallback')
m.loObj = CreateThreadObject('YourProject.ExampleObject', m.loCallback, .T.)

For m.xj = 1 To 5
    ? m.loObj.TestFunc(m.xj)
Endfor

&& 中止一个调用
m.lnCallId = m.loObj.TestFunc2()
? m.loObj.AbortCall(m.lnCallId)

&& 将值与下一个调用相关联
m.loObj.CallContext = 'someValue'
? m.loObj.TestFunc(5)
&& 您还可以将 CallContext 用作链接的方法 (它返回“THIS”)
? m.loObj.CallContext('someValue2').TestFunc(5)

```

&& 在“ Object”属性上进行同步调用，这将阻塞，直到该方法完成为止，就像任何普通方法调用一样
` m.loObj.Object.TestFunc(5)

```
Define Class ExampleCallback As Custom
    Function OnCallComplete(callid As Long, result As Variant, callcontext As
Variant) As VOID
        ? callid, result, callcontext
    Endfunc

    Function OnError(callid As Long, callcontext As Variant, errornumber As Long,
errorresource As String, errordescription As String) As VOID
        ? callid, callcontext, errornumber, errorresource, errordescription
    Endfunc
Enddefine
```

参考

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateGuid](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)

[RegisterActiveObject](#)

[RegisterObjectAsFileMoniker](#)

[RevokeActiveObject](#)

[StringFromCLSID](#)

使用的 WinApi

[beginthreadex](#)

[CloseHandle](#)

[CLSIDFromProgID](#)

[CoCreateInstance](#)

[CoGetInterfaceAndReleaseStream](#)

[CoMarshallInterThreadInterfaceInStream](#)

[CoInitializeEx](#)

[CoUninitialize](#)
[CreateEvent](#)
[DispatchMessage](#)
[EnterCriticalSection](#)
[GetLastError](#)
[InitializeCriticalSectionAndSpinCount](#)
[LeaveCriticalSection](#)
[MsgWaitForMultipleObjects](#)
[PeekMessage](#)
[ResetEvent](#)
[SetEvent](#)
[VariantChangeType](#)
[VariantClear](#)
[VariantInit](#)
[VariantCopy](#)
[WaitForSingleObject](#)

GetIUnknown (获取 IUnknown 接口的指针)

返回指向 COM 对象的 IUnknown 接口的指针。

GetIUnknown(cObjectName)

参数

cObjectName

引用 COM 对象作为字符串的变量名称，为其检索 IUnknown 指针。

返回值

指向传入的 COM 对象的 IUnknown 接口的指针（数值）。

参考

[CLSIDFromProgID](#)
[CLSIDFromString](#)

[CreateGuid](#)[CreateThreadObject](#)[IsEqualGuid](#)[ProgIDFromCLSID](#)[RegisterActiveObject](#)[RegisterObjectAsFileMoniker](#)[RevokeActiveObject](#)[StringFromCLSID](#)

IsEqualGuid (比较 GUID)

比较两个 GUID 是否相等。

IsEqualGuid(cGuid1Binary | cGuid1String | nGuid1Pointer, cGuid2Binary | cGuid2String | nGuid2Pointer)

返回值

如果两个值相等，返回.T.；否则返回.F.

参考

[CLSIDFromProgID](#)[CLSIDFromString](#)[CreateGuid](#)[CreateThreadObject](#)[GetIUnknown](#)[ProgIDFromCLSID](#)[RegisterActiveObject](#)[RegisterObjectAsFileMoniker](#)[RevokeActiveObject](#)[StringFromCLSID](#)

使用的 WinApi

[CLSIDFromString](#)

ProgIDFromCLSID (从 CLSID 返回 ProgID)

返回给定 CLSID 的 ProgID。

ProgIDFromCLSID(cCLSID | nCLSIDPointer)

参数

cCLSIDBinary | cCLSIDString | nCLSIDPointer

该函数采用三种不同格式获取 CLSID:

1. CLSID 作为二进制字符串 (16 字节宽)
2. 字符串格式的 CLSID-{002D2B10-C1FA-4193-B134-D86EAECC5250}
3. 以二进制形式指向 CLSID 的内存地址的数字指针

返回值

传入的 CLSID 的可读 ProgID (字符串)。

备注

程序标识符 (ProgID) 是可以与 CLSID 关联的注册表项。像 CLSID 一样, ProgID 标识一个类, 但精度较低, 因为不能保证它是全局唯一的。ProgID 的格式为<Program>.<Component>.<Version>, 用句点分隔并且没有空格, 例如: Word.Document.6。

CLSID 是标识 COM 类对象的全局唯一标识符。如果服务器或容器允许链接到其嵌入式对象, 则需要为每种受支持的对象类注册一个 CLSID。

示例

```
lcCLSID = CLSIDFromProgID('VisualFoxPro.Application')
&& lcCLSID 现在的值是 CLSID 的二进制格式
? ProgIdFromCLSID(lcCLSID) && 转换为可识别格式
&& 返回 VisualFoxpro.Application.9
```

参考

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateGuid](#)

[CreateThreadObject](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[RegisterActiveObject](#)[RegisterObjectAsFileMoniker](#)[RevokeActiveObject](#)[StringFromCLSID](#)

使用的 WinApi

[ProgIDFromCLSID](#)[CoTaskMemFree](#)[CLSIDFromString](#)

RegisterActiveObject (注册活动对象)

将传递的对象注册为 cProgID 中指定的类的活动对象。

注册会导致该对象在 OLE 的运行对象表 (ROT) 中列出，该对象是全局可访问的查找表，用于跟踪计算机上当前正在运行的对象。

RegisterActiveObject(oObjectReference, cProgID)

参数

oObjectReference

对要在 ROT (运行对象表) 中注册的 COM 对象的引用。

cProgID

COM 对象的 ProgID-与 CREATEOBJECT 中使用的字符串相同。

返回值

代表已注册对象的数字句柄。

示例

```

Local yourObject, nObjectHandle
yourObject = Createobject('some.ComObject')
nObjectHandle = RegisterActiveObject(m.yourObject, 'some.ComObject')

&& 现在可以从任何程序使用此对象
Local loComObjectRef
m.loComObjectRef = Getobject(, 'some.ComObject')

```

参考

[CLSIDFromProgID](#)[CLSIDFromString](#)[CreateGuid](#)[CreateThreadObject](#)[GetIUnknown](#)[IsEqualGuid](#)[ProgIDFromCLSID](#)[RegisterObjectAsFileMoniker](#)[RevokeActiveObject](#)[StringFromCLSID](#)

使用的 WinApi

[RegisterActiveObject](#)[CLSIDFromProgID](#)

RegisterObjectAsFileMoniker (注册对象为文件名字对象)

为传递的对象创建文件名字对象。

RegisterObjectAsFileMoniker(oObjectReference, cProgID, cFileName)

参量

cObjectReference

对您要注册为文件名字对象的 COM 对象的引用。

cProgID

COM 对象的 ProgID-与 CREATEOBJECT 中使用的字符串相同。

cFileName

文件名。

返回值

代表已注册对象的数字句柄。

备注

文件名字对象可用于识别存储在其自己文件中的任何对象。文件名字对象是本机文件系统分配给该

文件的路径名的封装。名字对象命名的对象的源必须提供 `IPersistFile` 接口的实现，以支持绑定文件名字对象。文件名字对象可以表示完整路径或相对路径。

示例

```
LOCAL yourObject, hObjectHandle
yourObject = CREATEOBJECT('some.ComObject')
 hObjectHandle= RegisterObjectAsFileMoniker(m.yourObject, 'some.ComObject',
'D:\objec.ref')
? hObjectHandle
```

&& 现在可以从任何程序使用此对象

```
LOCAL loComObjectRef
m.loComObjectRef = GETOBJECT('D:\object.ref')
```

参考

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateGuid](#)

[CreateThreadObject](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)

[RegisterActiveObject](#)

[RevokeActiveObject](#)

[StringFromCLSID](#)

使用的 WinApi

[CLSIDFromProgID](#)

[GetRunningObjectTable](#)

[StgOpenStorage](#)

[StgCreateDocfile](#)

[WriteClassStg](#)

[ReadClassStg](#)

[CreateFileMoniker](#)

[IRunningObjectTable Register](#)

RevokeActiveObject (撤销活动对象)

从运行对象表 (ROT) 中注销对象。

RevokeActiveObject(nObjectHandle)

参数

nObjectHandle

[RegisterActiveObject](#) 返回的数值。

返回值

.T.

参考

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateGuid](#)

[CreateThreadObject](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)

[RegisterActiveObject](#)

[RegisterObjectAsFileMoniker](#)

[StringFromCLSID](#)

使用的 WinApi

[RevokeActiveObject](#)

StringFromCLSID (从 CLSID 返回可识别字符串)

将全局唯一标识符 (GUID) 转换为可识别字符的字符串。

StringFromCLSID(cClsId | nClsIdPointer)

参数

返回值

可读形式的 CLSID (GUID 类)。 例如 {2C256447-3F0D-4CBB-9D12-575BB20CDA0A}

备注

CLSID 是标识 COM 类对象的全局唯一标识符。 如果服务器或容器允许链接到其嵌入式对象，则需要为每种受支持的对象类注册一个 CLSID。

示例

```
lcCLSID = CLSIDFromProgID('VisualFoxPro.Application')
&& lcCLSID 现在的值是 CLSID 的二进制格式
? ProgIdFromCLSID(lcCLSID) && 转换为可识别格式
&& 返回 VisualFoxpro.Application.9
```

参考

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateGuid](#)

[CreateThreadObject](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)

[RegisterActiveObject](#)

[RegisterObjectAsFileMoniker](#)

[RevokeActiveObject](#)

使用的 WinApi

[StringFromGUID2](#)

公共对话框

一些常见的对话框。

GetOpenFileName	创建一个“打开”对话框，该对话框允许用户指定驱动器，目录以及要打开的文件（集）的名称。
---------------------------------	---

GetSaveFileName	创建一个“保存”对话框，该对话框允许用户指定要保存的文件的驱动器，目录和名称。
MessageBoxEx	创建、显示和操作消息框。消息框包含应用程序定义的消息文本和标题，任何图标以及预定义按钮的任意组合。
SHBrowseFolder	显示一个对话框，允许用户选择一个 Shell 文件夹。

GetOpenFileName (打开文件对话框)

创建一个“打开”对话框，该对话框允许用户指定驱动器，目录以及要打开的文件（集）的名称。

GetOpenFileName([nFlags [, cFileFilters [, cFileName [, cInitialDirectory [, cDialogTitle [, nFlagsEx [, cArrayName [, cCallbackFunc]]]]]]])

参数

nFlags (可选，附加)

如果省略 nFlags 参数或传递 0，则使用以下标记：

OFN_EXPLORER | OFN_NOCHANGEDIR | OFN_NODEREferenceLINKS |

OFN_FILEMUSTEXIST | OFN_DONTADDTORECENT

有效标记：

标记	含义
OFN_CREATEPROMPT	如果用户指定了不存在的文件，则此标记将导致对话框提示用户来创建此文件。
OFN_DONTADDTORECENT	Windows 2000/XP：防止文件系统目录中包含用户最近使用的文档的链接中包含选定的文件（链接）。
OFN_EXPLORER	如果要使用旧式用户界面，请省略 OFN_EXPLORER 标志。
OFN_FILEMUSTEXIST	指定用户只能在“文件名”输入字段中键入现有文件的名称。 如果指定了此标记，并且用户输入了无效的名称，则对话框将在消息框中显示警告。 如果指定了此标志，那么还要使用 OFN_PATHMUSTEXIST 标志。 可以在“打开”对话框中使用此标记。“另存为”对话框不能使用它。
OFN_FORCESHOWHIDDEN	Windows 2000 / XP：强制显示系统文件和隐藏文件，从而覆盖用户设置以显示或不显示隐藏文件。但是，不会同时显示标记为系统和隐藏的文件。
OFN_LONGNAMES	对于旧式对话框，此标记使对话框可以使用长文件名。 如果未指定此标志，或者也设置了 OFN_ALLOWMULTISELECT 标志，则旧式对话框将短文件名（8.3 格式）用于带有空格的文件名。 资源管理器样式的对话框将忽略此标志，并始终显示长文件名。
OFN_NOCHANGEDIR	如果用户在搜索文件时更改了目录，则将当前目录恢复为其原始值。 Windows NT 4.0 / 2000 / XP：此标记对 GetOpenFileName 无效。

OFN_NODEREFERENCELINKS	指示对话框返回所选快捷方式 (.LNK) 文件的路径和文件名。 如果未指定此值，则对话框将返回快捷方式引用的文件的路径和文件名。
OFN_NOLONGNAMES	对于旧式对话框，此标记将使对话框使用短文件名 (8.3 格式)。 资源管理器样式的对话框将忽略此标记，并始终显示长文件名。
OFN_NONNETWORKBUTTON	隐藏和禁用“网络”按钮。
OFN_NOREADONLYRETURN	指定返回的文件没有选中“只读”复选框，并且不在受写保护的目录中。
OFN_NOTEFILECREATE	指定在关闭对话框之前不创建文件。 如果应用程序将文件保存在只能创建不能修改的网络共享上，应指定此标志。 当应用程序指定此标志时，库不会检查写保护、磁盘空间、驱动器是否打开或网络保护。 使用此标志的应用程序必须仔细执行文件操作，因为文件关闭后无法重新打开。
OFN_OVERWRITEPROMPT	如果所选文件已存在，则使“另存为”对话框生成一个消息框。 用户必须确认是否覆盖文件。
OFN_PATHMUSTEXIST	指定用户只能输入有效的路径和文件名。 如果使用此标志，并且用户在“文件名”输入了无效的路径和文件名，则对话框会在消息框中显示警告。
OFN_READONLY	在打开对话框时，使“只读”复选框最初处于选中状态。 当关闭对话框时，此标记指示“只读”复选框的状态。

cFileFilters (可选)

默认值: 'All' + CHR (0) + '*.*'

CHR (0) 用于分隔过滤器字符串，例如：“表” + CHR (0) + “*.dbf” + CHR (0) + “数据
库” + CHR (0) + “*.dbc”

您还可以传递一个空字符串以忽略过滤器值。

cFileName (可选)

默认值：空

最初显示在对话框中的文件名。

cInitialDirectory (可选)

默认：空

初始目录。选择初始目录的算法在不同平台上有所不同。

Windows 7：

如果 cInitialDirectory 的值与应用程序第一次使用“打开”或“另存为”对话框时传递的值相同，则将用户最近选择的路径用作初始目录。否则：

- 1、如果 cFileName 包含路径，则该路径是初始目录。
- 2、如果 cInitialDirectory 不为空，则它指定初始目录；如果 cInitialDirectory 为空，并且当前目录包含指定过滤器类型的任何文件，则初始目录为当前目录。
- 3、初始目录是当前用户的个人文件目录。
- 4、初始目录为桌面文件夹。

Windows 2000 / XP / Vista:

如果 cFileName 包含路径，则该路径是初始目录。否则，cInitialDirectory 为指定的初始目录，或者，如果应用程序过去曾经使用过“打开”或“另存为”对话框，则将最近使用的路径选作初始目录。但是，如果应用程序长时间不运行，则其保存的所选路径将被丢弃。

如果 cInitialDirectory 为空，并且当前目录包含指定过滤器类型的任何文件，则初始目录为当前目录。否则，初始目录是当前用户的个人文件目录或桌面文件夹。

cDialogTitle (可选)

默认值：空

放置在对话框标题栏中的字符串。如果此参数为空，则系统使用默认标题（即“另存为”或“打开”）。

nFlagsEx (可选)

默认值：0

如果将 OFN_EX_NOPLACESBAR (1) 作为 nFlagsEx 参数传递，则对话框不会显示位置栏（常用目录的左侧按钮）。

cArrayName (可选)

如果传递数组名，则对话框将允许选择多个文件。

该函数将返回选择的文件数，而不是文件名。

数组中的第一个元素将包含所选文件的路径，每个后续元素将包含文件名（无路径）。

cCallbackFunc (可选)

如果传递 VFP 函数名称，则在对话框中发生事件时将调用该函数。

该功能必须要遵循以下原型：

```
Function OpenFileCallback(lnHwnd, lnControlID, lnCode)
Endfunc
```

有关更多信息，请参见示例项目中的 dialogs.prg。

返回值

如果发生错误，则为 -1。

如果对话框中止，则为 0。

单选对话框为选择的文件名

如果为多选对话框并传递了数组名称，则为所选文件数。

参考

[GetSaveFileName](#)

[MessageBoxEx](#)

[SHBrowseFolder](#)

使用的 WinApi

[GetOpenFileName](#)

[CommDlgExtendedError](#)

GetSaveFileName (保存文件对话框)

创建一个“保存”对话框，该对话框允许用户指定要保存的文件的驱动器，目录和名称。

GetSaveFileName([nFlags [, cFileFilters [, cFileName [, cInitialDirectory [, cDialogTitle [, nFlagsEx [, cCallbackFunc]]]]]])

参数

nFlags (可选，附加)

如果省略 nFlags 参数或传递 0，则使用以下标记：

OFN_EXPLORER | OFN_NOCHANGEDIR

有效标记：

标记	含义
OFN_CREATEPROMPT	如果用户指定了不存在的文件，则此标记将导致对话框提示用户来创建此文件。
OFN_DONTADDTORECENT	Windows 2000/XP：防止文件系统目录中包含用户最近使用的文档的链接中包含选定的文件（链接）。
OFN_EXPLORER	如果要使用旧式用户界面，请省略 OFN_EXPLORER 标志。
OFN_FILEMUSTEXIST	指定用户只能在“文件名”输入字段中键入现有文件的名称。 如果指定了此标记，并且用户输入了无效的名称，则对话框将在消息框中显示警告。 如果指定了此标志，那么还要使用 OFN_PATHMUSTEXIST 标志。

	可以在“打开”对话框中使用此标记。“另存为”对话框不能使用它。
OFN_FORCESHOWHIDDEN	Windows 2000 / XP：强制显示系统文件和隐藏文件，从而覆盖用户设置以显示或不显示隐藏文件。但是，不会同时显示标记为系统和隐藏的文件。
OFN_LONGNAMES	对于旧式对话框，此标记使对话框可以使用长文件名。 如果未指定此标志，或者也设置了 OFN_ALLOWMULTISELECT 标志，则旧式对话框将短文件名（8.3 格式）用于带有空格的文件名。 资源管理器样式的对话框将忽略此标志，并始终显示长文件名。
OFN_NOCHANGEDIR	如果用户在搜索文件时更改了目录，则将当前目录恢复为其原始值。 Windows NT 4.0 / 2000 / XP：此标记对 GetOpenFileName 无效。
OFN_NODEREFERENCELINKS	指示对话框返回所选快捷方式 (.LNK) 文件的路径和文件名。 如果未指定此值，则对话框将返回快捷方式引用的文件的路径和文件名。
OFN_NOLONGNAMES	对于旧式对话框，此标记将使对话框使用短文件名（8.3 格式）。 资源管理器样式的对话框将忽略此标记，并始终显示长文件名。
OFN_NONNETWORKBUTTON	隐藏和禁用“网络”按钮。
OFN_NOREADONLYRETURN	指定返回的文件没有选中“只读”复选框，并且不在受写保护的目录中。
OFN_NOTEFILECREATE	指定在关闭对话框之前不创建文件。 如果应用程序将文件保存在只能创建不能修改的网络共享上，应指定此标志。 当应用程序指定此标志时，库不会检查写保护、磁盘空间、驱动器是否打开或网络保护。 使用此标志的应用程序必须仔细执行文件操作，因为文件关闭后无法重新打开。
OFN_OVERWRITEPROMPT	如果所选文件已存在，则使“另存为”对话框生成一个消息框。 用户必须确认是否覆盖文件。
OFN_PATHMUSTEXIST	指定用户只能输入有效的路径和文件名。 如果使用此标志，并且用户在“文件名”输入了无效的路径和文件名，则对话框会在消息框中显示警告。
OFN_READONLY	在打开对话框时，使“只读”复选框最初处于选中状态。 当关闭对话框时，此标记指示“只读”复选框的状态。

cFileFilters (可选)

默认值: 'All'+ CHR (0) +'*.*'

CHR (0) 用于分隔过滤器字符串，例如：“表” + CHR (0) + “*.dbf” + CHR (0) + “数据
库” + CHR (0) + “*.dbc”

您还可以传递一个空字符串以忽略过滤器值。

cFileName (可选)

默认值: 空

最初显示在对话框中的文件名。

cInitialDirectory (可选)

默认：空

初始目录。选择初始目录的算法在不同平台上有所不同。

Windows 7：

如果 cInitialDirectory 的值与应用程序第一次使用“打开”或“另存为”对话框时传递的值相同，则将用户最近选择的路径用作初始目录。否则：

- 1、如果 cFileName 包含路径，则该路径是初始目录。
- 2、如果 cInitialDirectory 不为空，则它指定初始目录；如果 cInitialDirectory 为空，并且当前目录包含指定过滤器类型的任何文件，则初始目录为当前目录。
- 3、初始目录是当前用户的个人文件目录。
- 4、初始目录为桌面文件夹。

Windows 2000 / XP / Vista：

如果 cFileName 包含路径，则该路径是初始目录。否则，cInitialDirectory 为指定的初始目录，或者，如果应用程序过去曾经使用过“打开”或“另存为”对话框，则将最近使用的路径选作初始目录。但是，如果应用程序长时间不运行，则其保存的所选路径将被丢弃。

如果 cInitialDirectory 为空，并且当前目录包含指定过滤器类型的任何文件，则初始目录为当前目录。否则，初始目录是当前用户的个人文件目录或桌面文件夹。

cDialogTitle (可选)

默认值：空

放置在对话框标题栏中的字符串。如果此参数为空，则系统使用默认标题（即“另存为”或“打开”）。

nFlagsEx (可选)

默认值：0

如果将 OFN_EX_NOPLACESBAR (1) 作为 nFlagsEx 参数传递，则对话框不会显示位置栏（常用目录的左侧按钮）。

cCallbackFunc (可选)

如果传递 VFP 函数名称，则在对话框中发生事件时将调用该函数。

该功能必须要遵循以下原型：

```
Function OpenFileCallback(lnHwnd, lnControlID, lnCode)
Endfunc
```

有关更多信息，请参见示例项目中的 dialogs.prg。

返回值

如果发生错误，则为-1。

如果对话框中止，则为 0。

键入的文件名

参考

[GetOpenFileName](#)

[MessageBoxEx](#)

[SHBrowseFolder](#)

使用的 WinApi

[GetSaveFileName](#)

[CommDlgExtendedError](#)

MessageBoxEx (高级消息框)

创建、显示和操作消息框。 消息框包含应用程序定义的消息文本和标题，任何图标以及预定义按钮的任意组合。

```
MessageBoxEx(cText [, nFlags [, cCaption [, nHwnd [, cnIconResource [, nHInstance [,  
nHelpContextId [, nLanguageId]]]]]])
```

参数

cText

指定显示在对话框中的文本。

注意:

VFP MESSAGEBOX () 函数允许在该参数中传递任何类型的表达式-该函数不能！

传递的值必须是字符串，只需使用 TRANSFORM () 或其他格式化函数从表达式中创建字符串即可。

nFlags (可选)

默认值: 0

nFlags 参数控制消息框的外观和行为。

按钮:

常量	描述
MB_OK	仅 OK 按钮
MB_OKCANCEL	OK 和 Cancel 按钮
MB_ABORTRETRYIGNORE	Abort, Retry, 和 Ignore 按钮
MB_YESNOCANCEL	Yes, No, 和 Cancel 按钮
MB_YESNO	Yes 和 No 按钮
MB_RETRYCANCEL	Retry 和 Cancel 按钮

图标:

常量	描述
MB_ICONSTOP	仅 OK 按钮
MB_ICONQUESTION	OK 和 Cancel 按钮
MB_ICONEXCLAMATION	Abort, Retry, 和 Ignore 按钮
MB_ICONINFORMATION	Yes, No, 和 Cancel 按钮

默认按钮:

常量	描述
MB_DEFBUTTON1	第一个按钮是默认按钮。 除非指定了 MB_DEFBUTTON2, MB_DEFBUTTON3 或 MB_DEFBUTTON4, 否则 MB_DEFBUTTON1 是默认设置。
MB_DEFBUTTON2	第二个按钮是默认按钮。
MB_DEFBUTTON3	第三个按钮是默认按钮。
MB_DEFBUTTON4	第四个按钮是默认按钮。

对话框样式:

常量	描述
MB_APPLMODAL	在继续在由 nWnd 参数标识的窗口中进行工作之前, 用户必须响应该消息框。 但是, 用户可以移至其他线程的窗口并在这些窗口中工作。 根据应用程序中窗口的层次结构, 用户可能能够移动到线程内的其他窗口。 消息框的父级的所有子窗口都将自动禁用, 但弹出窗口不会被禁用。 如果未指定 MB_SYSTEMMODAL 和 MB_TASKMODAL, 则 MB_APPLMODAL 是默认设置。
MB_SYSTEMMODAL	与 MB_APPLMODAL 相同, 除了消息框具有 WS_EX_TOPMOST 样式。 使用 系统模式消息框通知用户需要立即引起注意的严重, 潜在破坏性错误 (例如, 内 存不足)。 除了与 nHwnd 关联的窗口以外, 此标志对用户与窗口交互的能力 没有影响。

MB_TASKMODAL	与 MB_APPLMODAL 相同，不同之处在于如果 nWnd 参数为 0，则属于当前线程的所有顶级窗口都将被禁用。当调用的应用程序或库没有可用的窗口句柄但仍需要阻止输入其他窗口时，请使用此标志（在调用线程中而不挂起其他线程）。
--------------	--

其他各种选择：

常量	描述
MB_DEFAULT_DESKTOP_ONLY	与交互式 Window Station 的桌面相同。有关更多信息，请参见窗口站。 如果当前输入的桌面不是默认桌面，则在用户切换到默认桌面之前，MessageBox 不会返回。
MB_RIGHT	文本右对齐。
MB_RTLREADING	在希伯来语和阿拉伯语系统上，使用从右到左的阅读顺序显示消息和标题文本。
MB_SETFOREGROUND	该消息框将成为前台窗口。在内部，系统为消息框调用 SetForegroundWindow 函数。
MB_TOPMOST	消息框是使用 WS_EX_TOPMOST 窗口样式创建的。
MB_SERVICE_NOTIFICATION	呼叫者是向用户通知事件的服务。即使没有用户登录到计算机，该功能也会在当前活动的桌面上显示一个消息框。终端服务：如果调用线程具有模拟令牌，则该函数会将消息框定向到模拟令牌中指定的会话。如果设置了此标志，则 nWnd 参数必须为 0。这是为了使消息框可以显示在与 nWnd 对应的桌面以外的桌面上。 有关使用此标志的安全注意事项的信息，请参阅 Interactive Services。尤其要注意，此标志可以在锁定的桌面上产生交互式内容，因此应仅用于非常有限的一组场景，例如资源耗尽。

cCaption (可选)

默认值：NULL

消息框标题。如果此成员为 NULL，则使用默认标题“错误”。

nHwnd (可选)

默认值：NULL

所有者窗口的句柄。

如果 nHwnd 为 NULL 或省略，则将当前活动窗口设置为所有者窗口。

如果对话框中没有所有者窗口，也可以指定 0。

cnIconResource (可选)

默认值：NULL

标识自定义图标资源。

此参数可以是字符串或整数资源标识符。

要加载标准的系统定义图标之一，请在 nHInstance 参数中传递 NULL，并指定 [LoadIcon](#) 函数列出的值之一。

nHInstance (可选)

默认值：NULL | 当前可执行文件的提示

模块的句柄，其中包含由 cnIconResource 参数标识的图标资源。

如果省略此参数或传递 NULL 但在 cnIconResource 参数中指定自定义图标，则 nHInstance 默认为当前可执行文件的 HINSTANCE。

注意

可以使用 [GetModuleHandle](#) 或 [LoadLibrary](#) 函数检索模块 (dll) 的 HINSTANCE / HMODULE。

nHelpContextId (可选)

默认值：0

标识帮助上下文。

如果发生帮助事件，该值将传递到所有者窗口。

nLanguageId (可选)

默认： [GetUserDefaultUILanguage](#) ()

显示预定义按钮中包含的文本的语言。

有关受支持的语言标识符的列表，请参阅语言标识符。请注意，Windows 的每个本地化发行版通常仅包含用于少数语言的资源。因此，例如，美国版提供 LANG_ENGLISH，法语版提供 LANG_FRENCH，德语版提供 LANG_GERMAN，日文版提供 LANG_JAPANESE。每个版本都提供 LANG_NEUTRAL。这限制了可与 nLanguageId 参数一起使用的一组值。在指定语言标识符之前，应枚举系统上安装的语言环境。

返回值

如果该函数成功，则返回值为以下菜单项值之一。

如果消息框上有“取消”按钮，则按 ESC 键或选择“取消”按钮时，该函数将返回 IDCANCEL 值。如果消息框没有“取消”按钮，则按 ESC 无效。

如果没有足够的内存来创建消息框，则会引发错误 43。

常量	描述
IDOK	OK 按钮被选择
IDCANCEL	Cancel 按钮被选择

IDABORT	Abort 按钮被选择
IDRETRY	Retry 按钮被选择
IDIGNORE	Ignore 按钮被选择
IDYES	Yes 按钮被选择
IDNO	No 按钮被选择
IDTRYAGAIN	Try Again 按钮被选择
IDCONTINUE	Continue 按钮被选择

参考

[GetOpenFileName](#)

[GetSaveFileName](#)

[SHBrowseFolder](#)

使用的 WinApi

[MessageBoxIndirect](#)

[GetUserDefaultUILanguage](#)

[GetModuleHandle](#)

SHBrowseFolder

显示一个对话框，允许用户选择一个 Shell 文件夹。

SHBrowseFolder(cTitle, nFlags, @cPath [, cRootPath [, cCallback]])

参数

cTitle

在对话框的树视图控件上方显示的字符串。该字符串可用于向用户指定指令。

nFlags

指定对话框选项的标志。此参数可以是 0 或以下值的组合。

标记	含义
BIF_RETURNONLYFSDIRS	仅返回文件系统目录。如果用户选择不是文件系统的文件夹，则“确定”按钮将变灰。 注意“确定”按钮仍为“\\server”项以及“\\server\\share”和目录项保持启用状态。但是，如果用户选择“\\server”项，则将 SHBrowseForForForFor 返回的 PIDL 传递到 SHGetPathFromIDList 将失败。

BIF_DONTGOBELOWDOMAIN	在对话框的树视图控件中不包括域级别以下的网络文件夹。
BIF_STATUSTEXT	在对话框中包括一个状态区域。 回调函数可以通过将消息发送到对话框来设置状态文本。 指定 BIF_NEWDIALOGSTYLE 时不支持此标志。
BIF_RETURNFSANCESTORS	仅返回文件系统祖先。 祖先是位于名称空间层次结构中根文件夹下方的子文件夹。如果用户选择的根文件夹的祖先不属于文件系统，则“确定”按钮将显示为灰色。
BIF_EDITBOX	在浏览对话框中包括一个编辑控件，该控件允许用户键入项目的名称。
BIF_VALIDATE	如果用户在编辑框中输入了无效的名称，则浏览对话框将使用 BFFM_VALIDATEFAILED 消息调用应用程序的 BrowseCallbackProc 。如果未指定 BIF_EDITBOX，则忽略此标志。
BIF_NEWDIALOGSTYLE	使用新的用户界面。 设置此标志将为用户提供一个可以调整大小的较大对话框。该对话框具有几个新功能，包括：对话框内的拖放功能，重新排序，快捷菜单，新文件夹，删除和其他快捷菜单命令。
BIF_BROWSEINCLUDEURLS	浏览对话框可以显示 URL。 还必须设置 BIF_USENWUI 和 BIF_BROWSEINCLUDEFILES 标志。 如果未设置这三个标志中的任何一个，则浏览器对话框将拒绝 URL。 即使设置了这些标志，仅当包含所选项目的文件夹支持 URL 时，浏览对话框才会显示 URL。 当调用文件夹的 IShellFolder :: GetAttributesOf 方法以请求所选项目的属性时，该文件夹必须设置 SFGAO_FOLDER 属性标志。 否则，浏览对话框将不会显示 URL。
BIF_USENWUI	使用新的用户界面，包括一个编辑框。 此标志等效于 BIF_EDITBOX BIF_NEWDIALOGSTYLE。
BIF_UAHINT	与 BIF_NEWDIALOGSTYLE 结合使用时，将使用提示添加到对话框中，代替编辑框。 BIF_EDITBOX 会覆盖此标志。
BIF_NONEWFOLDERBUTTON	不要在浏览对话框中包括“新建文件夹”按钮。
BIF_NOTTRANSLATETARGETS	当所选项目是快捷方式时，返回快捷方式本身而不是其目标的 PIDL。
BIF_BROWSEFORCOMPUTER	只退回计算机。 如果用户选择的不是计算机，则“确定”按钮将显示为灰色。
BIF_BROWSEFORPRINTER	只允许选择打印机。如果用户选择打印机以外的其他任何东西，则“确定”按钮将显示为灰色。 在 Windows XP 和更高版本的系统中，最佳实践是使用 Windows XP 样式的对话框，将对话框的根设置为“打印机和传真”文件夹 (CSIDL_PRINTERES)。
BIF_BROWSEINCLUDEFILES	浏览对话框显示文件和文件夹。
BIF_SHAREABLE	浏览对话框可以显示远程系统上的可共享资源。这适用于希望在本地系统上公开远程共享的应用程序。还必须设置 BIF_NEWDIALOGSTYLE 标志。
BIF_BROWSEFILEJUNCTIONS	Windows 7 及更高版本。允许浏览文件夹连接，例如库或扩展名为.zip 的压缩文件。

@cPath

通过引用将所选文件夹存储在其中的变量。

cRootPath (可选)

默认值: NULL

从中开始浏览的根文件夹的位置。 对话框中仅显示名称空间层次结构中的指定文件夹及其子文件夹。

在这种情况下，此参数可以为空或 NULL，使用名称空间根目录（“桌面”文件夹）。

cCallback (可选)

默认值: 空

事件发生时对话框调用的应用程序定义的函数的名称。

该函数应具有以下原型。

```
Function BrowseCallbackProc(HWnd, uMsg, Lparam)
Endfunc
```

有关参数的说明，请参见 [BrowseCallbackProc](#) 文档。

返回值

如果选择了一个文件夹返回.T.; 如果对话框终止返回.F.

参考

[GetOpenFileName](#)

[GetSaveFileName](#)

[MessageBoxEx](#)

使用的 WinApi

[SHBrowseForFolder](#)

[SHILCreateFromPath](#)

[SHGetPathFromIDList](#)

转换

各种转换功能

Colors2RGB	将提供的红色、绿色和蓝色值转换为 32 位整数。
Double2Str	将双精度 (64 位数值) 值转换为二进制字符串。
Float2Str	将浮点 (32 位数值) 值转换为二进制字符串。

Int64ToStr	将 __int64 (64 位数值) 值转换为请求的格式。
Long2Str	将带符号的 long (32 位数值) 值转换为二进制字符串。
Num2Binary	将 32 位整数转换为可识别的二进制字符串。
PG_ByteA2Str	将 PostgreSQL 的 ByteA 值转换为字符串。
PG_Str2ByteA	将字符串转换为转义的 PostgreSQL ByteA 值。
RGB2Colors	将 RGB 拆分为其组成部分。
Short2Str	将带符号的 short (16 位数值) 值转换为二进制字符串。
Str2Double	将二进制字符串转换为双精度 (64 位数值)。
Str2Float	将二进制字符串转换为浮点数 (32 位数字值)。
Str2Int64	将二进制字符串转换为 __int64 (64 位数值)。
Str2Long	将二进制字符串转换为带符号的 long (32 位数字值)。
Str2Short	将二进制字符串转换为带符号的 short (16 位数值)。
Str2UInt64	将二进制字符串转换为无符号 __int64 (64 位数值)。
Str2ULong	将二进制字符串转换为 unsigned long (32 位数值)。
Str2UShort	将二进制字符串转换为无符号 short (16 位数值)。
UInt64ToStr	将无符号的 __int64 (64 位数值) 值转换为请求的格式。
ULong2Str	将无符号 long (32 位数字值) 值转换为二进制字符串。
UShort2Str	将无符号的 short (16 位数值) 值转换为二进制字符串。
Value2Variant	将任何类型的变量或字段转换为二进制字符串。
Variant2Value	将 Value2Variant 创建的二进制字符串转换为原始值。

Colors2RGB

将提供的红色，绿色和蓝色值转换为 32 位整数。

Colors2RGB(nRed, nGreen, nBlue [, nAlpha])

参数

nRed

红色。 0-255

nGreen

绿色。 0-255

nBlue

蓝色。 0-255

nAlpha (可选)

默认值: 0

颜色的 Alpha 通道 (透明度级别)。

返回值

组合的 RGB 值 (数字)。

参考

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)

[UShort2Str](#)

[Value2Variant](#)

[Variant2Value](#)

Double2Str

将双精度（64 位数值）值转换为二进制字符串。

Double2Str(nValue)

参数

nValue

在 1.7E +/- 308 (15 位数字) 范围内的数值。

返回值

一个二进制字符串，等于传入的 double 值。

参考

[Colors2RGB](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)

[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

Float2Str

将浮点（32 位数值）值转换为二进制字符串。

Float2Str(nValue)

参数

nValue

在 3.4E +/- 38 (7 位数字) 范围内的数值。

返回值

一个二进制字符串，等于传入的浮点值。

参考

[Colors2RGB](#)[Double2Str](#)[Int642Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)[PG_Str2ByteA](#)[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)

[Str2ULong](#)[Str2UShort](#)[UInt64ToStr](#)[ULong2Str](#)[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

Int64ToStr

将_int64 (64 位数值) 值转换为请求的格式。

Int64ToStr(yqcnValue [, nFormat])

参数

cnValue

从 -9,223,372,036,854,775,808 到 9,223,372,036,854,775,807 的数值。

该参数可以 4 种不同的格式传递。

- 1.作为货币值。
- 2.作为 8 字节的 varbinary 字符串。
- 3.作为字符串文字，例如 “ -1234567890123”
- 4.作为正常的 VFP 数值。

nFormat (可选)

默认值：1

指定返回值的格式。以下值之一。

格式	描述
1	8 字节二进制字符串。
2	字符串。

返回值

所请求格式的_int64 值。

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)[PG_Str2ByteA](#)[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)[UInt642Str](#)[ULong2Str](#)[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

Long2Str

将带符号的 long (32 位数值) 值转换为二进制字符串。

Long2Str(nValue)

参数

nValue

在-2,147,483,648 到 2,147,483,647 之间的数值。

返回值

一个二进制字符串，等于传入的 long 值。

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)

[UShort2Str](#)

[Value2Variant](#)

[Variant2Value](#)

Num2Binary

将 32 位整数转换为可识别的二进制字符串。

Num2Binary(nValue)

参数

nValue

有符号或无符号整数值，范围为-2,147,483,648 到 4,294,967,295。

返回值

一个字符串 (32 字节长)，带有传入的整数值的二进制表示形式。

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)

[UShort2Str](#)

[Value2Variant](#)

[Variant2Value](#)

PG[ByteA]2Str

将 PostgreSQL 的 ByteA 值转换为字符串。

PG[ByteA]2Str(cByteA)

参数

返回值

一个字符串。

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG\[Str\]2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)

[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

PG_Str2ByteA

将字符串转换为转义的 PostgreSQL ByteA 值。

PG_Str2ByteA(cString)

参数

返回值

一个字符串。

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Int642Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)

[UInt64ToStr](#)[ULong2Str](#)[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

RGB2Colors

将 RGB 拆分为其组成部分。

RGB2Colors(nRGB, @Red, @Green, @Blue [, @Alpha])

参数

RGB

组合的颜色值。

@Red

通过引用将颜色的红色部分存储到其中的变量。

@Green

通过引用将颜色的绿色部分存储到其中的变量。

@Blue

通过引用将颜色的蓝色部分存储到其中的变量。

@Alpha (可选)

通过引用将颜色的 Alpha 通道存储到其中的变量。

返回值

.T.

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Int64ToStr](#)[Long2Str](#)

[Num2Binary](#)[PG_ByteA2Str](#)[PG_Str2ByteA](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)[UInt64ToStr](#)[ULongToStr](#)[UShortToStr](#)[Value2Variant](#)[Variant2Value](#)

Short2Str

将带符号的 short (16 位数值) 值转换为二进制字符串。

Short2Str(nValue)

参数

nValue

介于-32,768 到 32,767 之间的数值。

返回值

一个二进制字符串，它等于传入的 short 值。

参考

[Colors2RGB](#)

[Double2Str](#)[Float2Str](#)[Int642Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)[PG_Str2ByteA](#)[RGB2Colors](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)[UInt642Str](#)[ULong2Str](#)[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

Str2Double

将二进制字符串转换为双精度（64位数值）。

Str2Double(qValue)

参数

qValue

要转换的二进制字符串。

返回值

数值

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)

[UShort2Str](#)

[Value2Variant](#)

[Variant2Value](#)

Str2Float

将二进制字符串转换为浮点数（32位数字值）。

Str2Float(qValue)

参数

qValue

要转换的二进制字符串。

返回值

数值

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)

[UShort2Str](#)

[Value2Variant](#)

[Variant2Value](#)

Str2Int64

将二进制字符串转换为_int64 (64位数值)。

Str2Int64(qValue [, nFormat])

参数

qValue

要转换的二进制字符串。

nFormat (可选)

默认值：1

指定返回值的格式。以下值之一。

格式	描述
1	货币
2	字符串
3	双精度(double)

注意

如果传递3，则该值可能会被截断！

返回值

以请求的格式转换的值。

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)[UInt64ToStr](#)[ULongToStr](#)[UShortToStr](#)[Value2Variant](#)[Variant2Value](#)

Str2Long

将二进制字符串转换为带符号的 long (32 位数字值)。

Str2Long(qValue)

参数

qValue

要转换的二进制字符串。

返回值

数值。

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Int64ToStr](#)

[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2Short

将二进制字符串转换为带符号的 short (16 位数值)。

Str2Short(qValue)

参数

qValue

要转换的二进制字符串。

返回值

数值。

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Int642Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)[PG_Str2ByteA](#)[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)[UInt642Str](#)[ULong2Str](#)[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

Str2UInt64

将二进制字符串转换为无符号__int64 (64位数值)。

Str2UInt64(qValue [, nFormat])

参数

qValue

要转换的二进制字符串。

nFormat (可选)

指定返回值的格式。以下值之一。

格式	描述
1	货币
2	字符串
3	双精度 (Double)

注意

如果传递 3，则该值可能会被截断！

返回值

以请求的格式转换的值。

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[ULong2Str](#)[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

Str2ULong

将二进制字符串转换为 unsigned long (32 位数值)。

Str2ULong(qValue)

参数

qValue

要转换的二进制字符串。

返回值

数值。

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Int642Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)[PG_Str2ByteA](#)[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)

[Str2Short](#)
[Str2UInt64](#)
[Str2UShort](#)
[UInt64ToStr](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2UShort

将二进制字符串转换为无符号 short (16 位数值)。

Str2UShort(qValue)

参数

qValue

要转换的二进制字符串。

返回值

数值

参考

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int64ToStr](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)

[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)[Str2ULong](#)[UInt64ToStr](#)[ULong2Str](#)[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

UInt64ToStr

将无符号的_int64 (64 位数值) 值转换为请求的格式。

UInt64ToStr(yqcnValue [, nFormat])

参数

cnValue

从 0 到 18,446,744,073,709,551,615 的数值。

该参数可以 4 种不同的格式传递。

1.作为货币值。

2.作为 8 字节的 varbinary 字符串。

3.作为字符串文字，例如 “ -1234567890123”

4.作为正常的 VFP 数值。

nFormat (可选)

默认值：1

指定返回值的格式。以下值之一。

格式	描述
1	8 位二进制字符串.

2	字符串
---	-----

返回值

所请求格式的无符号_int64 值。

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[ULong2Str](#)

[UShort2Str](#)

[Value2Variant](#)

[Variant2Value](#)

ULong2Str

将无符号 long (32 位数字值) 值转换为二进制字符串。

ULong2Str(nValue)

参数

nValue

范围为 0 到 4,294,967,295 的数值。

返回值

一个二进制字符串，它等于传入的无符号 long 值。

参考

[Colors2RGB](#)

[Double2Str](#)

[Float2Str](#)

[Int642Str](#)

[Long2Str](#)

[Num2Binary](#)

[PG_ByteA2Str](#)

[PG_Str2ByteA](#)

[RGB2Colors](#)

[Short2Str](#)

[Str2Double](#)

[Str2Float](#)

[Str2Int64](#)

[Str2Long](#)

[Str2Short](#)

[Str2UInt64](#)

[Str2ULong](#)

[Str2UShort](#)

[UInt642Str](#)

[UShort2Str](#)[Value2Variant](#)[Variant2Value](#)

UShort2Str

将无符号的 short (16 位数值) 值转换为二进制字符串。

UShort2Str(nValue)

参数

nValue

范围为 0 到 65,535 之间的数值。

返回值

一个二进制字符串，它等于传入的无符号 short 值。

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Int642Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)[PG_Str2ByteA](#)[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)

[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)[UInt64ToStr](#)[ULong2Str](#)[Value2Variant](#)[Variant2Value](#)

Value2Variant

将任何类型的变量或字段转换为二进制字符串。

Value2Variant(cValue)

参数

cValue

任何数据类型（包括 NULL）均有效。

返回大小优化的 LCK 值结构以及字符类型的实际数据，可以将其存储到

MEMO NOCPTRANS 或 BLOB 字段，然后使用 Variant2Value 反序列化为原始值。

技巧

SET BLOCKSIZE TO 0-使表更有效地存储。

返回值

字符串

参考

[Colors2RGB](#)[Double2Str](#)[Float2Str](#)[Int642Str](#)[Long2Str](#)[Num2Binary](#)[PG_ByteA2Str](#)

[PG_Str2ByteA](#)[RGB2Colors](#)[Short2Str](#)[Str2Double](#)[Str2Float](#)[Str2Int64](#)[Str2Long](#)[Str2Short](#)[Str2UInt64](#)[Str2ULong](#)[Str2UShort](#)[UInt64ToStr](#)[ULongToStr](#)[UShortToStr](#)[Variant2Value](#)

Variant2Value

将 Value2Variant 创建的二进制字符串转换为原始值。

Variant2Value(variant)

参数

Variant

Value2Variant 产生的二进制变体。

技巧

备注、NOCPTRANS 和 BLOB 字段必须通过引用此函数来传递!

返回值

原始值

参考

[Colors2RGB](#)

[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)

日期和时间

函数提供有关时区的信息，将日期时间值与常见的 C 类型进行相互转换。

ATimeZones	获取有关时区的信息。
Double2DT	将双精度（数字）值转换为日期时间。
DT2Double	将日期时间值转换为双精度（数字）值。
DT2FT	将日期时间值转换为 FILETIME 结构。
DT2ST	将日期时间值转换为 SYSTEMTIME 结构。
DT2Timet	将日期时间值转换为 Time_t (Unix) 时间戳。

DT2UTC	将当前活动时区的日期时间转换为 UTC 日期时间。
FT2DT	将 FILETIME 结构转换为日期时间值。
GetSystemTime	以协调世界时 (UTC) 检索当前系统日期和时间或当地时间。
SetSystemTime	设置当前系统时间和日期。
ST2DT	将 SYSTEMTIME 结构转换为日期时间值。
Timet2DT	将 Time_t (Unix) 时间戳转换为日期时间值。
UTC2DT	将日期时间值从 UTC 转换为本地时区。

ATimeZones

获取有关时区的信息。

ATimeZones(**cArrayName**)

参数

cArrayName

返回时，数组包含来自以下注册表项 “ HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows NT \ CurrentVersion \ Time Zones” 的信息。 另请参见 TIME_ZONE_INFORMATION 结构的说明。

列	内容
1	通用名
2	时区显示名称
3	标准时区名称
4	夏令时时区名称
5	偏差：以分钟为单位的时间转换偏差。 偏差是世界标准时间 (UTC) 与当地时间之间的差异 (以分钟为单位)。 UTC 与当地时间之间的所有转换都基于以下公式：UTC = 当地时间 + 偏差
6	标准偏差：在标准时间期间发生的本地时间转换期间使用的偏差值。 如果未提供标准日期成员的值，则忽略此成员。 此值将添加到偏差成员的值中，以形成标准时间中使用的偏差。在大多数时区中，此成员的值为零。
7	夏令时偏差：夏令时期间发生的本地时间转换中使用的偏差值。 如果未提供 DaylightDate 成员的值，则将忽略此成员。 将此值添加到“偏差”成员的值，以形成夏令时期间使用的偏差。在大多数时区，此成员的值为 -60。
8	标准日期：此操作系统上从夏令时到标准时间转换的日期和本地时间。
9	标准日期 (天)

10	标准日期的星期
11	标准日期的小时
	夏令时时间：在此操作系统上发生从标准时间到夏时制转换的日期和本地时间。
12	夏令时月份
13	夏令时日期(天)
14	夏令时的星期
15	夏令时的小时

返回值

时区数。

参考

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

使用的 WinApi

[RegOpenKeyEx](#)

[RegQueryInfoKey](#)

[RegEnumKeyEx](#)

[RegQueryValueEx](#)

[RegCloseKey](#)

Double2DT

将双精度（数字）值转换为日期时间。

Double2DT(nValue)

参数

nValue

数值

返回值

由传递的 double 值表示的 datetime 值。

参考

[ATimeZones](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

DT2Double

将日期时间值转换为双精度（数字）值。

DT2Double(tTime)

参数

tTime

日期时间值

返回值

代表传入的 Datetime 的双精度值。

参考

[ATimeZones](#)

[Double2DT](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

DT2FT

将日期时间值转换为 [FILETIME](#) 结构。

DT2FT(tTime, nFileTimePointer [, bIsUTC])

参数

tTime

日期时间值

nFileTimePointer

FILETIME 结构的指针

bIsUTC (可选)

默认值: .F.

如果传递 .T., 表示假定传递的日期时间已经是 UTC 时间。

返回值

.T.

参考

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

使用的 WinApi

[LocalFileTimeToFileTime](#)

DT2ST

将日期时间值转换为 [SYSTEMTIME](#) 结构。

DT2ST(tTime, nSystemTimePointer [, bToUTC])

参数

tTime

日期时间值

nSystemTimePointer

SYSTEMTIME 结构的指针

bIsUTC (可选)

默认值: .F.

如果传递 .T., 表示假定传递的日期时间已经是 UTC 时间。

返回值

.T.

参考

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

使用的 WinApi

[FileTimeToSystemTime](#)

[LocalFileTimeToFileTime](#)

DT2Timet

将日期时间值转换为 Time_t (Unix) 时间戳。

DT2Timet([tTime [, bLocalDateTime]])

参数

tTime (可选)

日期时间值。

bLocalDateTime (可选)

默认值= .F.

如果传递的 DateTime 值位于本地时区，则传递.T.; 如果传递的 DateTime 值位于 UTC 中，则传递.F. 或省略参数。

返回值

Time_t (UNIX) 时间戳-一个数字值，指示自 1970/01/01 起经过的秒数。

备注

如果您不向该函数传递任何参数，它将返回当前时间的 UNIX 时间戳。

示例

```
? "当前 UNIX 时间", DT2Timet(), DT2Timet(Datetime(), .T.)
? "当前时区的时间", Datetime(), TIMET2DT(DT2TIMET())
? "UTC 的当前时间", GetSystemTime(.T.), TIMET2DT(DT2TIMET(), .T.)
```

参考

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

使用的 WinApi

[GetSystemTime](#)

DT2UTC

将当前活动时区的日期时间转换为 UTC 日期时间。

DT2UTC(tTime)

参数

tTime

日期时间值

返回值

UTC 日期时间值

参考

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[FT2DT](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

FT2DT

将 [FILETIME](#) 结构转换为日期时间值。

FT2DT(nFileTimePointer [, bToLocalTime])

参数

nFileTimePointer

FILETIME 结构的指针

bToLocalTime

默认值: .F.

如果传递.T.表示 filetime 已经被转换为当前时区。

返回值

日期时间值

参考

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[GetSystemTime](#)

[SetSystemTime](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

GetSystemTime

以协调世界时 (UTC) 检索当前系统日期和时间或当地时间。

GetSystemTime([bUTCTime])

参数

bUTCTime (可选)

默认值: .F.

如果传递.T.表示返回的是 UTC 日期时间。

返回值

日期时间值

参考

[ATimeZones](#)

[Double2DT](#)[DT2Double](#)[DT2FT](#)[DT2ST](#)[DT2Timet](#)[DT2UTC](#)[FT2DT](#)[SetSystemTime](#)[ST2DT](#)[Timet2DT](#)[UTC2DT](#)

使用的 WinApi

[GetLocalTime](#)[GetSystemTime](#)

SetSystemTime

设置当前系统时间和日期。

SetSystemTime(tTime [, bUTCTime])

参数

tTime

日期时间值

bUTCTime (可选)

默认值: .F.

如果传入.T.表示传入的日期时间是 UTC 日期时间。

返回值

.T.

参考

[ATimeZones](#)

[Double2DT](#)[DT2Double](#)[DT2FT](#)[DT2ST](#)[DT2Timet](#)[DT2UTC](#)[FT2DT](#)[GetSystemTime](#)[ST2DT](#)[Timet2DT](#)[UTC2DT](#)

使用的 WinApi

[SetLocalTime](#)[SetSystemTime](#)

ST2DT

将 [SYSTEMTIME](#) 结构转换为日期时间值。

ST2DT(nSystemTimePointer [, bToLocalTime])

参数

nSystemTimePointer

SYSTEMTIME 结构的指针

bToLocalTime (可选)

默认值: .F.

如果传递.T.表示 filetime 已经被转换到当前时区。

返回值

日期时间

参考

[ATimeZones](#)

[Double2DT](#)
[DT2Double](#)
[DT2FT](#)
[DT2ST](#)
[DT2Timet](#)
[DT2UTC](#)
[FT2DT](#)
[GetSystemTime](#)
[SetSystemTime](#)
[Timet2DT](#)
[UTC2DT](#)

Timet2DT

将 Time_t (Unix) 时间戳转换为日期时间值。

Timet2DT(nTimestamp [, bToUTC])

参数

nTimestamp

Time_t (UNIX) 时间戳记-一个数字值，指示自 1970/01/01 起经过的秒数。

bToUTC (可选)

默认值：.F. (当前时区)

如果传递.T.，返回的日期时间是 UTC。

返回值

日期时间

参考

[ATimeZones](#)
[Double2DT](#)
[DT2Double](#)
[DT2FT](#)

[DT2ST](#)[DT2Timet](#)[DT2UTC](#)[FT2DT](#)[GetSystemTime](#)[SetSystemTime](#)[ST2DT](#)[UTC2DT](#)

UTC2DT

将日期时间值从 UTC 转换为本地时区。

UTC2DT(tTime)

参数

tTime

将被转换的日期时间值

返回值

日期时间

参考

[ATimeZones](#)[Double2DT](#)[DT2Double](#)[DT2FT](#)[DT2ST](#)[DT2Timet](#)[DT2UTC](#)[FT2DT](#)[GetSystemTime](#)[SetSystemTime](#)

[ST2DT](#)[Timet2DT](#)

文件系统

检索信息或处理文件和目录。

ADirectoryInfo	将有关目录的信息存储到数组中。
ADirEx	ADIR 函数的扩展。将有关文件的信息存储到数组，游标或为每个文件调用回调函数。
ADriveInfo	将有关当前可用磁盘驱动器的信息存储到数组中。
AFileAttributes	将指定文件或目录的属性存储到数组中。
AFileAttributesEx	将指定文件或目录的扩展属性存储到数组中。
CancelFileChange	停止监视指定目录的文件更改的线程。
CompareFileTimes	比较两个文件的最后写入时间。
CopyFileEx	复制文件时，可以传递一个可选的回调函数，该函数在复制操作进行时会接收状态。
DeleteDirectory	删除包括所有文件和子目录的目录。
DeleteFileEx	删除文件。
FindFileChange	监视目录以在单独的线程中更改文件。
GetFileAttributes	获取文件或目录的属性。
GetFileOwner	获取文件的所有者。
GetFileSize	获取文件的大小。
GetFileTimes	获取文件的创建时间，最后访问时间和最后写入时间。
GetLongPathName	将指定的路径转换为其长格式。
GetShortPathName	获取指定路径的短路径形式。
MoveFileEx	移动文件时，可以传递可选的回调函数，该函数在进行移动操作时接收状态。
SetFileAttributes	设置文件或目录的属性。
SetFileTimes	设置创建、上次访问或上次修改指定文件或目录的日期和时间。

ADirectoryInfo

将有关目录的信息存储到数组中。

ADirectoryInfo(cArrayName, cDirectory)

参数

cArrayname

存储有关目录信息的数组名称。

元素	含义
1	目录内文件的数量，包括子目录内的文件
2	子目录数
3	目录大小-所有文件大小的总和

cDirectory

目录的标准路径名。

返回值

.T.

参考

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)

[MoveFileEx](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[FindFirstFile](#)

[FindNextFile](#)

[FindClose](#)

ADirEx

ADIR 函数的扩展。将有关文件的信息存储到数组，游标或为每个文件调用回调函数。

ADirEx(cArrayName | cCursorName | cCallback, cFileSkeleton [, nFileFilter [, nFlags]])

参数

cArrayName|cCursorName|cCallback

数组，游标或回调函数的名称。

数组的结构：

列	描述	数据类型
1	文件名	C
2	短文件名	C
3	创建时间	T
4	最后访问时间	T
5	最后写入时间	T
6	文件大小	N
7	属性	N

如果在 nFlags 参数中指定 ADIREX_DEST_CURSOR 且游标不存在，则会创建具有默认字段名的游标。

```
Create Cursor theCursorName (filename C(254), dosfilename C(13), creationtime T,
accesstime T, writetime T, filesize N(20,0), fileattribs I)
```

可以自行创建的游标，但是必须与上述字段的字段类型和顺序完全匹配。

如果在 nFlags 参数中指定 ADIREX_DEST_CALLBACK，则回调过程的函数定义必须与此匹配：

Function

```
AdirExCallbackExample(cFileName,cDosFileName,tCreationTime,tLastAccessTime,tLastWriteTime,nFileSize,nFileAttributes)
```

Endfunc

cFileSkeleton

完全限定（驱动器：\路径\通配符）搜索字符串，例如“C:\Winnt*.dll”

nFileFilter (附加)

按属性附加过滤文件，过滤类型由 nFlags 参数控制。

0 或省略->不使用过滤器

例如：

FILE_ATTRIBUTE_DIRECTORY-仅列出目录

还有一个特殊的标志

FILE_ATTRIBUTE_FAKE DIRECTORY-如果您将其指定为“假”目录“.”和“..”也将列出。默认情况下，它们未列出

nFlags (附加)

nFlags 参数控制函数的三种行为：

1. 指定参数 1 的含义

ADIREX_DEST_ARRAY-文件存储到传递的 arrayname 中

ADIREX_DEST_CURSOR-文件存储到传递的游标名称中

ADIREX_DEST_CALLBACK-为每个找到的文件回调传递的函数名

2. 所应用的过滤行为：

默认情况下，至少必须将 nFileFilter 中传递的指定属性之一设置为匹配项：

ADIREX_FILTER_ALL-必须为匹配设置所有属性

ADIREX_FILTER_NONE-无需为匹配设置任何属性

ADIREX_FILTER_EXACT-GetFileAttributes 返回的值必须与提供的过滤器完全相同

3. 文件时间转换：

默认情况下，文件时间会转换为本地时区，

但是如果您指定 ADIREX_UTC_TIMES，则文件时间将以 UTC (GMT) 时间返回。

返回值

文件/目录的数量

参考

[ADirectoryInfo](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributes](#)
[GetFileOwner](#)
[GetFileSize](#)
[GetFileTimes](#)
[GetLongPathName](#)
[GetShortPathName](#)
[MoveFileEx](#)
[SetFileAttributes](#)
[SetFileTimes](#)

使用的 WinApi

[FindFirstFile](#)
[FindNextFile](#)
[FindClose](#)

ADriveInfo

将有关当前可用磁盘驱动器的信息存储到数组中。

ADriveInfo(cArrayName)

参数

cArrayName

数组的名称，作为函数应将信息存储在其中的字符串。

列	含义
---	----

1	盘符
2	驱动器类型(从 GetDriveType 返回)
3	设备号; 如果 DeviceIoControl 失败则返回 -1
4	分区; 如果 DeviceIoControl 失败则返回 -1

返回值

驱动器数量

参考

[ADirectoryInfo](#)

[ADirEx](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)

[MoveFileEx](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[GetLogicalDrives](#)

[GetDriveType](#)

[DeviceIoControl](#)[CreateFile](#)[CloseHandle](#)

AFileAttributes

将指定文件或目录的属性存储到数组中。

AFileAttributes(cArrayName, cFileName[, bUTCTimes])

参数

cArrayName

返回时，数组包含来自 [WIN32_FILE_ATTRIBUTE_DATA](#) 结构的以下信息：

元素	含义	数据类型
1	属性	N
2	文件大小	N
3	创建时间	T
4	最后访问时间	T
5	最后写入时间	T

cFileName

要为其获取属性的文件的名称。

bUTCTimes (可选)

默认值：.F.

如果为.T.，文件时间将以 UTC 时间检索，否则将转换为当前本地时区。

返回值

.T.

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)[CompareFileTimes](#)[CopyFileEx](#)[DeleteDirectory](#)[DeleteFileEx](#)[FindFileChange](#)[GetFileAttributes](#)[GetFileOwner](#)[GetFileSize](#)[GetFileTimes](#)[GetLongPathName](#)[GetShortPathName](#)[MoveFileEx](#)[SetFileAttributes](#)[SetFileTimes](#)

使用的 WinApi

[GetFileAttributesEx](#)

AFileAttributesEx

将指定文件或目录的扩展属性存储到数组中。

AFileAttributesEx(cArrayName, cFileName [, bUTCTimes])

参数

cArrayName

返回时，该数组包含来自 BY_HANDLE_FILE_INFORMATION 结构的以下信息。

元素	含义	数据类型
1	属性	N
2	文件大小	N
3	创建时间	T
4	最后访问时间	T

5	最后写入时间	T
6	指向该文件的链接数。 对于 FAT 文件系统，此成员始终为 0。 对于 NTFS 文件系统，它可以大于 1。	N
7	包含文件的卷的序列号。	N
8	与文件关联的唯一标识符的低位部分。 标识符（上下两部分）和卷序列号唯一地标识一台计算机上的文件。 要确定两个打开的句柄是否表示同一个文件，请组合每个文件的标识符和卷序列号并进行比较。	N
9	与文件关联的唯一标识符的高位部分。	N

cFileName

要为其获取属性的文件的名称。

bUTCTimes (可选)

默认值：.F.

如果为 .T.，文件时间将以 UTC 时间检索，否则将转换为当前本地时区。

返回值

.T.

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)[MoveFileEx](#)[SetFileAttributes](#)[SetFileTimes](#)

使用的 WinApi

[GetFileInformationByHandle](#)[CreateFile](#)[CloseHandle](#)

CancelFileChange

停止监视指定目录的文件更改的线程。

CancelFileChange(nThreadHandle)

参数

nThreadHandle

从 [FindFileChange](#) 返回的线程句柄。

返回值

如果监视目录的线程终止，返回 .T.，否则返回 .F.。

参考

[ADirectoryInfo](#)[ADirEx](#)[ADriveInfo](#)[AFileAttributes](#)[AFileAttributesEx](#)[CompareFileTimes](#)[CopyFileEx](#)[DeleteDirectory](#)[DeleteFileEx](#)[FindFileChange](#)

[GetFileAttributes](#)[GetFileOwner](#)[GetFileSize](#)[GetFileTimes](#)[GetLongPathName](#)[GetShortPathName](#)[MoveFileEx](#)[SetFileAttributes](#)[SetFileTimes](#)

使用的 WinApi

[SetEvent](#)

CompareFileTimes

比较两个文件的最后写入时间。

CompareFileTimes(cFileName1, cFileName2)

参数

cFileName1

带全路径的第一个文件。

cFileName2

带全路径的第二个文件。

返回值

如果最后写入时间相等，返回 0

如果第一个文件的最后写入时间大于第二个文件，则返回 1

如果第一个文件的最后写入时间小于第二个文件，则返回 2

参考

[ADirectoryInfo](#)[ADirEx](#)[ADriveInfo](#)

[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributes](#)
[GetFileOwner](#)
[GetFileSize](#)
[GetFileTimes](#)
[GetLongPathName](#)
[GetShortPathName](#)
[MoveFileEx](#)
[SetFileAttributes](#)
[SetFileTimes](#)

使用的 WinApi

[CreateFile](#)
[GetFileTime](#)

CopyFileEx

复制文件时，可以传递一个可选的回调函数，该函数在复制操作进行时会接收状态。

CopyFileEx(cSourceFile, cDestinationFile [, cCallback [, nShareMode]])

参数

cSourceFile

要复制的文件的标准路径。

cDestinationFile

复制操作目标的标准路径。

cCallback (可选)

在复制操作过程中使用进度信息回调的函数。

该函数应具有以下原型：

```
Function CopyCallback(nBytesCopied, nFileSize, nPercentCopied)
Endfunc
```

nShareMode (可选)

在 [CreateFile](#) API 调用中打开复制文件的共享模式。

默认为 0 (不共享)。

其他可能的值是 FILE_SHARE_READ, FILE_SHARE_WRITE 和 FILE_SHARE_DELETE 或这些值的任意组合。

返回值

如果文件已复制，返回.T.; 如果复制操作被回调程序终止，则返回.F.。

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)

[MoveFileEx](#)[SetFileAttributes](#)[SetFileTimes](#)

使用的 WinApi

[CreateFile](#)[GetFileInformationByHandle](#)[ReadFile](#)[WriteFile](#)

DeleteDirectory

删除包括所有文件和子目录的目录。

DeleteDirectory(cDirectory)

参数

cDirectory

带全路径的将被删除的目录。

返回值

.T.

备注

此函数递归删除整个目录，包括所有文件和子目录，使用时请小心！

参考

[ADirectoryInfo](#)[ADirEx](#)[ADriveInfo](#)[AFileAttributes](#)[AFileAttributesEx](#)[CancelFileChange](#)[CompareFileTimes](#)

[CopyFileEx](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributes](#)
[GetFileOwner](#)
[GetFileSize](#)
[GetFileTimes](#)
[GetLongPathName](#)
[GetShortPathName](#)
[MoveFileEx](#)
[SetFileAttributes](#)
[SetFileTimes](#)

使用的 WinApi

[RemoveDirectory](#)
[DeleteFile](#)
[FindFirstFile](#)
[FindNextFile](#)
[FindClose](#)

DeleteFileEx

删除文件。

DeleteFileEx(cFilename)

参数

cFilename

带全路径的文件名。

返回值

.T.

备注

函数将删除只读属性，从而防止使用 VFP DELETE FILE 命令（如有必要）事先删除。

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)

[MoveFileEx](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[DeleteFile](#)

[GetFileAttributes](#)

[SetFileAttributes](#)

FindFileChange

监视目录以在单独的线程中更改文件。

FindFileChange(cDirectory, bWatchSubtree, nFilter, cCallback)**参数****cDirectory**

要监视的目录的完整路径。 这不能是相对路径或空字符串。

名称不得超过 MAX_PATH (260) 个字符。

bWatchSubtree

如果此参数为.T.，则该函数监视以指定目录为根的目录树； 如果为.F.，则仅监视指定的目录。

nFilter

满足变更通知的过滤条件。 此参数可以是以下一个或多个值。

值	含义
FILE_NOTIFY_CHANGE_FILE_NAME	在监视的目录或子目录中的任何文件名更改都会导致更改通知等待操作返回。 更改包括重命名，创建或删除文件名。
FILE_NOTIFY_CHANGE_DIR_NAME	在监视的目录或子目录中的任何目录名称更改都会导致更改通知等待操作返回。 更改包括创建或删除目录。
FILE_NOTIFY_CHANGE_ATTRIBUTES	被监视的目录或子目录中的任何属性更改都会导致更改通知等待操作返回。
FILE_NOTIFY_CHANGE_SIZE	被监视目录或子目录中任何文件大小的更改都会导致更改通知等待操作返回。仅当文件写入磁盘时，操作系统才会检测到文件大小的变化。对于使用大量缓存的操作系统，仅在充分刷新了缓存后才进行检测。
FILE_NOTIFY_CHANGE_LAST_WRITE	对监视目录或子目录中文件的最后写入时间的任何更改都会导致更改通知等待操作返回。仅当文件写入磁盘时，操作系统才会检测到上次写入时间的更改。对于使用大量缓存的操作系统，仅在充分刷新了缓存后才进行检测。
FILE_NOTIFY_CHANGE_SECURITY	监视的目录或子目录中的任何安全描述符更改都会导致更改通知等待操作。

cCallback

在监视的目录或子目录中出现过滤条件之一时调用的函数。

该功能应具有以下格式：

```
Function FolderChanged
  Lparameters hHandle, cPath, nError
  If nError = 0
    ? cPath + ' 被更改。'
  Else
```

```
? '函数: ' + cPath + " 失败。错误编号: ", nError
Endif
Endfunc
```

返回值

表示监视目录的线程的数字句柄。

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)

[MoveFileEx](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[FindFirstChangeNotification](#)

[FindNextChangeNotification](#)

[CreateThread](#)

[WaitForMultipleObjects](#)

[PostMessage](#)

GetFileAttributes

获取文件或目录的属性。

GetFileAttributes(cFileName)

参数

cFileName

要为其获取属性的文件的名称。

注意：

这可以是相对路径，也可以只是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。

此函数使用 VFP 的搜索路径 (SET PATH)。

返回值

代表文件属性的数值。

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)
[GetLongPathName](#)
[GetShortPathName](#)
[MoveFileEx](#)
[SetFileAttributes](#)
[SetFileTimes](#)

使用的 WinApi

[GetFileAttributes](#)

GetFileOwner

获取文件的所有者。

GetFileOwner(cFileName, @rOwner[, @rDomain[, @rSidType]])

参数

cFileName

要获取其所有者的文件的名称。

注意

这可以是相对路径，也可以只是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。

此函数使用 VFP 的搜索路径 (SET PATH)。

rOwner

对接收文件所有者的变量的引用。

rDomain

对变量的引用，该变量接收文件所有者的域。

rSidType

对接收帐户类型的变量的引用。

该类型是 [SID_NAME_USE](#) 枚举中的值之一。

返回值

.T.

参考

[ADirectoryInfo](#)[ADirEx](#)[ADriveInfo](#)[AFileAttributes](#)[AFileAttributesEx](#)[CancelFileChange](#)[CompareFileTimes](#)[CopyFileEx](#)[DeleteDirectory](#)[DeleteFileEx](#)[FindFileChange](#)[GetFileAttributes](#)[GetFileSize](#)[GetFileTimes](#)[GetLongPathName](#)[GetShortPathName](#)[MoveFileEx](#)[SetFileAttributes](#)[SetFileTimes](#)

使用的 WinApi

[CreateFile](#)[GetKernelObjectSecurity](#)[GetSecurityDescriptorOwner](#)[LookupAccountSid](#)[CloseHandle](#)

GetFileSize

获取文件的大小。

GetFileSize(cFileName)

参数

cFileName

要获取文件大小的文件名称。

注意

这可以是相对路径，也可以只是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。

此函数使用 VFP 的搜索路径 (SET PATH)。

返回值

文件大小。

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)

[MoveFileEx](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[CreateFile](#)

[GetFileSizeEx or](#)

[GetFileSize](#)

[CloseHandle](#)

GetFileTimes

获取文件的创建时间，最后访问时间和最后写入时间。

```
GetFileTimes(cFileName, @rCreationTime [, @rLastAccessTime [, @rLastWriteTime [, bUTCTimes]]])
```

参数

cFileName

要为其获取文件时间的文件或目录的名称。

注意

这可以是相对路径，也可以只是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。

此函数使用 VFP 的搜索路径 (SET PATH)。

rCreationTime

通过引用将文件的创建时间存储到其中的变量。

注意

如果不应该获取，则还可以为任何时间参数传递 NULL。

rLastAccessTime (可选)

通过引用的变量，用于存储文件的最后访问时间。

rLastWriteTime (可选)

通过引用的变量，用于存储文件的最后写入时间。

bUTCTimes (可选)

默认值： .F.

如果.T.，文件时间以 UTC 时间返回。否则它们将转换为本地时区。

返回值

.T.

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetLongPathName](#)

[GetShortPathName](#)

[MoveFileEx](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[GetFileTime](#)

[CreateFile](#)

[CloseHandle](#)

GetLongPathName

将指定的路径转换为其长格式。

GetLongPathName(cFileName)

参数

cFileName

要转换的路径。

注意

名称限制为 MAX_PATH (260) 个字符，它可以是相对路径或仅是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。 函数使用 VFP 的搜索路径 (SET PATH)。

返回值

提供的文件的长路径名。

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetShortPathName](#)

[MoveFileEx](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[GetLongPathName](#)

GetShortPathName

获取指定路径的短路径形式。

GetShortPathName(cFileName)

参数

cFileName

要转换的路径。

注意

名称限制为 MAX_PATH (260) 个字符，它可以是相对路径或仅是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。 函数使用 VFP 的搜索路径 (SET PATH)。

返回值

提供的文件的短路径名。

参考

[ADirectoryInfo](#)[ADirEx](#)[ADriveInfo](#)[AFileAttributes](#)[AFileAttributesEx](#)[CancelFileChange](#)[CompareFileTimes](#)[CopyFileEx](#)[DeleteDirectory](#)[DeleteFileEx](#)[FindFileChange](#)[GetFileAttributes](#)

[GetFileOwner](#)
[GetFileSize](#)
[GetFileTimes](#)
[GetLongPathName](#)
[MoveFileEx](#)
[SetFileAttributes](#)
[SetFileTimes](#)

使用的 WinApi

[GetShortPathName](#)

MoveFileEx

移动文件时，可以传递可选的回调函数，该函数在进行移动操作时接收状态。

MoveFileEx(cSourceFile, cDestinationFile [, cCallback [, nShareMode]])

参数

cSourceFile

要移动的文件的标准路径。

cDestinationFile

移动操作目标的标准路径。

cCallback (可选)

在移动操作期间使用进度信息回调的函数。

该函数应具有以下原型：

```
Function MoveCallback(nBytesCopied, nFileSize, nPercentCopied)
Endfunc
```

nShareMode (可选)

在 [CreateFile](#) API 调用中打开复制文件的共享模式。

默认为 0 (不共享)。

其他可能的值是 FILE_SHARE_READ, FILE_SHARE_WRITE 和 FILE_SHARE_DELETE 或这些值的任意组合。

在此功能中，这些值仅在跨不同卷移动文件时适用。

返回值

如果移动成功返回.T., 否则返回.F.。

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)

[GetFileOwner](#)

[GetFileSize](#)

[GetFileTimes](#)

[GetLongPathName](#)

[GetShortPathName](#)

[SetFileAttributes](#)

[SetFileTimes](#)

使用的 WinApi

[MoveFile](#)

[GetVolumePathName](#)

[PathIsSameRoot](#)

SetFileAttributes

设置文件或目录的属性。

SetFileAttributes(cFileName, nAttributes)

参数

cFileName

需要设置属性的文件名。

注意

这可以是相对路径，也可以只是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。

此函数使用 VFP 的搜索路径 (SET PATH)。

nAttributes

以下值之一或组合。

属性	描述
FILE_ATTRIBUTE_ARCHIVE	作为存档文件或目录的文件或目录。 应用程序通常使用此属性将文件标记为备份或删除。
FILE_ATTRIBUTE_HIDDEN	文件或目录被隐藏。 它不包含在普通目录列表中。
FILE_ATTRIBUTE_NORMAL	没有设置其他属性的文件。 仅当单独使用时，此属性才有效。
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	内容索引服务不对文件或目录建立索引。
FILE_ATTRIBUTE_OFFLINE	文件数据无法立即使用。此属性表示文件数据已物理移动到脱机存储中。 远程存储（分层存储管理软件）使用此属性。应用程序不应随意更改此属性。
FILE_ATTRIBUTE_READONLY	只读文件。应用程序可以读取文件，但不能对其进行写入或删除。 在目录中不使用此属性。
FILE_ATTRIBUTE_SYSTEM	操作系统使用一部分或专门使用的文件或目录。
FILE_ATTRIBUTE_TEMPORARY	用于临时存储的文件。如果有足够的高速缓存可用，文件系统避免将数据写回大容量存储，因为通常，应用程序在句柄关闭后会删除临时文件。在这种情况下，系统可以完全避免写入数据。否则，将在关闭句柄之后写入数据。

返回值

.T.

参考

[ADirectoryInfo](#)[ADirEx](#)[ADriveInfo](#)[AFileAttributes](#)[AFileAttributesEx](#)[CancelFileChange](#)[CompareFileTimes](#)[CopyFileEx](#)[DeleteDirectory](#)[DeleteFileEx](#)[FindFileChange](#)[GetFileAttributes](#)[GetFileOwner](#)[GetFileSize](#)[GetFileTimes](#)[GetLongPathName](#)[GetShortPathName](#)[MoveFileEx](#)[SetFileTimes](#)

使用的 WinApi

[SetFileAttributes](#)

SetFileTimes

设置创建、上次访问或上次修改指定文件或目录的日期和时间。

```
SetFileTimes(cFileName, tCreationTime [, tLastAccessTime [, tLastWriteTime [, bUTCTimes]]])
```

参数

cFileName

要为其更改文件时间的文件或目录的名称。

注意

这可以是相对路径，也可以只是文件名，VFP 函数 FULLPATH 用于获取完全限定的文件名。

此函数使用 VFP 的搜索路径 (SET PATH)。

rCreationTime

通过引用将文件的创建时间存储到其中的变量。

注意

如果不应该获取，则还可以为任何时间参数传递 NULL。

rLastAccessTime (可选)

通过引用的变量，用于存储文件的最后访问时间。

rLastWriteTime (可选)

通过引用的变量，用于存储文件的最后写入时间。

bUTCTimes (可选)

默认值： .F.

如果.T.，文件时间以 UTC 时间返回。否则它们将转换为本地时区。

返回值

.T.

参考

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CompareFileTimes](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributes](#)[GetFileOwner](#)[GetFileSize](#)[GetFileTimes](#)[GetLongPathName](#)[GetShortPathName](#)[MoveFileEx](#)[SetFileAttributes](#)

使用的 WinApi

[SetFileTime](#)[CreateFile](#)[CloseHandle](#)

通用库

通用库函数。

AErrorEx	将有关库中最后发生的错误信息存储到数组中。
FormatMessageEx	检索指定错误号的错误消息。
InitVFP2C32	初始化库。
VFP2CSys	提供对库内部资源的访问或更改全局库行为。

AErrorEx

将有关库中最后发生的错误信息存储到数组中。

AErrorEx(cArrayName)

参数

cArrayName

最后一个 FLL 函数调用的其他错误信息所存储的数组的名称：

注意

如果未发生错误，则 AERROREX () 不会创建数组。

数组的结构：

列	内容
1	上一次 API 调用的错误号 (GetLastError())
2	报告错误的 API 函数的名称
3	错误消息
4	ODBC 状态值 (仅在与 ODBC 相关的调用上设置)

返回值

错误编号

参考

[FormatMessageEx](#)

[InitVFP2C32](#)

[VFP2CSys](#)

FormatMessageEx

检索指定错误号的错误消息。

FormatMessageEx(nLastError [, nLanguageId [, nModuleHandle]])

参数

nLastError

错误编号。

nLanguageId (可选)

默认值：0

请求的消息的语言标识符。

如果在此参数中传递特定的 LANGID，FormatMessageEx 将仅返回该 LANGID 的消息。如果该函数找不到该 LANGID 的消息，则会将 Last-Error 设置为 ERROR_RESOURCE_LANG_NOT_FOUND。如果您输入零，FormatMessageEx 将按以下顺序查找有关 LANGID 的消息：

1. 语言中立
2. 线程 LANGID，基于线程的语言环境值
3. 用户默认的 LANGID，基于用户的默认语言环境值
4. 系统默认的 LANGID，基于系统默认的语言环境值

5. 美国英语

nModuleHandle (可选)

默认值： 0

包含要搜索的消息表的模块句柄。

返回值

错误信息

参考

[AErrorEx](#)

[InitVFP2C32](#)

[VFP2CSys](#)

使用的 WinApi

[FormatMessage](#)

InitVFP2C32

初始化库。

InitVFP2C32(nFlags)

参数

nFlags (附加)

用于指定要初始化库的哪些部分的标志。

可以为以下值之一或组合：

VFP2C_INIT_MARSHAL

VFP2C_INIT_ENUM

VFP2C_INIT_ASYNC

VFP2C_INIT_FILE

VFP2C_INIT_WINSOCK

VFP2C_INIT_ODBC

VFP2C_INIT_PRINT

VFP2C_INIT_NETAPI

VFP2C_INIT_CALLBACK

VFP2C_INIT_SERVICES

VFP2C_INIT_WINDOWS

VFP2C_INIT_RAS

VFP2C_INIT_IPHELPER

VFP2C_INIT_URLMON

VFP2C_INIT_ALL

注意

库中并非所有函数都需要初始化。函数可能需要的初始化标志显示在标头部分中。

返回值

初始化成功返回.T., 否则返回.F.。

示例

初始化库中的所有功能

```
#INCLUDE vfp2c.h
If !InitVFP2C32(VFP2C_INIT_ALL)
  Local laError[1], lnCount, xj, lcError
  lnCount = AERROREX('laError')
  lcError = 'VFP2C32 初始化失败: ' + Chr(13)
  For xj = 1 To lnCount
    lcError = lcError + ;
    '错误编号: ' + Transform(laError[1]) + Chr(13) + ;
    '函数数: ' + laError[2] + Chr(13) + ;
    '错误消息: ' + laError[3] + ''
  Endfor
  && 显示/记录错误并中止程序的初始化 ..
Endif
```

参考

[AErrorEx](#)

[FormatMessageEx](#)

[VFP2CSys](#)

使用的 WinApi

[GetModuleHandle](#)

[LoadLibrary](#)
[HeapCreate](#)
[GetProcAddress](#)
[GetClassInfoEx](#)
[CreateWindowEx](#)
[RegisterClassEx](#)
[InitializeCriticalSection](#)
[WSAStartup](#)

VFP2CSys

提供对库内部资源的访问或更改全局库行为。

VFP2CSys(nOption [, nnewValue])

参数

nOption

选项	返回值
1	vfp2c32.fli 的 HMODULE
2	封送处理库内部使用的堆的句柄
3	用于回调的窗口的句柄
4	用于 Ansi 的默认代码页-Unicode 转换

nnewValue (可选)

4, nnewValue 应该包含一个新的代码页值, 该值将在所有从 Ansi 到 Unicode 字符串的转换中

使用

返回值

返回值取决于传入的值。

参考

[AErrorEx](#)
[FormatMessageEx](#)
[InitVFP2C32](#)

低级文件处理

扩展的低级文件处理功能。

主要功能/增强。

- ◆ 可以处理大于 2GB 的文件。
- ◆ 可以创建/打开文件以进行共享访问 (VFP 本机功能仅允许独占访问)。
- ◆ 用于锁定文件部分的附加功能-FLockFile, FLockFileEx, FUnlockFile 和 FUnlockFileEx。
- ◆ FCreateEx 和 FOpenEx 返回实际的 Windows 文件句柄，您可以将此句柄用于其他 API 函数。

您还可以将未使用 FCreateEx 或 FOpenEx 创建的 API 句柄传递给函数 FWriteEx, FPutEx, FReadEx
ect。 ...

AFHandlesEx	将使用 FCreateEx 或 FOpenEx 创建的所有打开文件句柄存储到数组中。
FChSizeEx	更改使用 FOpenEx 或 FCreateEx 函数打开的文件的大小。
FCloseEx	刷新并关闭使用 FCreateEx 或 FOpenEx 函数打开的文件或通信端口。
FCreateEx	创建并打开一个文件。
FEoFEx	确定文件指针是否位于文件末尾。
FFlushEx	将使用 FCreateEx 或 FOpenEx 打开的文件刷新到磁盘。
FGetsEx	从指定的文件或使用 FOpenEx 或 FCreateEx 打开的通信端口返回一系列字节, 直到遇到回车符为止。
FLockFile	在使用 FCreateEx 或 FOpenEx 打开的文件中锁定字节区域。
FLockFileEx	在使用 FCreateEx 或 FOpenEx 打开的文件中锁定字节区域。
FOpenEx	打开文件。
FPutEx	将字符串, 回车符和换行符写入使用 FCreateEx 或 FOpenEx 打开的文件中。
FReadEx	从使用 FCreateEx 或 FOpenEx 打开的文件中返回指定的字节数。
FSeekEx	在使用 FCreateEx 或 FOpenEx 打开的文件中移动文件指针。
FUnlockFile	解锁使用 FCreateEx 或 FOpenEx 打开的文件中的字节区域。
FUnlockFileEx	解锁使用 FCreateEx 或 FOpenEx 打开的文件中的字节区域。
FWriteEx	将字符串写入使用 FCreateEx 或 FOpenEx 打开的文件中。

AFHandlesEx

将使用 FCreateEx 或 FOpenEx 创建的所有打开文件句柄存储到数组中。

AFHandlesEx(cArrayName)

参数

cArrayName

数组的名称，作为函数应将文件句柄存储在其中的字符串。

数组的结构：

列	内容
1	文件、通信端口、管道等句柄。

返回值

文件句柄

示例

```
&& 当前打开的文件句柄
Local laFileHandles[1]
? AFHandlesEx('laFileHandles')
```

参考

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

FChSizeEx

更改使用 FOpenEx 或 FCreateEx 函数打开的文件的大小。

FChSizeEx(nFileHandle, nNewFileSize)

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

nNewFileSize

新的文件大小。

返回值

新的文件大小。

参考

[AFHandlesEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[SetEndOfFile](#)[SetFilePointer](#)

FCloseEx

刷新并关闭使用 FCreateEx 或 FOpenEx 函数打开的文件或通信端口。

FCloseEx(nHandle)

参数

nHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

返回值

如果文件被关闭，返回.T.; 否则返回 .F. 。

参考

[AFHandlesEx](#)[FChSizeEx](#)[FCreateEx](#)[FEoFEx](#)[FFlushEx](#)[FGetsEx](#)[FLockFile](#)[FLockFileEx](#)[FOpenEx](#)[FPutsEx](#)[FReadEx](#)[FSeekEx](#)[FUnlockFile](#)[FUnlockFileEx](#)[FWriteEx](#)

使用的 WinApi

[CloseHandle](#)

FCreateEx

创建并打开一个文件。

FCreateEx(cFileName [, nAttributesAndFlags [, nAccessMode [, nShareMode]]])

参数

cFileName

指定要创建的文件的名称。 您可以在文件名中包含驱动器指示符和路径。 如果不包含驱动器指示符或路径，则会在默认目录中创建文件。

nAttributesAndFlags (可选, 附加)

默认值: FILE_ATTRIBUTE_NORMAL

文件或设备属性和标志, FILE_ATTRIBUTE_NORMAL 是文件的最常见默认值。

此参数可以包括可用文件属性 (FILE_ATTRIBUTE_*) 的任意组合。 所有其他文件属性将覆盖 FILE_ATTRIBUTE_NORMAL。

此参数还可以包含标志 (FILE_FLAG_*) 的组合, 用于控制文件或设备的缓存行为, 访问模式以及其他特殊用途的标志。 它们可与任何 FILE_ATTRIBUTE_* 值组合。

文件属性:

值	描述
FILE_ATTRIBUTE_READONLY	该文件是只读的。应用程序可以读取文件，但不能写入或删除文件。
FILE_ATTRIBUTE_HIDDEN	该文件被隐藏。不要将其包括在普通目录列表中。
FILE_ATTRIBUTE_SYSTEM	该文件是操作系统的一部分或仅由操作系统使用。
FILE_ATTRIBUTE_ARCHIVE	该文件应被存档。应用程序使用此属性来标记要备份或删除的文件。
FILE_ATTRIBUTE_NORMAL	该文件未设置其他属性。仅当单独使用时，此属性才有效。
FILE_ATTRIBUTE_TEMPORARY	该文件正在用于临时存储。
FILE_ATTRIBUTE_OFFLINE	文件的数据不是立即可用的。此属性指示文件数据物理移动到脱机存储。此属性由远程存储 (分层存储管理软件) 使用。应用程序不应任意更改此属性。
FILE_ATTRIBUTE_ENCRYPTED	文件或目录已加密。对于文件，这意味着文件中的所有数据都已加密。对于目录，这意味着加密是新创建的文件和子目录的默认设置。有关更多信息，请参见文件加密。

	如果还指定了 FILE_ATTRIBUTE_SYSTEM，则此标志无效。
--	--------------------------------------

Flags:

值	描述
FILE_FLAG_BACKUP_SEMANTICS	正在为备份或还原操作打开或创建文件。当进程具有 SE_BACKUP_NAME 和 SE_RESTORE_NAME 特权时，系统将确保调用进程覆盖文件安全检查。 您必须设置此标志以获得目录的句柄。可以将目录句柄而不是文件句柄传递给某些函数。有关更多信息，请参见“备注”部分。
FILE_FLAG_DELETE_ON_CLOSE	关闭所有句柄(包括指定的句柄以及任何其他打开或重复的句柄)后，将立即删除该文件。 如果文件已有打开的句柄，则调用将失败，除非使用 FILE_SHARE_DELETE 共享模式全部打开了它们。 除非指定了 FILE_SHARE_DELETE 共享模式，否则随后对文件的打开请求将失败。
FILE_FLAG_NO_BUFFERING	正在打开文件或设备，而没有用于数据读取和写入的系统缓存。 该标志不影响硬盘缓存或内存映射文件。 使用 FILE_FLAG_NO_BUFFERING 标志成功处理通过 CreateFile 打开的文件有严格的要求。
FILE_FLAG_OPEN_NO_RECALL	已请求文件数据，但应继续将其放在远程存储中。不应将其传输回本地存储。该标志供远程存储系统使用。
FILE_FLAG_POSIX_SEMANTICS	访问将根据 POSIX 规则进行。对于支持该命名的文件系统，这包括允许多个文件的名称(大小写不同)。使用此选项时请格外小心，因为为 MS-DOS 或 16 位 Windows 编写的应用程序可能无法访问使用此标志创建的文件。
FILE_FLAG_RANDOM_ACCESS	访问意图是随机的。系统可以以此为提示来优化文件缓存。 如果文件系统不支持缓存的 I/O 和 FILE_FLAG_NO_BUFFERING，则此标志无效。
FILE_FLAG_SEQUENTIAL_SCAN	访问旨在从头到尾按顺序进行。系统可以使用此提示来优化文件缓存。 如果使用读后(即向后扫描)，则不应使用此标志。 如果文件系统不支持缓存的 I/O 和 FILE_FLAG_NO_BUFFERING。
FILE_FLAG_WRITE_THROUGH	写入操作将不会通过任何中间缓存，而将直接进入磁盘。

注意:

有关属性和标志的更多信息，请参见 [CreateFile](#) 文档。

nAccessMode (可选)

默认值: 0

有效值: 0-只读; 1-只写; 2-读/写

nSharemode (可选)

默认值： 0 (不共享)

可以是以下值之一或组合：

Sharemode	描述
0	如果其他进程请求删除、读取或写入访问，则阻止它们打开文件或设备。
FILE_SHARE_READ	启用文件或设备上的后续打开操作以请求读取访问。 否则，如果其他进程请求读取访问，则无法打开文件或设备。 如果未指定此标志，但已打开文件或设备进行读取访问，则函数将失败。
FILE_SHARE_WRITE	在文件或设备上启用后续打开操作以请求写访问权限。 否则，其他进程如果请求写访问权，则无法打开文件或设备。 如果未指定此标志，但是文件或设备已打开以进行写访问，或者具有具有写访问的文件映射，则该功能将失败。
FILE_SHARE_DELETE	在文件或设备上启用后续打开操作以请求删除访问。 否则，如果其他进程请求删除访问，则它们将无法打开文件或设备。 如果未指定此标志，但是已打开文件或设备以进行删除访问，则该功能将失败。 注意：删除访问权限允许删除和重命名操作。

返回值

创建文件的句柄（数值）值或 -1（如果发生错误）。

备注

nAccessMode 和 nShareMode 参数扩展了 FoxPro 的 FCREATE () 功能。

如果已经存在具有您指定名称的文件，则该文件将被覆盖而不会发出警告。

FCreateEx () 为文件分配文件句柄号，您可以使用它来识别其他 Visual FoxPro 低级文件功能中的文件。 FCreateEx () 在创建文件时返回文件句柄号，或者在无法创建时返回-1。

示例

在当前目录中创建一个只读 INI 文件（如果尚不存在）：

```
Local cIniFile, nFileHandle
cIniFile = [./MyFile.ini]
If Not File(cIniFile) && 文件是否存在
    nFileHandle= FCreateEx(cIniFile) && 如果不存在就用只读模式创建
    If nFileHandle< 0 && 检测打开文件时是否产生错误
        Wait "Cannot open or createINI file" Window Nowait
    Else && 如果没有错误，开始写文件
        FWriteEx(nFileHandle, "[程序信息]" + Chr(13) + "DataDir=" + Addbs(Getenv("APPDATA")) + "MyApp\")

    Endif
    FCloseEx(nFileHandle) && 关闭文件
Endif
```

Modify File (cIniFile) Nowait && 在编辑窗口打开文件

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[CreateFile](#)

FEoFEx

确定文件指针是否位于文件末尾。

FEoFEx(nFileHandle)

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

返回值

如果文件指针位于末尾返回.T., 否则返回.F.。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[SetFilePointer](#)

FFlushEx

将使用 FCreateEx 或 FOpenEx 打开的文件刷新到磁盘。

FFlushEx(nFileHandle)

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

返回值

如果成功，返回.T.; 否则返回 .F.。

示例

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[FlushFileBuffers](#)

FGetsEx

从指定的文件或使用 FOpenEx 或 FCreateEx 打开的通信端口返回一系列字节, 直到遇到回车符为止。

FGetsEx(nFileHandle [, nMaxBytesToRead])

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

nMaxBytesToRead (可选)

默认值: 256

读取的最大字节数。

返回值

读取的数据

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[ReadFile](#)

[SetFilePointer](#)

FLockFile

在使用 FCreateEx 或 FOpenEx 打开的文件中锁定字节区域。

FLockFile(nFileHandle, nLockOffset, nBytesToLock)

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

nLockOffset

字节部分的开始偏移量，用以锁定文件。

nBytesToLock

要锁定在文件中的字节数。

返回值

如果锁定成功返回.T.; 否则返回.F.。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[LockFile](#)

FLockFileEx

在使用 FCreateEx 或 FOpenEx 打开的文件中锁定字节区域。

FLockFileEx(nFileHandle, nLockOffset, nBytesToLock [, nLockMode])**参数****nFileHandle**

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

nLockOffset

字节部分的开始偏移量，用以锁定文件。

nBytesToLock

要锁定在文件中的字节数。

nLockMode (可选, 附加)

默认值：0

Lockmode	描述
LOCKFILE_FAIL_IMMEDIATELY	如果无法获取请求的锁，该函数将立即返回。 如果未指定此标志，则函数等待。
LOCKFILE_EXCLUSIVE_LOCK	该函数请求排他锁。 如果未指定此标志，则该函数请求共享锁。

返回值

如果锁定成功返回.T.; 否则返回.F.。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)[FUnlockFileEx](#)[FWriteEx](#)

使用的 WinApi

[LockFileEx](#)

FOpenEx

打开文件。

FOpenEx(cFileName [, nAttributesAndFlags [, nAccessMode [, nShareMode]]])

参数

cFileName

您要打开的文件的名称。

可以带有相对，绝对或使用 UNC 的路径。

nAttributesAndFlags (可选，附加)

标记如何打开文件。

值	描述
FILE_FLAG_BACKUP_SEMANTICS	正在为备份或还原操作打开或创建文件。当进程具有 SE_BACKUP_NAME 和 SE_RESTORE_NAME 特权时，系统将确保调用进程覆盖文件安全检查。 您必须设置此标志以获得目录的句柄。可以将目录句柄而不是文件句柄传递给某些函数。有关更多信息，请参见“备注”部分。
FILE_FLAG_DELETE_ON_CLOSE	关闭所有句柄(包括指定的句柄以及任何其他打开或重复的句柄)后，将立即删除该文件。 如果文件已有打开的句柄，则调用将失败，除非使用 FILE_SHARE_DELETE 共享模式全部打开了它们。 除非指定了 FILE_SHARE_DELETE 共享模式，否则随后对文件的打开请求将失败。
FILE_FLAG_NO_BUFFERING	正在打开文件或设备，而没有用于数据读取和写入的系统缓存。 该标志不影响硬盘缓存或内存映射文件。 使用 FILE_FLAG_NO_BUFFERING 标志成功处理通过 CreateFile 打开的文件有严格的要求。
FILE_FLAG_OPEN_NO_RECALL	已请求文件数据，但应继续将其放在远程存储中。不应将其传输回本地存储。该标志供远程存储系统使用。

FILE_FLAG_POSIX_SEMANTICS	访问将根据 POSIX 规则进行。对于支持该命名的文件系统，这包括允许多个文件的名称（大小写不同）。使用此选项时请格外小心，因为为 MS-DOS 或 16 位 Windows 编写的应用程序可能无法访问使用此标志创建的文件。
FILE_FLAG_RANDOM_ACCESS	访问意图是随机的。 系统可以以此为提示来优化文件缓存。如果文件系统不支持缓存的 I/O 和 FILE_FLAG_NO_BUFFERING，则此标志无效。
FILE_FLAG_SEQUENTIAL_SCAN	访问旨在从头到尾按顺序进行。系统可以使用此提示来优化文件缓存。 如果使用读后（即向后扫描），则不应使用此标志。 如果文件系统不支持缓存的 I/O 和 FILE_FLAG_NO_BUFFERING。
FILE_FLAG_WRITE_THROUGH	写入操作将不会通过任何中间缓存，而将直接进入磁盘。

注意：

有关属性和标志的更多信息，请参见 [CreateFile](#) 文档。

nAccessMode (可选)

默认值：0

有效值： 0-只读； 1-只写； 2-读/写

nSharemode (可选)

默认值： 0 (不共享)

可以是以下值之一或组合：

Sharemode	描述
0	如果其他进程请求删除、读取或写入访问，则阻止它们打开文件或设备。
FILE_SHARE_READ	启用文件或设备上的后续打开操作以请求读取访问。 否则，如果其他进程请求读取访问，则无法打开文件或设备。 如果未指定此标志，但已打开文件或设备进行读取访问，则函数将失败。
FILE_SHARE_WRITE	在文件或设备上启用后续打开操作以请求写访问权限。 否则，其他进程如果请求写访问权，则无法打开文件或设备。 如果未指定此标志，但是文件或设备已打开以进行写访问，或者具有具有写访问的文件映射，则该功能将失败。
FILE_SHARE_DELETE	在文件或设备上启用后续打开操作以请求删除访问。 否则，如果其他进程请求删除访问，则它们将无法打开文件或设备。 如果未指定此标志，但是已打开文件或设备以进行删除访问，则该功能将失败。 注意：删除访问权限允许删除和重命名操作。

返回值

返回文件的句柄（整数）。 正整数表示文件已成功打开，而 -1 表示失败。

示例

```
Local nFileHandle
nFileHandle=FOPENEX("c:\temp\audit.log")
? nFileHandle
&& 现在你可以使用 nFileHandle 作为其他函数的参数来操作该文件
```

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[CreateFile](#)

F_putsEx

将字符串，回车符和换行符写入使用 FCreateEx 或 FOpenEx 打开的文件中。

F_putsEx(nFileHandle, cData [, nMaxBytesToWrite])

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

cData

要写入文件的内容。

nMaxBytesToWrite (可选)

默认值: LEN (cData)

要写入的最大字节数。

返回值

写入的字节数, 失败返回 0。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[WriteFile](#)

FReadEx

从使用 FCreateEx 或 FOpenEx 打开的文件中返回指定的字节数。

FReadEx(nFileHandle, nBytesToRead)

参数

nFileHandle

指定 FReadEx 从中返回数据的文件的文件句柄号。 您可以从 FOpenEx, FCreateEx 或 Windows api 函数获取 nFileHandle。

nBytesToRead

指定要返回的字节数。 FReadEx 从当前文件指针的位置开始返回数据，并持续直到返回 nBytesToRead 字节或遇到文件末尾为止。

返回值

从文件读取的数据，如果发生错误，则为空字符串。

备注

您不能将本机 VFP 文件句柄功能与 VFP2C32 函数混合和匹配，因为用一个创建的文件句柄不能与另一个函数一起使用。

示例

```
Local nFileHandle, cBytesRead
nFileHandle=FOpenEx("c:\temp\audit.log")
cBytesRead = FReadEx(nFileHandle, 20)
? cBytesRead && 返回文件的第一个 20 字节，因为 FOpenEx()之后未移动读取位置
```

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)[FLockFileEx](#)[FOpenEx](#)[FPutsEx](#)[FSeekEx](#)[FUnlockFile](#)[FUnlockFileEx](#)[FWriteEx](#)

使用的 WinApi

[ReadFile](#)

FSeekEx

在使用 FCreateEx 或 FOpenEx 打开的文件中移动文件指针。

FSeekEx(nFileHandle, nBytesToMove [, nRelativePosition])

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

nBytesToMove

文件指针应移动的字节数。

nRelativePosition (可选)

默认值: FILE_BEGIN

可以是以下值之一:

值	描述
FILE_BEGIN	起始点为零或文件的开头。
FILE_CURRENT	起点是文件指针的当前值。
FILE_END	起点是当前文件结束位置。

返回值

文件新的指针位置。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[SetFilePointer](#)

FUnlockFile

解锁使用 FCreateEx 或 FOpenEx 打开的文件中的字节区域。

FUnlockFile(nFileHandle, nLockOffset, nBytesToLock)

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

nLockOffset

字节部分的开始偏移量以锁定文件。

nBytesToLock

要锁定在文件中的字节数。

返回值

如果成功解锁返回.T.; 否则返回.F.。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFileEx](#)

[FWriteEx](#)

使用的 WinApi

[UnlockFile](#)

FUnlockFileEx

解锁使用 FCreateEx 或 FOpenEx 打开的文件中的字节区域。

FUnlockFileEx(nFileHandle, nLockOffset, nBytesToLock)

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

nLockOffset

字节部分的开始偏移量以锁定文件。

nBytesToLock

要锁定在文件中的字节数。

返回值

如果成功解锁返回.T.; 否则返回.F.。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FWriteEx](#)

使用的 WinApi

[UnlockFileEx](#)

FWriteEx

将字符串写入使用 FCreateEx 或 FOpenEx 打开的文件中。

FWriteEx(nFileHandle, cData [, nMaxBytesToWrite])

参数

nFileHandle

从 FCreateEx, FOpenEx 或 Windows api 函数检索的文件句柄。

cData

要写入文件的内容。

nMaxBytesToWrite (可选)

默认值: LEN (cData)

要写入的最大字节数。

返回值

写入的字节数，失败返回 0。

参考

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

使用的 WinApi

[WriteFile](#)

内存管理

内存分配

AllocHGlobal	从全局堆分配指定数量的字节。
AllocMem	从自定义库堆分配指定数量的字节。
AllocMemTo	从自定义库堆中分配指定数量的字节，并在传递的地址处存储指向分配的内存的指针。
AMemBlocks	将有关从库内部堆分配的所有块的信息存储到数组中。
CompactMem	返回库特定堆中最大已提交自由块的大小。
FreeHGlobal	释放分配给 AllocHGlobal 的内存。
FreeMem	释放由 AllocMem 或 ReAllocMem 函数从库内部堆分配的内存块。
FreePMem	从库内部堆释放传入的指针指向的内存块。
FreeRefArray	释放分配给以 C 样式数组传递的所有内存。
LockHGlobal	锁定全局内存对象，并返回一个指向该对象内存块第一个字节的指针。
ReAllocHGlobal	更改指定的全局内存对象的大小。
ReAllocMem	更改以前由 AllocMem 分配的内存块的大小。
SizeOfMem	检索由 AllocMem 分配的内存块的大小。
UnlockHGlobal	递减与用 AllocHGlobal 分配一个内存对象关联的锁计数。
ValidateMem	验证库内部堆。该函数扫描堆中的所有内存块，并验证堆管理器维护的堆控制结构是否处于一致状态。您也可以使用 ValidateMem 函数来验证单个内存块，而无需检查整个堆的有效性。

AllocHGlobal

从全局堆分配指定数量的字节。

AllocHGlobal(nNumberOfBytes [, nFlags])

参数

nNumberOfBytes

要分配的字节数。

nFlags (可选, 附加)

默认值: GMEM_MOVEABLE | GMEM_ZEROINIT

可以为以下值之一或组合:

Flag	描述
GMEM_FIXED	分配固定内存。返回值是一个指针。

GMEM_MOVEABLE	分配可移动内存。内存块永远不会在物理内存中移动，但是可以在默认堆中移动它们。 返回值是内存对象的句柄。要将句柄转换为指针，请使用 LockHGlobal 函数。 该值不能与 GMEM_FIXED 结合使用。
GMEM_ZEROINIT	Initializes memory contents to zero.

返回值

指向分配内存的指针或句柄，具体取决于 nFlags 参数。

参考

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[GlobalAlloc](#)

AllocMem

从自定义库堆分配指定数量的字节。

AllocMem(nNumberOfBytes)

参数

nNumberOfBytes

要分配的字节数。

返回值

指向已分配内存块的指针（数字），如果发生错误则为 0。

备注

分配的内存被初始化为零。

参考

[AllocHGlobal](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[HeapAlloc](#)

AllocMemTo

从自定义库堆中分配指定数量的字节，并在传递的地址处存储指向分配的内存的指针。

AllocMemTo(nAddress, nNumberOfBytes)

参数

nAddress

存储指向新分配内存的指针的内存地址。

nNumberofbytes

要分配的字节数。

返回值

指向已分配内存的指针（数字），如果发生错误则为 0。

备注

分配的内存被初始化为零。

参考

[AllocHGlobal](#)

[AllocMem](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[HeapAlloc](#)

AMemBlocks

将有关从库内部堆分配的所有块的信息存储到数组中。

AMemBlocks(cArrayName)

参数

cArrayName

返回时，数组包含来自 [PROCESS HEAP ENTRY](#) 结构的以下信息。

列	内容	数据类型
1	指向堆元素的数据部分的指针。	N
2	堆元素的数据部分的大小，以字节为单位。	N
3	系统用于维护有关堆元素的信息的数据大小（以字节为单位）。	N

返回值

内存块数。

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[HeapWalk](#)

CompactMem

返回库特定堆中最大已提交自由块的大小。

CompactMem()

返回值

返回堆中最大已提交自由块的大小。

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[HeapCompact](#)

FreeHGlobal

释放分配给 AllocHGlobal 的内存。

FreeHGlobal(nHandle)

参数

nHandle

从 [AllocHGlobal](#) 返回的内存块的句柄或指针。

返回值

.T.

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[GlobalFree](#)

FreeMem

释放由 AllocMem 或 ReAllocMem 函数从库内部堆分配的内存块。

FreeMem(nAddress)

参数

nAddress

指向由 [AllocMem](#) 分配的内存块的指针。

返回值

.T.

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[HeapFree](#)

FreePMem

从库内部堆释放传入的指针指向的内存块。

FreePMem(nAddress)

参数

nAddress

指向内存位置的指针，释放此位置的指针。

返回值

.T.

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreeRefArray](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[HeapFree](#)

FreeRefArray

释放分配给以 C 样式数组传递的所有内存。

FreeRefArray(nAddress, nStartElement, nElements)

参数

返回值

如果所有元素都释放，返回.T.；否则返回.F.。

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[LockHGlobal](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[HeapFree](#)

LockHGlobal

锁定全局内存对象，并返回一个指向该对象内存块第一个字节的指针。

LockHGlobal(nHandle)

参数

nHandle

从 [AllocHGlobal](#) 返回的内存块的句柄。

注意

必须使用 GMEM_MOVEABLE 标志分配该句柄。

返回值

指向锁定存储区的指针（数字）。

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[ReAllocHGlobal](#)

[ReAllocMem](#)

[SizeOfMem](#)

[UnlockHGlobal](#)

[ValidateMem](#)

使用的 WinApi

[GlobalLock](#)

ReAllocHGlobal

更改指定的全局内存对象的大小。

ReAllocHGlobal(nHandle, nNumberOfBytes [, nFlags])

参数

nHandle

从 [AllocHGlobal](#) 返回的内存块的句柄。

nNumberOfBytes

内存块的新大小。

nFlags (可选)

默认值: GMEM_ZEROINIT

重新分配选项。

如果指定了 GMEM_MODIFY，则该函数仅修改内存对象的属性（忽略 nNumberOfBytes 参数）。否则，该函数将重新分配内存对象。

您可以选择将 GMEM_MODIFY 与以下值组合。

Flag	描述
GMEM_MOVEABLE	分配可移动内存。 如果内存是锁定的 GMEM_MOVEABLE 内存块或 GMEM_FIXED 内存块，并且未指定此标志，则只能在适当位置重新分配内存。

如果此参数未指定 GMEM_MODIFY，则可以使用以下值。

Flag	描述
GMEM_ZEROINIT	如果内存对象的大小增加，则导致将其他内存内容初始化为零。

返回值

指向已分配内存区域的句柄或指针。

备注

如果 ReAllocHGlobal 重新分配可移动对象，则返回值是内存对象的句柄。要将句柄转换为指针，请使用 [LockHGlobal](#) 函数。

如果 ReAllocHGlobal 重新分配一个固定对象，则返回的句柄的值是内存块第一个字节的地址。

如果 ReAllocHGlobal 失败，则不会释放原始内存，并且原始句柄和指针仍然有效。

参考

[AllocHGlobal](#)

[AllocMem](#)

[AllocMemTo](#)

[AMemBlocks](#)

[CompactMem](#)

[FreeHGlobal](#)

[FreeMem](#)

[FreePMem](#)

[FreeRefArray](#)

[LockHGlobal](#)[ReAllocMem](#)[SizeOfMem](#)[UnlockHGlobal](#)[ValidateMem](#)

使用的 WinApi

[GlobalReAlloc](#)

ReAllocMem

更改以前由 AllocMem 分配的内存块的大小。

ReAllocMem(nAddress, nNumberOfBytes)

参数

nAddress

指向从 [AllocMem](#) 返回的内存块的指针。

nNumberOfBytes

内存块的新大小。

返回值

指向请求的存储块的指针 (数字)。

参考

[AllocHGlobal](#)[AllocMem](#)[AllocMemTo](#)[AMemBlocks](#)[CompactMem](#)[FreeHGlobal](#)[FreeMem](#)[FreePMem](#)[FreeRefArray](#)

[LockHGlobal](#)[ReAllocHGlobal](#)[SizeOfMem](#)[UnlockHGlobal](#)[ValidateMem](#)

使用的 WinApi

[HeapReAlloc](#)

SizeOfMem

检索由 AllocMem 分配的内存块的大小。

SizeOfMem(nAddress)

参数

nAddress

指向从 [AllocMem](#) 返回的内存块的指针。

返回值

如果函数成功，则返回值是分配的内存块（以字节为单位）的请求大小。

如果函数失败，返回值为 -1。

如果 nAddress 参数引用不在库堆中的堆分配，则 SizeOfMem 函数的行为未定义。

参考

[AllocHGlobal](#)[AllocMem](#)[AllocMemTo](#)[AMemBlocks](#)[CompactMem](#)[FreeHGlobal](#)[FreeMem](#)[FreePMem](#)[FreeRefArray](#)

[LockHGlobal](#)[ReAllocHGlobal](#)[ReAllocMem](#)[UnlockHGlobal](#)[ValidateMem](#)

使用的 WinApi

[HeapSize](#)

UnlockHGlobal

递减与用 AllocHGlobal 分配一个内存对象关联的锁计数。

UnlockHGlobal(nHandle)

参数

nHandle

从 [AllocHGlobal](#) 返回的内存块的句柄。

返回值

内存句柄可以多次锁定。

如果句柄完全解锁，则函数返回 1;如果仍有锁，则函数返回 2。

参考

[AllocHGlobal](#)[AllocMem](#)[AllocMemTo](#)[AMemBlocks](#)[CompactMem](#)[FreeHGlobal](#)[FreeMem](#)[FreePMem](#)[FreeRefArray](#)[LockHGlobal](#)

[ReAllocHGlobal](#)[ReAllocMem](#)[SizeOfMem](#)[ValidateMem](#)

使用的 WinApi

[GlobalUnlock](#)

ValidateMem

验证库内部堆。该函数扫描堆中的所有内存块，并验证堆管理器维护的堆控制结构是否处于一致状态。

您也可以使用 ValidateMem 函数来验证单个内存块，而无需检查整个堆的有效性。

ValidateMem(nAddress)

参数

nAddress

指向从 [AllocMem](#) 返回的内存块的指针。此参数可以为 0。

如果此参数为 0，则该函数尝试验证整个堆。

如果此参数不为 0，则该函数尝试验证 nAddress 指向的存储块。

它不会尝试验证堆的其余部分。

返回值

如果指定的堆或内存块有效，返回.T.；否则返回.F.。

备注

当您使用 ValidateMem 验证堆中的单个内存块时，它仅检查与该元素有关的控件结构。

[HeapValidate](#) 只能验证分配的内存块。在已释放的内存块上调用 ValidateMem 将返回.F.。因为没有要验证的控制结构。

参考

[AllocHGlobal](#)[AllocMem](#)[AllocMemTo](#)

[AMemBlocks](#)[CompactMem](#)[FreeHGlobal](#)[FreeMem](#)[FreePMem](#)[FreeRefArray](#)[LockHGlobal](#)[ReAllocHGlobal](#)[ReAllocMem](#)[SizeOfMem](#)[UnlockHGlobal](#)

使用的 WinApi

[HeapValidate](#)

杂项

一些有用的函数

AFontInfo	从 Ture Type 字体文件中检索有关包含字体的信息到对象中。
ASplitStr	根据提供的长度将字符串拆分为数组。
Decimals	检索数字的小数位数。
GetCursorPosEx	检索鼠标光标的位置。
Int64 Add	64-bit integers 相加。
Int64 Div	除以 64-bit integers。
Int64 Mod	将一个 64-bit integer 除以另一个，然后返回余数。
Int64 Mul	乘以 64-bit integers.
Int64 Sub	减去 64-bit integers.

AFontInfo

从 Ture Type 字体文件中检索有关包含字体的信息到对象中。

AFontInfo(cFileName [, nLanguageId [, nPlatformId]])

参数

cFileName

TTF (True Type Font) 文件的标准文件名。

nLanguageId (可选)

默认值: [GetSystemDefaultLangID \(\)](#)

检索信息所用的语言。

nPlatformId (可选)

默认值: PLATFORMID_WINDOWS

返回值

一个对象，其属性包含有关字体文件的信息。

示例

参考

[ASplitStr](#)

[Decimals](#)

[GetCursorPosEx](#)

[Int64_Add](#)

[Int64_Div](#)

[Int64_Mod](#)

[Int64_Mul](#)

[Int64_Sub](#)

使用的 WinApi

[GetSystemDefaultLangID](#)

ASplitStr

根据提供的长度将字符串拆分为数组。

ASplitStr(cArrayName, cString, nLen)

参数

cArrayName

字符串应拆分为的数组的名称。

cString

要分割的字符串。

nLen

每个子串的长度。

返回值

子串的数量。

示例

```
Local lcString, lnCount, laSubString[1]
m.lcString = '12345678901234567890123456789012345'
m.lnCount = ASplitStr('laSubString', m.lcString, 10)
Display Memory Like laSubString
```

参考

[AFontInfo](#)

[Decimals](#)

[GetCursorPosEx](#)

[Int64 Add](#)

[Int64 Div](#)

[Int64 Mod](#)

[Int64 Mul](#)

[Int64 Sub](#)

Decimals

检索数字的小数位数。

Decimals(nValue)**参数****nValue**

数值

返回值

返回小数点后的位数。

参考

[AFontInfo](#)

[ASplitStr](#)

[Decimals](#)

[GetCursorPosEx](#)

[Int64 Add](#)

[Int64 Div](#)

[Int64 Mod](#)

[Int64 Mul](#)

[Int64 Sub](#)

GetCursorPosEx

检索鼠标光标的位置。

GetCursorPosEx(@nXCoord, @nYCoord [, bRelative [, nHwnd | cWindowName]])

参数

@nXCoord

通过引用将光标的 X 坐标存储到其中的变量。

@nYCoord

通过引用将光标的 Y 坐标存储到其中的变量。

bRelative (可选)

如果省略 bRetative 或通过.F。 坐标是相对于整个屏幕的，

否则，相对于 nHwnd |cWindowName 传递的指定窗口计算坐标。

nHwnd | cWindowName (可选)

hWnd 或 VFP 窗口的名称。 相对于该窗口计算坐标。

返回值

.T.

参考

[AFontInfo](#)

[ASplitStr](#)

[Decimals](#)

[Int64_Add](#)

[Int64_Div](#)

[Int64_Mod](#)

[Int64_Mul](#)

[Int64_Sub](#)

使用的 WinApi

[GetCursorPos](#)

[ScreenToClient](#)

Int64_Add

64-bit integers 相加。

Int64_Add(cqnBigInt, cqnBigInt2 [, nFormat])

参数

ycqnBigInt

加法的第一个操作数。

参数可以 4 种不同的格式传递。

1.作为货币值。

2.作为 8 位的 varbinary 字符串。

3.作为字符串文字，例如 “ -1234567890123”

4.作为正常的 VFP 数值。

ycqnBigInt2

加法的第二个操作数。

nFormat (可选)

默认值: 1

指定返回值的格式。

可选值为以下值之一。

Format	描述
1	货币
2	字符串
3	8 位字符串

返回值

以要求的格式相加的结果。

备注

该函数引发错误 39 = "数字溢出。如果结果超过 64 位整数的限制 (-9223372036854775808 到 92233720368547775807)，则数据丢失。"

函数返回的货币值应仅与以下函数一起使用：

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int64ToStr](#),
[WriteInt64](#), [WritePInt64](#) 和 [WriteRegistryKey](#)。

VFP 货币数据类型具有隐含的小数点。当您使用 MTON () 转换从 Int_Add、Int64_Sub 等返回的货币值时，返回的数值不是您所期望的，此外，如果值为 -9,223,372,036,854,775,808 则每个 VFP 函数都会引发误差 1988。

由于此函数可以返回无效的货币值 (-9,223,372,036,854,775,808)，您应使用 Q (8) 数据类型而不是 Y 在表中存储 64 位整数。

示例

```
? Int64_Add("-9223372036854775808", 8, 2)
```

参考

[AFontInfo](#)

[ASplitStr](#)

[Decimals](#)

[GetCursorPosEx](#)[Int64_Add](#)[Int64_Div](#)[Int64_Mod](#)[Int64_Mul](#)[Int64_Sub](#)

Int64_Div

除以 64-bit integers。

Int64_Div(yqcnBigInt, yqcnBigInt2 [, nFormat])

参数

yqcnBigInt

除法的第一个操作数。

参数可以 4 种不同的格式传递。

1.作为货币值。

2.作为 8 位的 varbinary 字符串。

3.作为字符串文字，例如 “ -1234567890123”

4.作为正常的 VFP 数值。

yqcnBigInt2

除法的第二个操作数。

nFormat (可选)

默认值：1

指定返回值的格式。

可选值为以下值之一。

Format	描述
1	货币
2	字符串
3	8 位字符串

返回值

以要求的格式相除的结果。

备注

如果将 0 作为除数，则函数引发错误 1307 = "不能除以 0。"。

当您将 -9223372036854775808 除以 -1 时该函数引发错误 39 “数字溢出。数据丢失。”，因为 +9223372036854775808 不在带符号的 64 位整数的可表示范围内。

函数返回的货币值应仅与以下函数一起使用：

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int64ToStr](#),
[WriteInt64](#), [WritePInt64](#) 和 [WriteRegistryKey](#)。

VFP 货币数据类型具有隐含的小数点。当您使用 MTON () 转换从 Int_Add、Int64_Sub 等返回的货币值时，返回的数值不是您所期望的，此外，如果值为 -9,223,372,036,854,775,808 则每个 VFP 函数都会引发误差 1988。

由于此函数可以返回无效的货币值 (-9,223,372,036,854,775,808)，您应使用 Q (8) 数据类型而不是 Y 在表中存储 64 位整数。

参考

[AFontInfo](#)
[ASplitStr](#)
[Decimals](#)
[GetCursorPosEx](#)
[Int64_Add](#)
[Int64_Mod](#)
[Int64_Mul](#)
[Int64_Sub](#)

Int64_Mod

将一个 64-bit integer 除以另一个，然后返回余数。

Int64_Mod(yqcnBigInt, yqcnBigInt2 [, nFormat])

参数

yqcnBigInt

被除数

参数可以 4 种不同的格式传递。

- 1.作为货币值。
- 2.作为 8 位的 varbinary 字符串。
- 3.作为字符串文字，例如 “ -1234567890123”
- 4.作为正常的 VFP 数值。

yqcnBigInt2

除数。

nFormat (可选)

默认值：1

指定返回值的格式。

可选值为以下值之一。

Format	描述
1	货币
2	字符串
3	8 位字符串

返回值

以要求的格式取余数的结果。

备注

如果将 0 作为除数，则函数引发错误 1307 = "不能除以 0。"。

函数返回的货币值应仅与以下函数一起使用：

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#),

[WriteInt64](#), [WritePInt64](#) 和 [WriteRegistryKey](#)。

VFP 货币数据类型具有隐含的小数点。当您使用 MTON () 转换从 Int_Add、Int64_Sub 等返回的货币值时，返回的数值不是您所期望的，此外，如果值为 -9,223,372,036,854,775,808 则每个 VFP 函数都会引发误差 1988。

参考

[AFontInfo](#)

[ASplitStr](#)

[Decimals](#)

[GetCursorPosEx](#)

[Int64_Add](#)

[Int64_Div](#)

[Int64_Mul](#)

[Int64_Sub](#)

Int64_Mul

乘以 64-bit integers.

Int64_Mul(yqcnBigInt, yqcnBigInt2 [, nFormat])

参数

yqcnBigInt

乘法的第一个操作数。

参数可以 4 种不同的格式传递。

- 1.作为货币值。
- 2.作为 8 位的 varbinary 字符串。
- 3.作为字符串文字，例如 “ -1234567890123”
- 4.作为正常的 VFP 数值。

yqcnBigInt2

乘法的第二个操作数。

nFormat (可选)

默认值：1

指定返回值的格式。

可选值为以下值之一。

Format	描述
1	货币
2	字符串
3	8 位字符串

返回值

以要求的格式相乘的结果。

备注

如果结果超过 64 个有符号整数的限制 (-9223372036854775808 到 9223372036854775807)

该函数引发错误 39 “数字溢出。数据丢失。”

函数返回的货币值应仅与以下函数一起使用：

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int64ToStr](#),
[WriteInt64](#), [WritePInt64](#) 和 [WriteRegistryKey](#)。

VFP 货币数据类型具有隐含的小数点。当您使用 MTON () 转换从 Int_Add、Int64_Sub 等返回的货币值时，返回的数值不是您所期望的，此外，如果值为 -9,223,372,036,854,775,808 则每个 VFP 函数都会引发误差 1988。

由于此函数可以返回无效的货币值 (-9,223,372,036,854,775,808)，您应使用 Q (8) 数据类型而不是 Y 在表中存储 64 位整数。

参考

[AFontInfo](#)

[ASplitStr](#)

[Decimals](#)

[GetCursorPosEx](#)

[Int64_Add](#)[Int64_Div](#)[Int64_Mod](#)[Int64_Sub](#)

Int64_Sub

减去 64-bit integers.

Int64_Sub(yqcnBigInt, yqcnBigInt2 [, nFormat])

参数

yqcnBigInt

减法的第一个操作数。

参数可以 4 种不同的格式传递。

- 1.作为货币值。
- 2.作为 8 位的 varbinary 字符串。
- 3.作为字符串文字，例如 “ -1234567890123”
- 4.作为正常的 VFP 数值。

yqcnBigInt2

减法的第二个操作数。

nFormat (可选)

默认值：1

指定返回值的格式。

可选值为以下值之一。

Format	描述
1	货币
2	字符串
3	8 位字符串

返回值

以要求的格式相减的结果。

备注

如果结果超过 64 个有符号整数的限制 (-9223372036854775808 到 9223372036854775807)
该函数引发错误 39 “数字溢出。数据丢失。”

函数返回的货币值应仅与以下函数一起使用：

[Int64 Add](#), [Int64 Sub](#), [Int64 Mul](#), [Int64 Div](#), [Int64 Mod](#), [Int64ToStr](#),
[WriteInt64](#), [WritePInt64](#) 和 [WriteRegistryKey](#)。

VFP 货币数据类型具有隐含的小数点。当您使用 MTON () 转换从 Int_Add、Int64_Sub 等返回的货币值时，返回的数值不是您所期望的，此外，如果值为 -9,223,372,036,854,775,808 则每个 VFP 函数都会引发误差 1988。

由于此函数可以返回无效的货币值 (-9,223,372,036,854,775,808)，您应使用 Q (8) 数据类型而不是 Y 在表中存储 64 位整数。

参考

[AFontInfo](#)
[ASplitStr](#)
[Decimals](#)
[GetCursorPosEx](#)
[Int64 Add](#)
[Int64 Div](#)
[Int64 Mod](#)
[Int64 Mul](#)

网络

网络相关

AbortUrlDownloadToFileEx	中止以 UrlDownloadToFileEx 开始的异步下载。
AlpAddresses	将机器的所有 IP 地址存储到一个数组中。
ANetFiles	将有关服务器上已打开文件的信息存储到数组中。
ANetServers	将在域中可见的所有指定类型的服务器存储到数组中。

GetServerTime	从 Windows 服务器检索当前时间。
IcmpPing	发送 IPv4 ICMP 回显请求 (也称为 ping)，并返回所有回显响应答复。
Ip2MacAddress	返回给定 IP 地址的 MAC 地址。
ResolveHostToIp	解析指定主机名的 IP 地址。
SyncToSNTPServer	与 SNTP 服务器同步系统时间和日期。
UrlDownloadToFileEx	从 Internet 下载资源并将其保存到文件中。

AbortUrlDownloadToFileEx

中止以 UrlDownloadToFileEx 开始的异步下载。

AbortUrlDownloadToFileEx(nThreadHandle)

参数

nThreadHandle

从 [UrlDownloadToFileEx](#) 返回的线程句柄。

返回值

如果执行 UrlDownloadToFile 的线程停止返回.T., 否则返回 .F.。

参考

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

AIPAddresses

将机器的所有 IP 地址存储到一个数组中。

AIpAddresses(cArrayName)**参数****cArrayName**

返回时，数组包含以下信息

列	内容
1	IP 地址

返回值

IP 地址的个数。

参考

[AbortUrlDownloadToFileEx](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[ResolveHostTolp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

使用的 WinApi

[gethostname](#)

[gethostbyname](#)

ANetFiles

将有关服务器上已打开文件的信息存储到数组中。

ANetFiles(cArrayName[, cServer [, cBasepath [, cUser]]])**参数****cArrayName**

返回时，该数组包含来自 [FILE_INFO_3](#) 结构的以下信息。

列	内容	数据类型
1	打开的资源的路径。	C
2	一个字符串，用于指定哪个用户（在具有用户级安全性的服务器上）或哪个计算机（在具有共享级安全性的服务器上）打开了资源。	C
3	打开资源时分配给该资源的标识号。	N
4	与打开的应用程序关联的访问权限。	N
5	文件、设备或管道上的文件锁数量。	N

cServer (可选)

默认值：NULL

要在其上执行功能的远程服务器的 DNS 或 NetBIOS 名称。如果此参数为 NULL，则使用本地计算机。

cBasePath (可选)

默认值：NULL

返回信息的限定符。如果此参数为 NULL，则将枚举所有打开的资源。如果此参数不为 NULL，则该函数仅枚举以 basepath 参数的值为前缀的资源。

cUser (可选)

默认值：NULL

用户名。如果此参数不为 NULL，则其值将用作枚举的限定符。返回的文件仅限于用户名与限定符匹配的文件。如果此参数为 NULL，则不使用用户名限定符。

返回值

文件数量。

参考

[AbortUrlDownloadToFileEx](#)

[AIpAddresses](#)

[ANetServers](#)

[GetServerTime](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[SyncToSNTPServer](#)[UrlDownloadToFileEx](#)

使用的 WinApi

[NetFileEnum](#)

ANetServers

将在域中可见的所有指定类型的服务器存储到数组中。

ANetServers(cArrayName [, nServerType [, nLevel [, cDomain]]])

参数

cArrayName

如果 nLevel = 1, 该数组包含来自 [SERVER_INFO_101](#) 结构的以下信息。

列	内容	数据类型
1	平台 ID	N
2	服务器名	C
3	主版本号和服务器类型。	N
4	操作系统的次要发行版本号。	N
5	计算机正在运行的软件类型。	N
6	描述服务器的注释。	C

如果 nLevel = 2, 该数组包含来自 [SERVER_INFO_100](#) 结构的以下信息。

列	内容	数据类型
1	平台 ID	C
2	服务器名	C

nServerType (可选)

默认值: SV_TYPE_SERVER

过滤服务器条目的值。

有关可选值的列表, 请参阅 [NetServerEnum](#) 文档。

nLevel (可选)

默认值: 1

可用值: 1 或 2。

cDomain (可选)

默认值: NULL

要为其返回服务器列表的域的名称。

该域名必须是 NetBIOS 域名 (例如, microsoft)。

ANetServers 函数不支持 DNS 样式的名称 (例如, microsoft.com)。

返回值

服务器数量

参考

[AbortUrlDownloadToFileEx](#)

[AlpAddresses](#)

[ANetFiles](#)

[GetServerTime](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

使用的 WinApi

[NetServerEnum](#)

GetServerTime

从 Windows 服务器获取当前时间。

GetServerTime(cServer [, nTimeZone])

参数

cServer

从中获取时间的远程服务器的 DNS 或 NetBIOS 名称。

注意

Windows NT: 该字符串必须以[\开头](#)。

nTimeZone (可选)

默认值：1

返回的时间应转换成的时区。

可选以下值之一：

Timezone	含义
1	UTC
2	本机时区
3	服务器时区

返回值

日期时间值。

参考

[AbortUrlDownloadToFileEx](#)

[AlpAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

使用的 WinApi

[NetRemoteTOD](#)

IcmpPing

发送 IPv4 ICMP 回显请求（也称为 ping），并返回所有回显响应答复。

IcmpPing(cArrayName, cHost [, nTTL [, nTOS [, nTimeout [, nDatasize [, bDontFragment [, nPingCount]]]]])

参数

cArrayname

返回时，数组包含以下信息：

如果 ping 成功

列	内容	数据类型
1	返回的 IP 地址	C
2	往返时间（以毫秒为单位）	N
3	状态	N
4	数据已成功取回	L

如果 ping 失败

列	内容
1	空字符串
2	-1
3	-1
4	.F.

cHost

IP 地址或主机名。

例如 “192.168.1.128”，“www.google.com”

nTTL (可选)

默认值：30

网络数据包的生存时间。

nTOS (可选)

默认值=0

服务类型。

nTimeout (可选)

默认：3000 毫秒

超时（以毫秒为单位）。

nDatasize (可选)

默认值：32（字节）

要发送的字节数。

bDontFragment (可选)

默认值：.F.

数据包是否可以分段。

nPingCount (可选)

默认值：1

要发送的 ping 数，每个 ping 结果将存储到数组的新行中。

返回值

执行的 ping 的次数

参考

[AbortUrlDownloadToFileEx](#)

[AlpAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[Ip2MacAddress](#)

[ResolveHostTolp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

使用的 WinApi

[IcmpCreateFile](#)

[IcmpSendEcho](#)

[IcmpCloseHandle](#)

[gethostbyname](#)

Ip2MacAddress

返回给定 IP 地址的 MAC 地址。

Ip2MacAddress(cIP)

参数

cIP

需要检索 MAC 地址的 IP

返回值

MAC 地址 (字符串)

参考

[AbortUrlDownloadToFileEx](#)

[AlpAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[IcmpPing](#)

[ResolveHostTolp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

使用的 WinApi

[SendARP](#)

[inet_addr](#)

ResolveHostTolp

解析指定主机名的 IP 地址。

ResolveHostTolp(cHostname [, cArrayName])

参数

cHostname

主机名，例如 “www.google.com” 或您网络上的服务器

cArrayname (可选)

如果传递数组名，则该函数将返回 IP 地址的数量并将 IP 存储到数组中。

返回值

如果没有将数组名作为第二个参数传递，则返回给 IP (字符串)；如果无法解析主机则返回空字符串

串

如果传递了数组名称，返回则为主机分配 IP 地址数；如果无法解析该主机，则返回 0。

参考

[AbortUrlDownloadToFileEx](#)

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

使用的 WinApi

[gethostbyname](#)

[inet_ntoa](#)

SyncToSNTPServer

与 SNTP 服务器同步系统时间和日期。

SyncToSNTPServer(cServer [, nPort [, nTimeout]])

参数

cServer

SNTP 服务器的主机名。

nPort (可选)

默认值：123

SNTP 服务正在侦听的端口。

nTimeout (可选)

默认值：4000

超时（以毫秒为单位）。

返回值

.T.

参考

[AbortUrlDownloadToFileEx](#)

[AlpAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[UrlDownloadToFileEx](#)

使用的 WinApi

[gethostbyname](#)

[GetSystemTime](#)

[socket](#)

[connect](#)

[send](#)

[recv](#)

[closesocket](#)

[SetSystemTime](#)

UrlDownloadToFileEx

从 Internet 下载资源并将其保存到文件中。

UrlDownloadToFileEx(cUrl, cLocalFile [, cCallback [, bAsynchronous]])

参数

cUrl

要下载的网址，必须包含协议前缀（例如 http://, ftp://）！

cLocalFile

下载位置-有效的本地文件名。

cCallback (可选)

在下载过程中调用以报告进度的回调函数。

回调函数必须具有以下原型：

```
Function UrlProgress
    Lparameters ulProgress, ulProgressMax, ulStatusCode
    /*!* 参数含义
    /*!* ulProgress      已下载字节
    /*!* ulprogressMax   资源大小
    /*!* ulStatusCode    回调状态码
Endfunc
```

ulStatusCode 可以是下表之一：

值	含义
BINDSTATUS_DOWNLOADINGDATA	新的数据块已到达。
BINDSTATUS_FINDINGRESOURCE	网址已解析。
BINDSTATUS_CONNECTING	正在连接
BINDSTATUS_REDIRECTING	重定向。
BINDSTATUS_BEGINDOWNLOADDATA	开始下载。
BINDSTATUS_ENDDOWNLOADDATA	下载已完成，文件现在位于本地 IE 缓存中。
BINDSTATUS_DOWNLOAD_FINISHED	现在可以从提供的文件目标位置读取下载内容。
BINDSTATUS_DOWNLOAD_ABORTED	下载被中止（通过 AbortUrlDownloadToFileEx 或 FLL 发行版）。
BINDSTATUS_USINGCACHEDCOPY	该文件可在 IE 缓存中找到，不会重新下载。
BINDSTATUS_SENDINGREQUEST	请求文件。

bAsynchronous (可选)

默认值 = .F.

如果传递 .T.，通过下载后，下载将在单独的线程中异步执行，该函数立即返回。否则，下载将同步执行，下载完成后函数将返回。

进度回调适用于两种类型。

要中止异步下载，您必须使用从 `UrlDownloadToFileEx` 返回的数字句柄值调用 [AbortUrlDownloadToFileEx](#)。

要中止同步下载，您必须从回调函数返回.F.。

返回值

如果下载是同步的，则该函数从 `UrlDownloadToFile API` 返回结果（返回值 | <0 失败|> 0 成

功)。

如果下载异步，则该函数返回一个内部句柄，该内部句柄表示执行下载的线程。

参考

[AbortUrlDownloadToFileEx](#)

[AlpAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[IcmpPing](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[SyncToSNTPServer](#)

使用的 WinApi

[URLDownloadToFile](#)

ODBC

扩展的 ODBC 函数

ASQDDataSource	将有关已配置的 ODBC 数据源的信息存储到数组中。
ASQLDrivers	将有关已安装的 ODBC 驱动程序的信息存储到数组中。
ChangeSQLDataSource	修改 ODBC 数据源。
CreateSQLDataSource	创建 ODBC 数据源
DeleteSQLDataSource	删除 ODBC 数据源
SQLCancelEx	取消准备好的 SQL 语句。
SQLExecEx	扩展的 SQLEXEC, 将 SQL 语句发送到数据源, 在该数据源中处理该语句。
SQLGetPropEx	扩展的 SQLGETPROP, 检索 ODBC 连接属性。
SQLPrepareEx	扩展的 SQLPREPARE。 准备一个 SQL 语句供 SQLExecEx () 远程执行。
SQLSetPropEx	扩展的 SQLSETPROP, 设置 ODBC 连接属性。

ASQLDatasources

将有关已配置的 ODBC 数据源的信息存储到数组中。

ASQLDatasources(cArrayName [, nDataSourceType])

参数

cArrayName

返回时，数组包含以下信息：

列	内容	数据类型
1	数据源名称。	C
2	数据源描述。	C

nDataSourceType (可选)

默认值：ODBC_BOTH_DSN

可选以下值之一：

DatasourceType	描述
ODBC_BOTH_DSN	列出用户和系统数据源。
ODBC_USER_DSN	仅列出用户数据源。
ODBC_SYSTEM_DSN	仅列出系统数据源。

返回值

数据源的数量。

参考

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLExecEx](#)

[SQLGetPropEx](#)

[SQLPrepareEx](#)

[SQLSetPropEx](#)

使用的 WinApi

[SQLDataSources](#)

ASQLDrivers

将有关已安装的 ODBC 驱动程序的信息存储到数组中。

ASQLDrivers(cArrayName)

参数

cArrayName

返回时，数组包含以下信息：

列	内容	数据类型
1	驱动程序说明。	C
2	驱动程序属性值对的列表，每个条目以 CHR(0)分隔。	C

返回值

驱动的数量。

参考

[ASQLDataSources](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLExecEx](#)

[SQLGetPropEx](#)

[SQLPrepareEx](#)

[SQLSetPropEx](#)

使用的 WinApi

[SQLDrivers](#)

ChangeSQLDataSource

修改 ODBC 数据源。

ChangeSQLDataSource(cDataSource, cDriver, nDataSourceType)

参数

cDataSource

以 CHR(0)分隔的关键字-值对形式的属性列表。

例如：“DSN =PERSONNEL Data” + CHR(0) + “UID = Smith” + CHR(0) + “PWD
=sesame” + CHR(0) + “DATABASE =Personne”

cDriver

驱动程序名称，可以使用 [ASQLDrivers](#) 函数检索驱动程序列表。

nDataSourceType

以下值之一：

DatasourceType	描述
ODBC_USER_DSN	更改用户数据源。
ODBC_SYSTEM_DSN	更改系统数据源。

返回值

如果 [SQLConfigDataSource](#) API 调用成功，返回.T.；否则返回.F.。

参考

[ASQLDatasources](#)

[ASQLDrivers](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLExecEx](#)

[SQLGetPropEx](#)

[SQLPrepareEx](#)

[SQLSetPropEx](#)

使用的 WinApi

[SQLConfigDataSource](#)

CreateSQLDataSource

创建 ODBC 数据源

CreateSQLDataSource(cDataSource, cDriver, nDataSourceType)

参数

cDataSource

以 CHR(0)分隔的关键字-值对形式的属性列表。

例如：“ DSN =PERSONNEL Data” + CHR(0)+ “ UID = Smith” + CHR(0)+ “ PWD
=sesame” + CHR(0) +“ DATABASE =Personne”

cDriver

驱动程序名称，可以使用 [ASQLDrivers](#) 函数检索驱动程序列表。

nDataSourceType

以下值之一：

DatasourceType	描述
ODBC_USER_DSN	创建用户数据源。
ODBC_SYSTEM_DSN	创建系统数据源。

返回值

如果成功返回.T.；否则返回.F.。

参考

[ASQLDatasources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLExecEx](#)

[SQLGetPropEx](#)

[SQLPrepareEx](#)

[SQLSetPropEx](#)

使用的 WinApi

[SQLConfigDataSource](#)

DeleteSQLDataSource

删除 ODBC 数据源

DeleteSQLDataSource(cDataSource, cDriver, nDataSourceType)

参数

cDataSource

数据源名称。

cDriver

驱动程序名称，可以使用 [ASQLDrivers](#) 函数检索驱动程序列表。

nDataSourceType

以下值之一：

DatasourceType	描述
ODBC_USER_DSN	删除用户数据源。
ODBC_SYSTEM_DSN	删除系统数据源。

返回值

如果成功返回.T.；否则返回.F.。

参考

[ASQLDataSources](#)[ASQLDrivers](#)[ChangeSQLDataSource](#)[CreateSQLDataSource](#)[SQLCancelEx](#)[SQLExecEx](#)[SQLGetPropEx](#)[SQLPrepareEx](#)

[SQLSetPropEx](#)

使用的 WinApi

[SQLConfigDataSource](#)

SQLCancelEx

取消准备好的 SQL 语句。

SQLCancelEx(nStatement)

参数

nStatement

从 SQLPrepareEx 返回的准备好的语句句柄。

返回值

.T.

参考

[ASQLDatasources](#)[ASQLDrivers](#)[ChangeSQLDataSource](#)[CreateSQLDataSource](#)[DeleteSQLDataSource](#)[SQLExecEx](#)[SQLGetPropEx](#)[SQLPrepareEx](#)[SQLSetPropEx](#)

使用的 WinApi

[SQLFreeHandle](#)

SQLExecEx

扩展的 SQLEXEC，将 SQL 语句发送到数据源，在该数据源中处理该语句。

SQLExecEx(nConn | nStatement, cSQL [, cCursors | cVariables [, cArrayName [, nFlags [, cCursorSchema [, cParamSchema [, cCallback [, nCallbackInterval]]]]]]])

参数

nConn |nStatement

从 [SQLPrepareEx](#) 返回的有效 ODBC 连接句柄或准备好的语句句柄。

cSQL

要执行的 SQL 语句。

通过将参数括在以 “?” 为前缀的花括号中，可以将参数嵌入到 SQL 语句中，语句中用 “或” 括起来的所有内容均按原样传递。

cCursors| cVariables (可选)

指定一个或多个用逗号分隔的游标名称，例如 “ yourCursor1, yourCursor2”

如果您省略此参数或传递一个空字符串，则会像在 SQLEXEC 中那样自动创建名称，第一个游标被命名为 “sqlresult” ，并为每个附加游标附加结果集编号 (sqlresult2, sqlresult3)。

如果在 nFlags 参数中传递 SQLEXCEEX_DEST_VARIABLE，则该参数将解释为变量或字段名的逗号分隔列表，结果集第一行的数据按顺序存储在该列表中，即列 1 存储变量/字段编号 1。2 存储到 2，依此类推。

cArray (可选)

每个语句存储有返回/删除/更新/插入行数的数组的名称

列	内容
1	游标名称或空字符串（如果未生成结果集）。
2	返回，删除，更新或插入的行数。

这与 VFP9 中引入的行为完全相同。如果在 nFlags 参数中指定 SQLEXEC STORE_INFO，则其他信息将存储在数组的第一列中。

nFlags (可选，附加)

默认值： SQLEXCEEX_DEST_CURSOR | SQLEXCEEX_CALLBACK_PROGRESS |

SQLEXCEEX_CALLBACK_INFO

如果省略 nFlags 参数或传递值 0，则使用默认值。

Flag	描述
SQLEXCEEX_DEST_CURSOR	将结果集存储到一个或多个 VFP 游标中。
SQLEXCEEX_DEST_VARIABLE	将结果集存储到 VFP 变量/字段中。

	如果传递此标志，则仅将结果集第一行的值保存到变量/字段中。
SQL_EXCEX_REUSE_CURSOR	将结果集存储到现有游标中（首先进行 ZAP 处理）。
SQL_EXCEX_NATIVE_SQL	不要尝试解析传递的 SQL 语句中的参数，只需将其照原样传递到数据源即可。 当然，如果设置此标志，则不能嵌入任何参数。
SQL_EXCEX_CALLBACK_PROGRESS	在获取行时回调到 cCallback 参数中传递的函数，如果省略 cCallback 参数，则忽略此标志。
SQL_EXCEX_CALLBACK_INFO	如果从后端返回的其他信息（例如，从存储过程中中的 PRINT 或 RAISERROR 语句），则回调到 cCallback 参数中传递的函数。
SQL_EXCEX_STORE_INFO	将来自后端的其他信息存储到 cArray 参数中传递的数组中。

cCursorSchema:

Schema 的语法类似于 CREATE TABLE / CURSOR 中的列名和类型描述的逗号分隔列表，例如：

```
" myIntCol I, myCharCol C (254) NULL, myBinaryCol C (254) NULL NOCPTRANS, myText
M, myBlob W"
```

注意

不支持长类型名，例如“ Numeric” 或“ Character”！

如果在数据库中查询 Unicode 字符数据，则默认为转换为 Ansi。如果要使 Unicode 数据保持不变，则可以为架构中的列指定 NOCPTRANS。例如“ yourUnicodeSmallText C (254) NOCPTRANS, yourUnicodeLongtext M NOCPTRANS”

如果将 unicode 列转换为类型“ W” 或“ Q”，则数据也将采用 Unicode 格式。

&& 参数名称必须用“（双引号）或'（单引号）引起来

```
lcDocument = Strtofile('someFileWithUnicodeContent.txt')
?SQL_EXCEX(yourConnection, 'UPDATE yourTable SET someField = ?{someVar},
someUnicodeTextField = ?{lcDocument} WHERE yourKey = ?{lnID}', '', '', 0, '', '2
SQL_WCHAR')
```

&& 您还可以通过以数字形式指定sql类型来传递特定于后端的数据类型

```
?SQL_EXCEX(yourConnection, 'UPDATE yourTable SET someField = ?{someVar} WHERE yourKey
= ?{lnID}', '', '1aResult', 0, '', '1 43')
```

注意

仅当您调用存储过程并且后端和 ODBC 驱动程序支持它时，才可以使用命名参数-例如 Microsoft SQL Server。

如果使用命名参数，则必须命名所有参数。

cCallback (可选)

在获取结果集或检索 PRINT 或 RAISERROR 语句或其他附加后端信息时进行回调的函数。

功能原型:

```
Function SQLCallback(lnSet, lnRow, lnRowCount)
```

&& && lnSet: 当前获取的结果集的编号; 如果捕获到严重性较低的PRINT或RAISERROR语句, 则为-1

&& lnRow: 已获取的当前行, 或者lnSet为-1时lnRow包含PRINT或RAISERROR语句的消息/警告

&& lnRowCount: 要整体读取的行数; 如果所使用的ODBC驱动程序不支持返回查询的匹配行, 则为0 / -1; 如果lnSet等于-1, 则此参数没有意义, 并且始终为.F.。

```
Endfunc
```

nCallbackinterval

默认值: 100

在获取行时用于回调到回调过程的时间间隔。

您的回调过程将在获取第一行之前调用一次, 然后在第 n 个记录上调用, 并且在获取最后一行之后调用一次。

返回值

结果集的数量, 或者对于不产生结果集的 SQL 语句为 1。

如果函数失败, 则返回-1; 如果该函数被回调函数中止, 则返回-2。

示例

嵌入参数

```
Local laInfo[1], laError[1]

lcSQL = "SELECT * FROM someTable WHERE someCol = ?{someVar}"
If SQLEXCEX(yourConnection, lcSQL, 'yourResult', 'laInfo')
    ? "游标 " + laInfo[1,1] + " 具有 " + Alltrim(Str(laInfo[1,2])) + " 行。"
Else
    AEROREX('laError')
    Display Memory Like laError
Endif

lcSQL = "UPDATE someTable SET someCol = ?{myCursor.someField} WHERE pk
= ?{myCursor.pk}"
If SQLEXCEX(yourConnection, lcSQL, '', 'laInfo') > 0
    ? laInfo[1,2], " 行已更新。"
Else
    AEROREX('laError')
```

```
Display Memory Like laError
Endif
```

将结果存储到变量而不是游标中。

```
Local lnSum, lnAvg
?SQLExecute(yourConnection, 'SELECT SUM(someCol), AVG(someCol2) FROM yourTable',
'lnSum, lnAvg', '', SQLExecute_DEST_VARIABLE)
```

使用命名参数。

MSSQL T-SQL:

```
CREATE PROCEDURE teststoredproc (@lnPK int = 1, @description varchar(8000) =
'Hello' OUTPUT,
@thirdParam int = 2, @fourthParam
varchar(4000))
)
AS
BEGIN
PRINT '过程开始'
SET @description = REPLICATE('HelloWorld',50)
PRINT '过程结束! '
END
```



```
Local lcMessage, lcMessage2, lcSQL
lcMessage = ''
lcMessage2 = 'Hi SQL-Server'
lcSQL = "{ CALL teststoredproc({@lcMessage},{lcMessage2}) }"
?SQLExecute(yourConnection,lcSQL,'','aResult',0,'','1      "@description",      2
"@fourthParam"  )
```

参考

[ASQLDatasources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLGetPropEx](#)

[SQLPrepareEx](#)

[SQLSetPropEx](#)**使用的 WinApi**[SQLAllocHandle](#)[SQLBindParameter](#)[SQLGetStmtAttr](#)[SQLSetDescRec](#)[SQLSetDescField](#)[SQLExecDirect](#)[SQLParamData](#)[SQLPutData](#)[SQLGetDiagRec](#)[SQLNumResultCols](#)[SQLRowCount](#)[SQLDescribeCol](#)[SQLColAttribute](#)[SQLGetInfo](#)[SQLBindCol](#)[SQLFetch](#)[SQLFreeStmt](#)[SQLMoreResults](#)[SQLFreeHandle](#)

SQLGetPropEx

扩展的 SQLGETPROP，检索 ODBC 连接属性。

SQLGetPropEx(nConnectionHandle | cCursorName, cAttribute, @vAttributeValue)

参数

nConnectionHandle | cCursorName

从 SQLCONNECT / SQLSTRINGCONNECT 返回的有效连接句柄或由远程连接创建的游标的名称

-CURSORGETPROP (" ConnectHandle" , cCursorName) 必须返回有效的连接句柄。

cAttribute

以下值之一：

值	描述
Trace	查询是否启用/禁用了 ODBC 调试跟踪。
TraceFile	查询当前的 ODCB 调试跟踪文件。
Connected	查询基础连接的状态。
IsolationLevel	查询隔离级别。
Perfdata	获取指向 MS SQL Server 性能数据结构的指针。

@vAttributeValue

通过引用将请求的连接属性存储到其中的变量。

返回值

如果成功检索到属性，则为 1；如果通过 AErrorEx 获得其他信息，则为 2。

参考

[ASQLDatasources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLExecEx](#)

[SQLPrepareEx](#)

[SQLSetPropEx](#)

使用的 WinApi

[SQLGetConnectAttr](#)

SQLPrepareEx

扩展的 SQLPREPARE。

准备一个 SQL 语句供 SQLExecEx () 远程执行。

```
SQLPrepareEx(nConn, cSQL [, cCursors | cVariables [, cArrayName [, nFlags [, cCursorSchema [, cParamSchema [, cCallback [, nCallbackinterval]]]]]]])
```

参数

nConn

有效的 ODBC 连接句柄。

cSQL

要执行的 SQL 语句。

通过将参数括在以 “?” 为前缀的花括号中，可以将参数嵌入到 SQL 语句中，语句中用 “或” 括起来的所有内容均按原样传递。

cCursors| cVariables (可选)

指定一个或多个用逗号分隔的游标名称，例如 “ yourCursor1, yourCursor2”

如果您省略此参数或传递一个空字符串，则会像在 SQLEXEC 中那样自动创建名称，第一个游标被命名为 “sqlresult” ，并为每个附加游标附加结果集编号 (sqlresult2, sqlresult3)。

如果在 nFlags 参数中传递 SQLEXCEEX_DEST_VARIABLE，则该参数将解释为变量或字段名的逗号分隔列表，结果集第一行的数据按顺序存储在该列表中，即列 1 存储变量/字段编号 1。2 存储到 2，依此类推。

cArray (可选)

每个语句存储有返回/删除/更新/插入行数的数组的名称

列	内容
1	游标名称或空字符串（如果未生成结果集）。
2	返回，删除，更新或插入的行数。

这与 VFP9 中引入的行为完全相同。如果在 nFlags 参数中指定 SQLEXEC STORE_INFO，则其他信息将存储在数组的第一列中。

nFlags (可选，附加)

默认值： SQLEXCEEX_DEST_CURSOR | SQLEXCEEX_CALLBACK_PROGRESS |

SQLEXCEEX_CALLBACK_INFO

如果省略 nFlags 参数或传递值 0，则使用默认值。

Flag	描述
SQLEXCEEX_DEST_CURSOR	将结果集存储到一个或多个 VFP 游标中。
SQLEXCEEX_DEST_VARIABLE	将结果集存储到 VFP 变量/字段中。

	如果传递此标志，则仅将结果集第一行的值保存到变量/字段中。
SQL_EXCEX_REUSE_CURSOR	将结果集存储到现有游标中（首先进行 ZAP 处理）。
SQL_EXCEX_NATIVE_SQL	不要尝试解析传递的 SQL 语句中的参数，只需将其照原样传递到数据源即可。 当然，如果设置此标志，则不能嵌入任何参数。
SQL_EXCEX_CALLBACK_PROGRESS	在获取行时回调到 cCallback 参数中传递的函数，如果省略 cCallback 参数，则忽略此标志。
SQL_EXCEX_CALLBACK_INFO	如果从后端返回的其他信息（例如，从存储过程中中的 PRINT 或 RAISERROR 语句），则回调到 cCallback 参数中传递的函数。
SQL_EXCEX_STORE_INFO	将来自后端的其他信息存储到 cArray 参数中传递的数组中。

cCursorSchema:

Schema 的语法类似于 CREATE TABLE / CURSOR 中的列名和类型描述的逗号分隔列表，例如：

```
" myIntCol I, myCharCol C (254) NULL, myBinaryCol C (254) NULL NOCPTRANS, myText
M, myBlob W"
```

注意

不支持长类型名，例如“ Numeric” 或“ Character”！

如果在数据库中查询 Unicode 字符数据，则默认为转换为 Ansi。如果要使 Unicode 数据保持不变，则可以为架构中的列指定 NOCPTRANS。例如“ yourUnicodeSmallText C (254) NOCPTRANS, yourUnicodeLongtext M NOCPTRANS”

如果将 unicode 列转换为类型“ W” 或“ Q”，则数据也将采用 Unicode 格式。

&& 参数名称必须用“（双引号）或'（单引号）引起来

```
lcDocument = Strtofile('someFileWithUnicodeContent.txt')
?SQL_EXCEX(yourConnection, 'UPDATE yourTable SET someField = ?{someVar},
someUnicodeTextField = ?{lcDocument} WHERE yourKey = ?{lnID}', '', '', 0, '', '2
SQL_WCHAR')
```

&& 您还可以通过以数字形式指定sql类型来传递特定于后端的数据类型

```
?SQL_EXCEX(yourConnection, 'UPDATE yourTable SET someField = ?{someVar} WHERE yourKey
= ?{lnID}', '', '1aResult', 0, '', '1 43')
```

注意

仅当您调用存储过程并且后端和 ODBC 驱动程序支持它时，才可以使用命名参数-例如 Microsoft SQL Server。

如果使用命名参数，则必须命名所有参数。

cCallback (可选)

在获取结果集或检索 PRINT 或 RAISERROR 语句或其他附加后端信息时进行回调的函数。

功能原型：

```
Function SQLCallback(lnSet, lnRow, lnRowCount)
```

&& && lnSet: 当前获取的结果集的编号；如果捕获到严重性较低的PRINT或RAISERROR语句，则为-1

&& lnRow: 已获取的当前行，或者lnSet为-1时lnRow包含PRINT或RAISERROR语句的消息/警告

&& lnRowCount: 要整体读取的行数；如果所使用的ODBC驱动程序不支持返回查询的匹配行，则为0 / -1；如果lnSet等于-1，则此参数没有意义，并且始终为.F.。

```
Endfunc
```

nCallbackInterval

默认值：100

在获取行时用于回调到回调过程的时间间隔。

您的回调过程将在获取第一行之前调用一次，然后在第 n 个记录上调用，并且在获取最后一行之后调用一次。

返回值

准备好的语句句柄，如果函数失败，则返回-1。

参考

[ASQLDatasources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLExecEx](#)

[SQLGetPropEx](#)

[SQLSetPropEx](#)

使用的 WinApi

[SQLAllocHandle](#)

SQLSetPropEx

扩展的 SQLSETPROP，设置 ODBC 连接属性。

SQLSetPropEx(nConnectionHandle | cCursorName, cAttribute, vValue)

参数

nConnectionHandle | cCursorName

从 SQLCONNECT / SQLSTRINGCONNECT 返回的有效连接句柄或由远程连接创建的游标的名称
-CURSORGETPROP ("ConnectHandle", cCursorName) 必须返回有效的连接句柄。

cAttribute

以下值之一：

值	描述
Trace	启动/停止 ODBC 调试跟踪。
TraceFile	设置 ODBC 调试跟踪文件。
Perfdata	启动/停止 MS SQL Server 性能数据记录。
PerfdataFile	设置 MS SQL Server 性能数据日志文件。
PerfdataLog	记录 MS SQL Server 性能数据。

vValue

不同属性的有效值是：

值	有效值
Trace	.F. 或. T. 停止 - 开始跟踪
TraceFile	有效的文件名。
Perfdata	.F. 或 .T. 以启动/停止 MS SQL Server 性能数据日志记录。
PerfdataFile	有效的文件名。
PerfdataLog	不需要任何值，只需传递此属性即可更新日志。

返回值

如果成功设置属性，则为 1；如果通过 AErrorEx 获得其他信息，则为 2。

参考

[ASQLDatasources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)[DeleteSQLDataSource](#)[SQLCancelEx](#)[SQLExecEx](#)[SQLGetPropEx](#)[SQLPrepareEx](#)

使用的 WinApi

[SQLSetConnectAttr](#)

打印机信息

提供关于打印机的信息

APaperSizes	将有关支持的纸张尺寸的信息存储到数组中。
APrinterForms	将有关指定打印机支持的窗体的信息存储到数组中。
APrintersEx	将有关可用打印机, 打印服务器, 域或打印提供程序的信息存储到数组中。
APrinterTrays	将有关打印机可用纸盒的信息存储到数组中。
APrintJobs	将有关指定打印机的一组指定打印作业的信息存储到数组中。

APaperSizes

将有关支持的纸张尺寸的信息存储到数组中。

APaperSizes(cArrayName, cPrinter, cPort [, nUnit])

参数

cArrayName

返回时, 数组包含以下信息:

列	内容	数据类型
1	纸张 ID-请参阅 DEVMODE 结构的 dmPaperSize 成员的文档。	N
2	纸张名称。	C
3	纸张尺寸的宽度, 以请求的单位。	N
4	纸张尺寸的长度, 以请求的单位。	N

cPrinter

要枚举纸张尺寸的打印机的名称。

cPort

打印机连接到的端口的名称。

nUnit (可选)

默认值: PAPER_SIZE_UNIT_MM

控制将纸张尺寸的宽度和长度存储到阵列中的度量单位。

此参数的可能值:

单位	描述
PAPER_SIZE_UNIT_MM	十分之一毫米
PAPER_SIZE_UNIT_INCH	英制
PAPER_SIZE_UNIT_POINT	点

返回值

纸张尺寸的数量。

参考

[APaperSizes](#)

[APrinterForms](#)

[APrintersEx](#)

[APrinterTrays](#)

[APrintJobs](#)

使用的 WinApi

[DeviceCapabilities](#) (使用参数 DC_PAPERS, DC_PAPERNAME 和 DC_PAPERSIZE)

APrinterForms

将有关指定打印机支持的窗体(Forms)的信息存储到数组中。

APrinterForms(cArrayName [, cPrinter])

参数

cArrayName

返回时，该数组包含来自 FORM_INFO_1 结构的以下信息。

列	内容	数据类型
1	窗体属性。 有关可能值的列表, 请参见 FORM_INFO_1 。	N
2	窗体名称。	C
3	窗体宽带, 以千分之一毫米为单位	N
4	窗体高度, 以千分之一毫米为单位。	N
5	窗体可打印区域的底部位置, 以千分之一毫米为单位。	N
6	窗体可打印区域的最高位置, 以千分之一毫米为单位。	N
7	表格可打印区域的左侧位置, 以千分之一毫米为单位。	N
8	表格可打印区域的正确位置, 以千分之一毫米为单位。	N

cPrinter

要枚举窗体的打印机的名称。

返回值

打印机窗体的数量。

参考

[APaperSizes](#)

[APrinterForms](#)

[APrintersEx](#)

[APrinterTrays](#)

[APrintJobs](#)

使用的 WinApi

[EnumForms](#)

[OpenPrinter](#)

[ClosePrinter](#)

APrintersEx

将有关可用打印机, 打印服务器, 域或打印提供程序的信息存储到数组中。

APrintersEx(cArrayName [, cName [, nFlags [, nLevel [, nOutputType]]]])

参数

cArrayName

如果 nLevel = 1

该数组包含来自 [PRINTER_INFO_1](#) 结构的以下信息。

列	内容	数据类型
1	标志。查看 PRINTER_INFO_1 以获取可能值的列表。	N
2	描述结构的内容。	C
3	命名结构的内容。	C
4	描述结构的其他数据。	C

如果 nLevel = 2

该数组包含来自 [PRINTER_INFO_2](#) 结构的以下信息。

列	内容	数据类型
1	控制打印机的服务器。 如果该字符串为空，则在本地控制打印机。	C
2	打印机的名称。	C
3	打印机的共享点。	C
4	用于将数据传输到打印机的端口。	C
5	打印机驱动程序的名称。	C
6	打印机的简要说明。	C
7	指定打印机的物理位置	C
8	用于创建分隔页的文件名。此页面用于分隔发送到打印机的打印作业。	C
9	打印机使用的打印处理器的名称。	C
10	用于记录打印作业的数据类型。	C
11	默认的打印处理器参数。	C
12	打印机属性。有关可能值的列表，请查看 PRINTER_INFO_2 。	N
13	后台打印程序用来路由打印作业的优先级值。	N
14	分配给每个打印作业的默认优先级值。	N
15	打印机最早打印作业的时间。 此值表示为格林尼治标准时间 12:00 AM GMT (格林尼治标准时间) 以来经过的分钟数。	N
16	打印机最晚打印作业的时间。 此值表示为格林尼治标准时间 12:00 AM GMT (格林尼治标准时间) 以来经过的分钟数。	N
17	打印机状态。有关可能值的列表，请查看 PRINTER_INFO_2 。	C
18	已排队等待打印机的打印作业数。	N
19	每分钟平均已在打印机上打印的页面数。	N

如果 nLevel = 4

该数组包含来自 [PRINTER_INFO_4](#) 结构的以下信息。

列	内容	数据类型
1	打印机的名称 (本地或远程)。	C
2	服务器的名称。	C
3	属性。有关可能值的列表, 请查看 PRINTER_INFO_4 。	N

如果 nLevel = 5

该数组包含来自 [PRINTER_INFO_5](#) 结构的以下信息。

列	内容	数据类型
1	打印机的名称。	C
2	用于将数据传输到打印机的端口。	C
3	打印机属性。有关可能值的列表, 请查看 PRINTER_INFO_5 。	N
4	设备未选择超时	N
5	传输重试超时	N

cName (可选)

默认值: 空

要枚举打印机的打印处理器, 域或服务器的名称。

如果 nLevel 为 1, nFlags 包含 PRINTER_ENUM_NAME, 而 cName 不为空, 则 cName 指定要枚举的对象的名称。

该字符串可以是服务器, 域或打印提供程序的名称。

如果 nLevel 为 1, nFlags 包含 PRINTER_ENUM_NAME, 而 cName 为空, 则该函数枚举可用的打印提供程序。

如果 nLevel 为 1, nFlags 包含 PRINTER_ENUM_REMOTE, 而 cName 为空, 则该函数枚举用户域中的打印机。

如果 nLevel 为 2 或 5, 则 cName 指定要枚举其打印机的服务器的名称。如果 cName 为空, 则该函数枚举安装在本地计算机上的打印机。

如果 nLevel 为 4, 则 cName 应该为空。该函数始终在本地计算机上查询。

当 cName 为空时，将 nFlags 设置为 PRINTER_ENUM_LOCAL + PRINTER_ENUM_CONNECTIONS 会枚举安装在本地计算机上的打印机。这些打印机包括物理连接到本地计算机的打印机以及与它具有网络连接的远程打印机。

nFlags (可选, 附加)

默认值: PRINTER_ENUM_LOCAL

可选以下值之一或组合:

Flag	描述
PRINTER_ENUM_LOCAL	该函数将忽略 cName 参数，并枚举本地安装的打印机。 Windows 95/98 / Me: 该功能还将枚举网络打印机，因为它们由本地打印提供程序处理。
PRINTER_ENUM_NAME	该函数枚举 cName 标识的打印机。这可以是服务器，域或打印提供程序。如果 cName 为空，则该函数枚举可用的打印提供程序。
PRINTER_ENUM_SHARED	该函数枚举具有共享属性的打印机。 不能单独使用；使用+操作与另一个 PRINTER_ENUM 类型组合。
PRINTER_ENUM_DEFAULT	Windows 95/98 / Me: 该函数返回有关默认打印机的信息。
PRINTER_ENUM_CONNECTIONS	Windows NT / 2000 / XP: 该功能枚举用户先前已连接到的打印机的列表。
PRINTER_ENUM_NETWORK	Windows NT / 2000 / XP: 此功能枚举计算机域中的网络打印机。仅当 nLevel 为 1 时，此值才有效。
PRINTER_ENUM_REMOTE	Windows NT / 2000 / XP: 该功能枚举计算机域中的网络打印机和打印服务器。仅当 nLevel 为 1 时，此值才有效。

nLevel (可选)

默认值: 2

指定函数返回的信息级别。

有效值为 1、2、4 和 5，它们对应于 [PRINTER_INFO_1](#), [PRINTER_INFO_2](#), [PRINTER_INFO_4](#) 和 [PRINTER_INFO_5](#) 数据结构。

Windows 95/98 / Me: 值可以是 1、2 或 5。

Windows NT / 2000 / XP: 此值可以是 1、2、4 或 5。

nOutputType (可选)

默认值: APRINT_OUTPUT_ARRAY

有效值:

APRINT_OUTPUT_ARRAY-信息以多维数组形式返回

APRINT_OUTPUT_OBJECTARRAY-信息以单维数组的形式返回，其中每一行包含一个具有自定义

集属性的对象

返回值

打印机数量。

参考

[APaperSizes](#)

[APrinterForms](#)

[APrintersEx](#)

[APrinterTrays](#)

[APrintJobs](#)

使用的 WinApi

[EnumPrinters](#)

APrinterTrays

将有关打印机可用纸盒的信息存储到数组中。

APrinterTrays(cArrayName, cPrinter, cPort)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	打印机纸盒的名称。	C
2	纸盒 ID。	N

cPrinter

要枚举纸盒的打印机的名称。

cPort

打印机连接到的端口的名称。

返回值

打印机纸盒的数量。

参考

[APaperSizes](#)

[APrinterForms](#)

[APrintersEx](#)

[APrinterTrays](#)

[APrintJobs](#)

使用的 WinApi

[DeviceCapabilities](#) (使用参数 DC_BINS 和 DC_BINNAMES)

APrintJobs

将有关指定打印机的一组指定打印作业的信息存储到数组中。

APrintJobs(cArrayName, cPrinter [, nLevel])

参数

cArrayName

如果 nLevel = 1

该数组包含来自 [JOB_INFO_1](#) 结构的以下信息。

列	内容	数据类型
1	打印作业的名称。	C
2	后台处理作业的打印机的名称。	C
3	拥有打印作业的用户名。	C
4	创建打印作业的机器的名称。	C
5	用于记录打印作业的数据类型。	C
6	打印作业的状态。	C
7	作业标识符。	N
8	工作状态。有关可能值的列表，请查看 JOB_INFO_1 。	N
9	作业优先级。有关可能值的列表，请查看 JOB_INFO_1 。	N
10	作业在打印队列中的位置。	N
11	文档包含的总页数。 如果打印作业不包含页面定界信息，则该值为零。	N
12	已打印的页数。	N

	如果打印作业不包含页面定界信息，则该值为零。	
13	<p>此文档的后台处理时间。</p> <p>此时间值以通用时间坐标（UTC）格式显示。在显示之前，应将其转换为本地时间值。</p> <p>您可以使用 UTC2DT 函数执行转换。</p>	T

如果 nLevel = 2

该数组包含来自 JOB_INFO_2 结构的以下附加信息。

列	内容	数据类型
14	在打印作业或打印作业时发生错误时应通知的用户名。	C
15	应该用于打印作业的打印处理器的名称。	C
16	打印处理器参数。	C
17	用于处理打印作业的打印机驱动程序的名称。	C
18	自作业开始打印以来经过的总时间（以毫秒为单位）。	N
19	可以打印作业的最早时间。	N
20	可以打印作业的最晚时间。	N
21	作业的大小（以字节为单位）。	N

cPrinter

要枚举其 printjobs 的打印机的名称。

nLevel (可选)

默认值：1

要返回的信息级别，为 1 或 2。

返回值

打印作业的数量。

参考

[APaperSizes](#)

[APrinterForms](#)

[APrintersEx](#)

[APrinterTrays](#)

[APrintJobs](#)

使用的 WinApi

[EnumJobs](#)

[OpenPrinter](#)

[ClosePrinter](#)

进程信息

提供有关系统上运行的进程的信息。

AHeapBlocks	将有关进程已分配的堆块的信息存储到数组中。
AProcesses	将有关系统当前正在运行的进程的信息存储到数组中。
AProcessHeaps	将有关进程堆的信息存储到数组中。
AProcessModules	将有关进程加载的模块 (DLL) 的信息存储到数组中。
AProcessThreads	将有关进程的线程信息存储到数组中。

AHeapBlocks

将有关进程已分配的堆块的信息存储到数组中。

AHeapBlocks(cArrayName, nProcessID, nHeapID)

参数

cArrayName

返回时，该数组包含来自 [HEAPENTRY32](#) 结构的以下信息。

列	内容	数据类型
1	堆块的句柄。	N
2	块开始的线性地址。	N
3	堆块的大小，以字节为单位。	N
4	标记	N

nProcessID

拥有堆的进程上下文的标识符。

nHeapID

要枚举的堆的标识符。

返回值

堆块数。

参考

[AProcesses](#)

[AProcessHeaps](#)[AProcessModules](#)[AProcessThreads](#)

使用的 WinApi

[Heap32First](#)[Heap32Next](#)

AProcesses

将有关系统当前正在运行的进程的信息存储到数组中。

AProcesses(cArrayName)

参数

cArrayName

返回时，数组包含来自 [PROCESSENTRY32](#) 结构的以下信息。

列	内容	数据类型
1	可执行文件名	C
2	进程 ID	N
3	父进程 ID	N
4	线程数	N
5	基本优先级	N

返回值

进程数

参考

[AHeapBlocks](#)[AProcessHeaps](#)[AProcessModules](#)[AProcessThreads](#)

使用的 WinApi

[CreateToolhelp32Snapshot](#)

[Process32First](#)[Process32Next](#)[EnumProcesses](#) (Windows NT only)[OpenProcess](#) (Windows NT only)[EnumProcessModules](#) (Windows NT only)[NtQueryInformationProcess](#) (Windows NT only)[CloseHandle](#) (Windows NT only)

AProcessHeaps

将有关进程堆的信息存储到数组中。

AProcessHeaps(cArrayName, nProcessID)

参数

cArrayName

返回时，该数组包含来自 [HEAPLIST32](#) 结构的以下信息。

列	内容	数据类型
1	堆 ID	N
2	标记	N

nProcessID

要枚举其堆的进程的进程 ID。

返回值

进程堆数。

参考

[AHeapBlocks](#)[AProcesses](#)[AProcessModules](#)[AProcessThreads](#)

使用的 WinApi

[CreateToolhelp32Snapshot](#)

[Heap32ListFirst](#)[Heap32ListNext](#)

AProcessModules

将有关进程加载的模块（DLL）的信息存储到数组中。

AProcessModules(cArrayName, nProcessID)

参数

cArrayName

返回时，该数组包含来自 [MODULEENTRY32](#) 结构的以下信息。

列	内容	数据类型
1	模块	C
2	执行路径	C
3	模块句柄	N
4	基本尺寸	N
5	基本地址	N

nProcessID

要枚举已加载模块的进程的进程 ID。

返回值

模块数。

参考

[AHeapBlocks](#)

[AProcesses](#)

[AProcessHeaps](#)

[AProcessThreads](#)

使用的 WinApi

[CreateToolhelp32Snapshot](#)

[Module32First](#)

[Module32Next](#)

AProcessThreads

将有关进程的线程信息存储到数组中。

AProcessThreads(cArrayName, nProcessID)

参数

cArrayName

返回时，数组包含 [THREADENTRY32](#) 结构的以下信息。

列	内容	数据类型
1	进程 ID	N
2	所有者进程 ID	N
3	基本优先级	N

nProcessID

要枚举线程的进程的进程 ID。

返回值

线程数。

参考

[AHeapBlocks](#)

[AProcesses](#)

[AProcessHeaps](#)

[AProcessModules](#)

使用的 WinApi

[CreateToolhelp32Snapshot](#)

[Thread32First](#)

[Thread32Next](#)

RAS(远程访问)

管理 RAS (远程访问) 连接。

AbortRasConnectionNotification	终止监视由 RasConnectionNotificationEx 启动的 RAS 连接的线程。
--	--

ARasConnections	将有关所有活动 RAS 连接的信息存储到数组中。
ARasDevices	将有关所有可用的支持 RAS 的设备的信息存储到数组中。
ARasPhonebookEntries	将有关远程访问电话簿中所有条目名称的信息存储到数组中。
RasClearConnectionStatistics	RasClearConnectionStatistics 函数清除指定 RAS 连接的所有累积统计信息。
RasConnectionNotificationEx	每当创建或终止连接时，都会启动一个新线程来监视系统的 RAS 连接，该线程将调用指定的回调过程。
RasDialDlgEx	使用指定的电话簿条目和登录用户的凭据建立 RAS 连接。
RasDialEx	RasDial 函数在 RAS 客户端和 RAS 服务器之间建立 RAS 连接。
RasGetConnectStatusEx	将有关指定远程访问连接的当前状态的信息检索到数组中。
RasHangUpEx	终止远程访问连接。
RasPhonebookDlgEx	显示主“拨号网络”对话框。 在此模式对话框中，用户可以拨打、编辑或删除所选的电话簿条目，创建新的电话簿条目或指定用户首选项。对话框关闭时，该函数返回。

AbortRasConnectionNotification

终止监视由 [RasConnectionNotificationEx](#) 启动的 RAS 连接的线程。

AbortRasConnectionNotification(nThreadHandle)

参数

nThreadHandle

从 [RasConnectionNotificationEx](#) 返回的线程句柄。

返回值

如果监视 RAS 连接的线程终止，返回.T.；否则返回.F.。

参考

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)[RasConnectionNotificationEx](#)[RasDialDlgEx](#)[RasDialEx](#)[RasGetConnectStatusEx](#)[RasHangUpEx](#)[RasPhonebookDlgEx](#)

使用的 WinApi

[SetEvent](#)

ARasConnections

将有关所有活动 RAS 连接的信息存储到数组中。

ARasConnections(cArrayName)

参数

返回时，该数组包含来自 [RASCONN](#) 和 [RASPPPIP](#) 结构的以下信息。

列	内容
1	拨号连接的名称。
2	设备名称。
3	设备类型。
4	IP 地址
5	HRASCONN 句柄

返回值

RAS 连接数。

参考

[AbortRasConnectionNotification](#)[ARasDevices](#)[ARasPhonebookEntries](#)[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)[RasDialDlgEx](#)[RasDialEx](#)[RasGetConnectStatusEx](#)[RasHangUpEx](#)[RasPhonebookDlgEx](#)

使用的 WinApi

[RasEnumConnections](#)[RasGetProjectionInfo](#)

ARasDevices

将有关所有可用的支持 RAS 的设备的信息存储到数组中。

ARasDevices(cArrayName)

参数

cArrayName

返回时，该数组包含来自 [RASDEVINFO](#) 结构的以下信息。

列	内容
1	设备类型。
2	设备名称。

返回值

RAS 设备数。

参考

[AbortRasConnectionNotification](#)[ARasConnections](#)[ARasPhonebookEntries](#)[RasClearConnectionStatistics](#)[RasConnectionNotificationEx](#)[RasDialDlgEx](#)

[RasDialEx](#)[RasGetConnectStatusEx](#)[RasHangUpEx](#)[RasPhonebookDlgEx](#)

使用的 WinApi

[RasEnumDevices](#)

ARasPhonebookEntries

将有关远程访问电话簿中所有条目名称的信息存储到数组中。

ARasPhonebookEntries(cArrayName[, cPhonebookfile])

参数

cArrayName

返回时，数组包含来自 [RASENTRYNAME](#) 结构的以下信息。

列	内容
1	电话簿条目的名称。
2	电话簿入口的标志。

标志值可以是：

REN_User 或 REN_AllUsers (在 rasapi32.h 中定义)

在 Windows 9x 上，不支持标志，因此始终为 0。

cPhonebookfile (可选)

默认值：使用系统默认电话簿

否则，传递有效的电话簿文件名。

返回值

RAS 电话簿条目的数量。

参考

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[RasClearConnectionStatistics](#)[RasConnectionNotificationEx](#)[RasDialDlgEx](#)[RasDialEx](#)[RasGetConnectStatusEx](#)[RasHangUpEx](#)[RasPhonebookDlgEx](#)

使用的 WinApi

[RasEnumEntries](#)

RasClearConnectionStatistics

RasClearConnectionStatistics 函数清除指定 RAS 连接的所有累积统计信息。

RasClearConnectionStatistics(nRasConn)

参数

nRasConn

有效的 HRASCONN 连接句柄。

返回值

.T.

参考

[AbortRasConnectionNotification](#)[ARasConnections](#)[ARasDevices](#)[ARasPhonebookEntries](#)[RasConnectionNotificationEx](#)[RasDialDlgEx](#)[RasDialEx](#)[RasGetConnectStatusEx](#)[RasHangUpEx](#)

[RasPhonebookDlgEx](#)

使用的 WinApi

[RasClearConnectionStatistics](#)

RasConnectionNotificationEx

每当创建或终止连接时，都会启动一个新线程来监视系统的 RAS 连接，该线程将调用指定的回调过程。

RasConnectionNotificationEx(nRasConn, nFlags, cCallback)

参数

nRasConn

为其接收通知的 RAS 连接的句柄。

这可以是 [RasDialEx](#) 或 [ARasConnections](#) 返回的句柄。

如果此参数为 INVALID_HANDLE_VALUE (-1)，则会收到有关本地客户端上所有 RAS 连接的通知。

nFlags

以下值（在 rasapi32.h 中定义）的一个或组合。

常量	描述
RASCN_Connection	如果 nRasConn 为 INVALID_HANDLE_VALUE，则在创建任何 RAS 连接时都会进行回调。
RASCN_Disconnection	当 nRasConn 连接终止时，将进行回调。 如果 nRasConn 是多链路连接，则在断开所有子条目时进行回调。 如果 nRasConn 为 INVALID_HANDLE_VALUE，则在任何 RAS 连接终止时进行回调。
RASCN_BandwidthAdded	Windows NT：如果 nRasConn 是组合的多链接连接的句柄，则在连接子条目时进行回调。
RASCN_BandwidthRemoved	Windows NT：如果 nRasConn 是组合的多链接连接的句柄，则在断开子条目时进行回调。

cCallBack

创建/终止 RAS 连接时要调用的函数的名称。

该函数应具有以下原型。

```
Function OnConnectionChange(lnConn, lnError)
If lnError = 0
    && 调用 ARasConnections 以查看是否建立了新连接或已断开连接
    Local lnConnCount, laRasConns[1,5]
```

```

lnConnCount = ARasConnections('laRasConns')
If AScan(laRasConns, lnConn, 1, 0, 5, 8) > 0
    ? "RAS 连接", lnConn, "已连接"
Else
    ? "RAS 连接", lnConn, "已断开"
Endif
Else
    ? "WaitForMultipleObjects失败, 错误代码为", lnError
Endif
Endfunc

```

注意

监视单个活动 RAS 连接时，断开连接后，通知会自动结束。

返回值

监视 RAS 连接的线程的句柄。

参考

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasHangUpEx](#)

[RasPhonebookDlgEx](#)

使用的 WinApi

[RasConnectionNotification](#)

[WaitForMultipleObjects](#)

[PostMessage](#)

RasDialDlgEx

使用指定的电话簿条目和登录用户的凭据建立 RAS 连接。

```
RasDialDlgEx([cPhonebook [, cPhonebookEntry [, cPhoneNumber [, nSubEntry [, nHwndOwner]]]]])
```

参数

cPhonebook (可选)

默认值：空

如果 cPhonebook 为空，则使用系统默认电话簿，否则应指定有效的电话簿文件名。

默认电话簿文件是用户在“拨号网络”对话框的“用户首选项”属性表中选择的文件。

cPhonebookEntry (可选)

要拨打的电话簿条目的名称。

cPhoneNumber (可选)

默认值：空

一个字符串，它指定一个电话号码，该电话号码将覆盖存储在电话簿条目中的号码。如果此参数为空，则 RasDialDlgEx 使用电话簿条目中的数字。

nSubEntry (可选)

默认值： 0

指定要拨打的一个或多个子条目。如果 nSubEntry 为 0，则 RasDialDlgEx 拨打与指定电话簿条目关联的所有子条目。

否则，要指示要拨打的单个子条目的索引，nSubEntry 必须是一个从 1 到子条目数的数字。

nHwndOwner (可选)

默认值： 0

指定拥有模态 RasDialDlgEx 对话框的窗口。

此参数可以是任何有效的窗口句柄，如果对话框没有所有者，则可以为 0。

返回值

如果建立了 RAS 连接返回.T.；否则返回.F.。

参考

[AbortRasConnectionNotification](#)

[ARasConnections](#)[ARasDevices](#)[ARasPhonebookEntries](#)[RasClearConnectionStatistics](#)[RasConnectionNotificationEx](#)[RasDialEx](#)[RasGetConnectStatusEx](#)[RasHangUpEx](#)[RasPhonebookDlgEx](#)

使用的 WinApi

[RasDialDlg](#)

RasDialEx

RasDial 函数在 RAS 客户端和 RAS 服务器之间建立 RAS 连接。

RasDialEx(cPhonebookEntry | nRasDialParamsPtr [, cPhonebook [, cCallback [, nFlags [, nRasConn]]]])

参数

cPhonebookEntry | nRasDialParamsPtr

电话簿条目的名称或 [RASDIALPARAMS](#) 结构的指针。

cPhonebook (可选)

默认：空

如果 cPhonebook 为空，则使用系统默认电话簿，否则应指定有效的电话簿文件名。

默认电话簿文件是用户在“拨号网络”对话框的“用户首选项”属性表中选择的文件。

cCallback (可选)

默认值：在拨号操作期间不进行任何回调。

使用 FoxPro 函数进行回调。

如果传递回调函数，则 [RasDial](#) 函数将异步执行。

它将立即返回，并且在拨号过程中将调用您的回调函数，并且向您提供操作的当前状态。

请参阅 ras.prg 示例中的类 RasConnection: DialCallback。

nFlags (可选)

默认值: 0

以下标志之一或组合。

常量	描述
RDEOPT_UsePrefixSuffix	<p>如果设置了此标志，则 RasDialEx 使用 RAS 电话簿中的前缀和后缀。</p> <p>如果未设置此标志，则 RasDialEx 将忽略 RAS 电话簿中的前缀和后缀。</p> <p>如果在对 RasDialEx 的调用中未指定电话簿条目名称，则此标志的实际值将被忽略，并假定为零。</p>
RDEOPT_PausedStates	<p>如果设置了此标志，则 RasDialEx 接受暂停状态。 暂停状态的示例包括终端模式，重试登录，更改密码，设置回叫号码和 EAP 身份验证。</p> <p>如果未设置此标志，则 RasDialEx 进入暂停状态时将报告致命错误。</p>
RDEOPT_IgnoreModemSpeaker	<p>如果设置了此标志，则 RasDialEx 将忽略 RAS 电话簿中的调制解调器扬声器设置，并使用 RDEOPT_SetModemSpeaker 位标志指定的设置。</p> <p>如果未设置此标志，则 RasDialEx 使用 RAS 电话簿中的调制解调器扬声器设置，并忽略 RDEOPT_SetModemSpeaker 位标志指定的设置。</p> <p>如果在对 RasDialEx 的调用中未指定电话簿条目名称，则在默认设置或 RDEOPT_SetModemSpeaker 位标志指定的设置之间进行选择。 如果 RDEOPT_IgnoreModemSpeaker 为零，则使用默认设置。 如果 RDEOPT_IgnoreModemSpeaker 为 1，则使用 RDEOPT_SetModemSpeaker 指定的设置。</p>
RDEOPT_SetModemSpeaker	<p>如果设置了此标志，并且 RDEOPT_IgnoreModemSpeaker 为 1，则 RasDialEx 将调制解调器扬声器设置为打开。</p> <p>如果未设置此标志，并且 RDEOPT_IgnoreModemSpeaker 为 1，则 RasDialEx 将调制解调器扬声器设置为关闭。</p> <p>如果未设置 RDEOPT_IgnoreModemSpeaker，则 RasDialEx 将忽略 RDEOPT_SetModemSpeaker 的值，并根据 RAS 电话簿设置或默认设置来设置调制解调器扬声器。</p>
RDEOPT_IgnoreSoftwareCompression	<p>如果设置了此标志，则 RasDialEx 将忽略 RAS 电话簿中的软件压缩设置，并使用 RDEOPT_SetSoftwareCompression 位标志指定的设置。</p> <p>如果未设置此标志，则 RasDialEx 使用 RAS 电话簿中的软件压缩设置，并忽略 RDEOPT_SetSoftwareCompression 标志指定的设置。</p>

	如果在对 RasDialEx 的调用中未指定电话簿条目名称，则在默认设置或 RDEOPT_SetSoftwareCompression 位标志指定的设置之间进行选择。如果 RDEOPT_IgnoreSoftwareCompression 为零，则使用默认设置。如果 RDEOPT_IgnoreSoftwareCompression 为 1，则使用 RDEOPT_SetSoftwareCompression 指定的设置。
RDEOPT_SetSoftwareCompression	<p>如果设置了此标志，并且 RDEOPT_IgnoreSoftwareCompression 为 1，则 RasDialEx 使用软件压缩。</p> <p>如果未设置此标志，并且 RDEOPT_IgnoreSoftwareCompression 为 1，则 RasDialEx 不使用软件压缩。</p> <p>如果 RDEOPT_IgnoreSoftwareCompression 为零，则 RasDialEx 将忽略 RDEOPT_SetSoftwareCompression 的值，并根据 RAS 电话簿设置或默认设置来设置软件压缩状态。</p>
RDEOPT_UseCustomScripting	<p>如果设置了此标志，则 RasDialEx 建立与服务器的连接后将调用自定义脚本 DLL。</p> <p>只有满足以下所有条件，RasDialEx 才会调用 DLL：</p> <ul style="list-style-type: none"> 设置此标志 自定义脚本 DLL 已在系统中正确注册。 RASEO_CustomScript 选项在电话簿条目中指定。
RDEOPT_CustomDial	如果设置了此标志，则 RasDialEx 将正常拨号，而不是调用自定义拨号器的 RasCustomDial 入口点。当 RasCustomDialDlg 调用 RasDialEx 时使用此标志，这样 RasDialEx 不必回叫自定义拨号器。
RDEOPT_NoUser	Windows Vista 或更高版本：如果设置了此标志，则从 Windows 登录上下文拨打连接。在这种情况下，RasDialEx 使用默认的用户首选项进行连接。如果在 RasCustomDialDlg 的 dwfOptions 参数中设置了 RCD_Logon，则必须设置此标志。

nRasConn (可选)

默认值：未使用从暂停状态恢复异步 RasDial 调用时，传递现有的 HRASCONN 句柄。

返回值

代表 RAS 连接的句柄 (HRASCONN)。

示例

```
Local lnConn, loRasParams
```

```

loRasParams = Createobject('RASDIALPARAMS')
loRasParams.szPhoneNumber = '111 2453'
loRasParams.szCallbackNumber = '*'
loRasParams.szUserName = 'username'
loRasParams.szPassword = '*****'
loRasParams.szDomain = ''
lnConn = RasDialEx(loRasParams.Address)

```

参考

[AbortRasConnectionNotification](#)
[ARasConnections](#)
[ARasDevices](#)
[ARasPhonebookEntries](#)
[RasClearConnectionStatistics](#)
[RasConnectionNotificationEx](#)
[RasDialDlgEx](#)
[RasGetConnectStatusEx](#)
[RasHangUpEx](#)
[RasPhonebookDlgEx](#)

使用的 WinApi

[RasDial](#)
[RasGetEntryDialParams](#)
[RasGetEapUserIdentity](#)
[RasGetConnectStatus](#)
[RasHangUp](#)
[RasFreeEapUserIdentity](#)

RasGetConnectStatusEx

将有关指定远程访问连接的当前状态的信息检索到数组中。

RasGetConnectStatusEx(nRasConn, cArrayname)

参数

nRasConn

一个有效的 HRASCONN 句柄，以获取其状态信息。

cArrayname

返回时，数组包含来自 RASCONNSTATUS 结构的以下信息。

元素	内容
1	RASCONNSTATE
2	错误代码
3	设备类型
4	设备名称
5	电话号码

返回值

.T.

参考

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasHangUpEx](#)

[RasPhonebookDlgEx](#)

使用的 WinApi

[RasGetConnectStatus](#)

RasHangUpEx

终止远程访问连接。

RasHangUpEx(nRasConn)

参数

nRasConn

有效的 HRASCONN 连接句柄。

返回值

.T.

参考

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasPhonebookDlgEx](#)

使用的 WinApi

[RasHangUp](#)

[RasGetConnectStatus](#)

RasPhonebookDlgEx

显示主“拨号网络”对话框。

在此模式对话框中，用户可以拨打，编辑或删除所选的电话簿条目，创建新的电话簿条目或指定用户首选项。 对话框关闭时，该函数返回。

RasPhonebookDlgEx([cInitialSelectedEntry [, cPhonebook [, cCallback [, nFlags]]]])

参数

cInitialSelectedEntry (可选)

默认值：空

电话簿条目的名称，应首先在对话框中选择。

cPhonebook (可选)

默认值：空

如果 cPhonebook 为空，则使用系统默认电话簿，否则应指定有效的电话簿文件名。

默认电话簿文件是用户在“拨号网络”对话框的“用户首选项”属性表中选择的文件。

cCallback (可选)

默认值：空

如果此参数为空，则不会从对话框进行回调。

否则，在对话框运行时将回调传递的函数。

该函数应具有以下原型。

```
Function DialogCallback(lnEvent, lcText, lnData)
Do Case
    Case lnEvent = RASPBDEVENT_AddEntry
        ? "RasEntry 添加: ", lcText
    Case lnEvent = RASPBDEVENT_EditEntry
        ? "RasEntry 编辑: ", lcText
    Case lnEvent = RASPBDEVENT_RemoveEntry
        ? "RasEntry 移除: ", lcText
    Case lnEvent = RASPBDEVENT_DialEntry
        ? "RasEntry 拨号: ", lcText
    Case lnEvent = RASPBDEVENT_EditGlobals
        ? "RasEntry (global) 编辑: ", lcText
    Case lnEvent = RASPBDEVENT_NoUser
        ? "RasEntry NoUser 事件"
    Case lnEvent = RASPBDEVENT_NoUserEdit
        ? "RasEntry NoUserEdit 事件"
Endcase
Endfunc
```

nFlags (可选)

默认值：0

以下值之一（或在 rasapi32.h 中定义）的组合。

常量	描述
RASPBDFLAG_PositionDlg	使 RasPhonebookDlg 使用 xDlg 和 yDlg 成员指定的值来放置对话框。如果未设置此标志，则除非 hwndOwner 为 NULL，否则对话框将位于所有者窗口的中央，在这种情况下，对话框将位于屏幕的中央。

RASPBDFLAG_ForceCloseOnDial	打开拨号关闭选项，覆盖用户的首选项。此选项适用于 RASAutoDial 等功能，用户的目标是立即建立连接。
RASPBDFLAG_NoUser	导致 pCallback 成员指定的 RasPBDlgFunc 回调函数在对话框启动时接收 RASPBDEVENT_NoUser 通知。该标志用于 WinLogon 应用程序中没有登录用户的情况。通常，应用程序不应使用此标志。
RASPBDFLAG_UpdateDefaults	使默认窗口位置在退出时保存。此标志主要由 RASPHONE.EXE 使用，典型应用程序不应使用。

返回值

如果建立 RAS 连接返回.T.；否则返回.F.。

参考

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasHangUpEx](#)

使用的 WinApi

[RasPhonebookDlg](#)

注册表

管理注册表的函数

ARegistryKeys	将有关指定注册表项的所有子项的信息存储到数组中。
ARegistryValues	将有关指定注册表项的所有值的信息存储到数组中。
CancelRegistryChange	停止监视指定注册表项更改的线程。
CloseRegistryKey	关闭提供的注册表项句柄 (HKEY)。
CreateRegistryKey	创建或打开注册表项。
DeleteRegistryKey	删除指定的注册表项。

FindRegistryChange	监视注册表项中单独线程中的更改。
OpenRegistryKey	打开指定的注册表项。
ReadRegistryKey	检索指定注册表值的数据。
RegistryHiveToObject	将包括所有子项的注册表项存储到对象中。
RegistryValuesToObject	将注册表项的所有值存储到一个对象中。
WriteRegistryKey	在注册表项下设置数据和指定值的类型。

ARegistryKeys

将有关指定注册表项的所有子项的信息存储到数组中。

ARegistryKeys(cArrayName, nRegKey, cKeyName [, nFlags])

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	注册表项名称	C
2	注册表项类名称 (仅当 nFlags 包含 REG_ENUMCLASSENAME 时)	C
3	上一次写入时间 (仅当 nFlags 包含 REG_ENUMWRITETIME 时)	T

nRegKey

从 [OpenRegistryKey](#) 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。 该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。 不应在模拟其他用户的服务或应用程序中使用此句柄。
HKEY_CURRENT_CONFIG	包含有关本地计算机系统的当前硬件配置文件的信息。 HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。 有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。

	HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。使用此键可以更轻松地建立当前用户的设置。密钥映射到 HKEY_USERS 中当前用户的分支。在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项（HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外）都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包含在漫游用户配置文件的按用户注册表部分中。</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000：从 Windows 7 和 Windows Server 2008 R2 开始支持此键。</p>
HKEY_LOCAL_MACHINE	属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。它包含用于保存当前配置数据的子项，这些子项包括即插即用信息（Enum 分支，其中包括系统上所有硬件的完整列表），网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。
HKEY_PERFORMANCE_DATA	从属于此注册表项的注册表项允许您访问性能数据。数据实际上没有存储在注册表中；注册表功能使系统从其源收集数据。

HKEY_PERFORMANCE_NLSTEXT	从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。这些条目不适用于Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_PERFORMANCE_TEXT	属于此键的注册表项引用了描述美式英语计数器的文本字符串。这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

cKeyName (可选)

nRegKey 中传递的键的子键的路径。

nFlags (可选)

默认值= 0

可以是以下值的组合。

Flag	含义
REG_ENUMCLASSENTRY	注册表项类名称也将被检索。
REG_ENUMWRITETIME	也检索注册表的上次写入时间。

返回值

注册表项的数量。

参考

[ARegistryValues](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)

[ReadRegistryKey](#)

[RegistryHiveToObject](#)

[RegistryValuesToObject](#)

[WriteRegistryKey](#)

使用的 WinApi

[RegEnumKeyEx](#)[RegOpenKeyEx](#)[RegQueryInfoKey](#)

ARegistryValues

将有关指定注册表项的所有值的信息存储到数组中。

ARegistryValues(cArrayName, nRegKey, cKeyName [, nFlags])

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	注册表值名称	C
2	注册表值类型 (仅当 nFlags 包含 REG_ENUMTYPE 时)	C
3	用于注册表值的数据 (仅当 nFlags 包含 REG_ENUMVALUE 时)	variant

nRegKey

从 [OpenRegistryKey](#) 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	<p>从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。</p> <p>该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。</p> <p>HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。 有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p>

	HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。使用此键可以更轻松地建立当前用户的设置。密钥映射到 HKEY_USERS 中当前用户的分支。在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项（HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外）都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包含在漫游用户配置文件的按用户注册表部分中。</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000：从 Windows 7 和 Windows Server 2008 R2 开始支持此键。</p>
HKEY_LOCAL_MACHINE	属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。它包含用于保存当前配置数据的子项，这些子项包括即插即用信息（Enum 分支，其中包括系统上所有硬件的完整列表），网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。
HKEY_PERFORMANCE_DATA	从属于此注册表项的注册表项允许您访问性能数据。数据实际上没有存储在注册表中；注册表功能使系统从其源收集数据。

HKEY_PERFORMANCE_NLSTEXT	从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。这些条目不适用于Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_PERFORMANCE_TEXT	属于此键的注册表项引用了描述美式英语计数器的文本字符串。这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

cKeyName (可选)

nRegKey 中传递的键的子键的路径。

nFlags (可选)

默认值= 0

可以是以下值的组合。

Flag	含义
REG_ENUMTYPE	也检索注册表值的类型。
REG_ENUMVALUE	也检索注册表值的数据。

返回值

注册表值的数量。

参考

[ARegistryKeys](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)

[ReadRegistryKey](#)

[RegistryHiveToObject](#)

[RegistryValuesToObject](#)

[WriteRegistryKey](#)

使用的 WinApi

[RegEnumValue](#)

[RegOpenKeyEx](#)

[RegQueryInfoKey](#)

CancelRegistryChange

停止监视指定注册表项更改的线程。

CancelRegistryChange(nThreadHandle)

参数

nThreadHandle

从 [FindRegistryChange](#) 返回的线程句柄。

返回值

如果监视注册表项的线程终止返回.T.; 否则返回.F.。

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)

[ReadRegistryKey](#)

[RegistryHiveToObject](#)

[RegistryValuesToObject](#)

[WriteRegistryKey](#)

使用的 WinApi

[SetEvent](#)

CloseRegistryKey

关闭提供的注册表项句柄 (HKEY)。

CloseRegistryKey(nRegKey)

参数

nRegKey

从 [CreateRegistryKey](#) 或 [OpenRegistryKey](#) 返回的注册表项的句柄。

返回值

.T.

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CancelRegistryChange](#)

[CreateRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)

[ReadRegistryKey](#)

[RegistryHiveToObject](#)

[RegistryValuesToObject](#)

[WriteRegistryKey](#)

使用的 WinApi

[RegCloseKey](#)

CreateRegistryKey

创建或打开注册表项。

CreateRegistryKey(nRegKey, cKeyName [, nAccessRights [, nOptions [, cClass]]])

参数

nRegKey

从 [OpenRegistryKey](#) 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	<p>从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。</p> <p>该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。</p> <p>HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。 有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。 这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。 使用此键可以更轻松地建立当前用户的设置。 密钥映射到 HKEY_USERS 中当前用户的分支。 在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。 例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。 该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。 如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。 建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项 (HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除</p>

	<p>外) 都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目, 请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是, 调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。 这些条目不包括在漫游用户配置文件的按用户注册表部分中。</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000: 从 Windows 7 和 Windows Server 2008 R2 开始支持此键。</p>
HKEY_LOCAL_MACHINE	<p>属于此键的注册表项定义了计算机的物理状态, 包括有关总线类型, 系统内存以及已安装的硬件和软件的数据。 它包含用于保存当前配置数据的子项, 这些子项包括即插即用信息 (Enum 分支, 其中包括系统上所有硬件的完整列表), 网络登录首选项, 网络安全信息, 与软件相关的信息 (例如 (如服务器名称和服务器位置), 以及其他系统信息)。</p>
HKEY_PERFORMANCE_DATA	<p>从属于此注册表项的注册表项允许您访问性能数据。 数据实际上没有存储在注册表中; 注册表功能使系统从其源收集数据。</p>
HKEY_PERFORMANCE_NLSTEXT	<p>从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。</p> <p>Windows 2000: 不支持此键。</p>
HKEY_PERFORMANCE_TEXT	<p>属于此键的注册表项引用了描述美式英语计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。</p> <p>Windows 2000: 不支持此键。</p>
HKEY_USERS	<p>从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。</p>

cKeyName

应该创建的子项的名称。

如果 cKeyName 是一个空字符串, 则该函数将为 nRegKey 指定的键返回一个新的句柄。

nAccessRights (可选)

默认值: KEY_ALL_ACCESS

可以为以下值之一或组合。

常量	描述

KEY_ALL_ACCESS	合并 STANDARD_RIGHTS_REQUIRED, KEY_QUERY_VALUE, KEY_SET_VALUE, KEY_CREATE_SUB_KEY, KEY_ENUMERATE_SUB_KEYS, KEY_NOTIFY 和 KEY_CREATE_LINK 访问权限。
KEY_CREATE_LINK	保留供系统使用。
KEY_CREATE_SUB_KEY	创建注册表项的子项所必需。
KEY_ENUMERATE_SUB_KEYS	枚举注册表项的子项所必需。
KEY_EXECUTE	等效于 KEY_READ。
KEY_NOTIFY	要求为注册表项或注册表项的子项请求更改通知。
KEY_QUERY_VALUE	查询注册表项的值所必需。
KEY_READ	合并 STANDARD_RIGHTS_READ, KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS 和 KEY_NOTIFY 值。
KEY_SET_VALUE	创建, 删除或设置注册表值所必需。
KEY_WOW64_32KEY	表示 64 位 Windows 上的应用程序应在 32 位注册表视图上运行。 有关更多信息, 请参见访问备用注册表视图。 必须使用 OR 运算符将此标志与该表中查询或访问注册表值的其他标志组合。 Windows 2000: 不支持此标志。
KEY_WOW64_64KEY	表示 64 位 Windows 上的应用程序应在 64 位注册表视图上运行。 有关更多信息, 请参见访问备用注册表视图。 必须使用 OR 运算符将此标志与该表中查询或访问注册表值的其他标志组合。 Windows 2000: 不支持此标志。
KEY_WRITE	合并 STANDARD_RIGHTS_WRITE, KEY_SET_VALUE 和 KEY_CREATE_SUB_KEY 访问权限。

注意

有关更多信息, 请参见[注册表项安全性和访问权限](#)。

nOptions (可选)

默认值: REG_OPTION_NON_VOLATILE

可以为以下值之一:

值	描述
REG_OPTION_BACKUP_RESTORE	如果设置了此标志, 则该函数将忽略 samDesired 参数, 并尝试使用备份或还原密钥所需的访问权限来打开密钥。 如果调用线程启用了 SE_BACKUP_NAME 特权, 则将以 ACCESS_SYSTEM_SECURITY 和 KEY_READ 访问权限打开密钥。 如果调用线程启用了 SE_RESTORE_NAME 特权, 则将以 ACCESS_SYSTEM_SECURITY 和 KEY_WRITE 访问权限打开密钥。

	如果同时启用了两个特权，则密钥具有两个特权的组合访问权限。有关更多信息，请参见以特殊特权运行。
REG_OPTION_CREATE_LINK	此键是一个符号链接。 目标路径分配给键的 L "SymbolicLinkValue" 值。 目标路径必须是绝对注册表路径。仅在绝对需要应用程序兼容性时，才应使用注册表符号链接。
REG_OPTION_NON_VOLATILE	此键不易变;这是默认值。 信息存储在文件中，并在重新启动系统时保留。 RegSaveKey 功能保存不可变的密钥。
REG_OPTION_VOLATILE	函数创建的所有键都是可变的。 信息存储在内存中，在卸载相应的注册表配置单元时不会保留。 对于 HKEY_LOCAL_MACHINE，当系统关闭时，将发生这种情况。 对于 RegLoadKey 函数加载的注册表项，当执行相应的 RegUnLoadKey 时，将发生这种情况。 RegSaveKey 函数不会保存可变密钥。 对于已存在的键，将忽略此标志。

cClass (可选)

默认= NULL

此项的用户定义的类类型。 此参数可以忽略。

返回值

注册表项的句柄 (数字)。

备注

CreateRegistryKey 函数在指定的路径中创建所有缺少的键。 应用程序可以利用此行为一次创建多个键。 例如，应用程序可以通过为 lpSubKey 参数指定以下形式的字符串，来与前三个子项同时创建四个深度的子项：“ subkey1 \ subkey2 \ subkey3 \ subkey4”

请注意，如果路径中的现有键拼写错误，此行为将导致创建不需要的键。

应用程序无法创建作为 HKEY_USERS 或 HKEY_LOCAL_MACHINE 的直接子项的键，但是它可以在 HKEY_USERS 或 HKEY_LOCAL_MACHINE 树的较低级别中创建子项。

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)[ReadRegistryKey](#)[RegistryHiveToObject](#)[RegistryValuesToObject](#)[WriteRegistryKey](#)

使用的 WinApi

[RegCreateKeyEx](#)

DeleteRegistryKey

删除指定的注册表项。

DeleteRegistryKey(nRegKey, cKeyName [, nDeleteFunc])

参数

nRegKey

从 CreateRegistryKey , OpenRegistryKey 返回的注册表句柄或以下键常量之一:

常量	描述
HKEY_CLASSES_ROOT	<p>从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。</p> <p>该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。</p> <p>HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。 有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	从属于此注册表项的注册表项定义了当前用户的首选项。 这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和

	<p>应用程序首选项的数据。使用此键可以更轻松地建立当前用户的设置。密钥映射到 HKEY_USERS 中当前用户的分支。在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项 (HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外) 都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包括在漫游用户配置文件的按用户注册表部分中。</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000：从 Windows 7 和 Windows Server 2008 R2 开始支持此键。</p>
HKEY_LOCAL_MACHINE	<p>属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。它包含用于保存当前配置数据的子项，这些子项包括即插即用信息 (Enum 分支，其中包括系统上所有硬件的完整列表)，网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。</p>
HKEY_PERFORMANCE_DATA	<p>从属于此注册表项的注册表项允许您访问性能数据。数据实际上没有存储在注册表中；注册表功能使系统从其源收集数据。</p>
HKEY_PERFORMANCE_NLSTEXT	<p>从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。这些条目不适用于 Regedit.exe 和 Regedt32.exe。</p> <p>Windows 2000：不支持此键。</p>

HKEY_PERFORMANCE_TEXT	属于此键的注册表项引用了描述美式英语计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

cKeyName

要删除的键的名称。 它必须是 nRegKey 标识的键的子键。

如果 nDeleteFunc 为 REG_DELETE_NORMAL，则该键不能具有子键。

如果 nDeleteFunc 为 REG_DELETE_SHELL，该函数将删除子项及其所有后代。

注意

键名不区分大小写。

nDeleteFunc (可选)

默认值值: REG_DELETE_NORMAL

指定要使用的 API 函数。

可以是以下值之一。

值	描述
REG_DELETE_NORMAL	该函数调用 RegDeleteKey 删除注册表项。
REG_DELETE_SHELL	该函数调用 SHDeleteKey 删除注册表项。

返回值

如果删除成功返回.T.; 否则返回.F.。

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)

[ReadRegistryKey](#)

[RegistryHiveToObject](#)

[RegistryValuesToObject](#)[WriteRegistryKey](#)

使用的 WinApi

[RegDeleteKey](#)[SHDeleteKey](#)

FindRegistryChange

监视注册表项中单独线程中的更改。

FindRegistryChange(nRegKey, cKeyName, bWatchSubtree, nFilter, cCallback)

参数

nRegKey

以下常量之一。

常量	描述
HKEY_CLASSES_ROOT	<p>从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。</p> <p>该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。</p> <p>HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。 有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。 这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。 使用此键可以更轻松地建立当前用户的设置。 密钥映射到 HKEY_USERS 中当前用户的分支。 在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使</p>

	<p>用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项 (HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外) 都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包括在漫游用户配置文件的按用户注册表部分中。</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000：从 Windows 7 和 Windows Server 2008 R2 开始支持此键。</p>
HKEY_LOCAL_MACHINE	<p>属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。它包含用于保存当前配置数据的子项，这些子项包括即插即用信息 (Enum 分支，其中包括系统上所有硬件的完整列表)，网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。</p>
HKEY_PERFORMANCE_DATA	<p>从属于此注册表项的注册表项允许您访问性能数据。数据实际上没有存储在注册表中；注册表功能使系统从其源收集数据。</p>
HKEY_PERFORMANCE_NLSTEXT	<p>从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。这些条目不适用于 Regedit.exe 和 Regedt32.exe。</p> <p>Windows 2000：不支持此键。</p>
HKEY_PERFORMANCE_TEXT	<p>属于此键的注册表项引用了描述美式英语计数器的文本字符串。这些条目不适用于 Regedit.exe 和 Regedt32.exe。</p>

	Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

cKeyName

要监视的注册表子项的名称。

注意

键名不区分大小写。

bWatchSubtree

如果此参数为.T., 则该函数报告指定键及其子键的更改。如果参数为.F., 则该功能仅报告指定键中的更改。

nFilter

一个值, 指示应报告的更改。

此参数可以是以下值之一或组合。

值	含义
REG_NOTIFY_CHANGE_NAME	通知呼叫者是否添加或删除了一个子项。
REG_NOTIFY_CHANGE_ATTRIBUTES	通知调用者键属性的更改, 例如安全描述符信息。
REG_NOTIFY_CHANGE_LAST_SET	通知调用者键值的更改。这可以包括添加或删除值, 或更改现有值。
REG_NOTIFY_CHANGE_SECURITY	通知调用者键安全描述符的更改。

cCallback

当监视的注册表项或子树中出现过滤条件之一时调用的函数。

该函数应具有以下形式:

```
Function RegistryKeyChanged
    Lparameters hHandle, nError
    If nError = 0
        ? '键已更改。'
    Else
        ? 'API 函数失败 - 错误编号: ', nError
    Endif
Endfunc
```

返回值

表示监视注册表项的线程的数字句柄。

参考

[ARegistryKeys](#)[ARegistryValues](#)[CancelRegistryChange](#)[CloseRegistryKey](#)[CreateRegistryKey](#)[DeleteRegistryKey](#)[OpenRegistryKey](#)[ReadRegistryKey](#)[RegistryHiveToObject](#)[RegistryValuesToObject](#)[WriteRegistryKey](#)

使用的 WinApi

[RegNotifyChangeKeyValue](#)[RegOpenKeyEx](#)[CreateThread](#)[WaitForMultipleObjects](#)[PostMessage](#)

OpenRegistryKey

打开指定的注册表项。

OpenRegistryKey(nRegKey, cKeyName [, nAccessRights])

参数

nRegKey

从 OpenRegistryKey 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。 该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其

	<p>OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。使用此键可以更轻松地建立当前用户的设置。密钥映射到 HKEY_USERS 中当前用户的分支。在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项 (HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外) 都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包括在漫游用户配置文件的按用户注册表部分中。

	Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000: 从 Windows 7 和 Windows Server 2008 R2 开始支持此键。
HKEY_LOCAL_MACHINE	属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。 它包含用于保存当前配置数据的子项，这些子项包括即插即用信息（Enum 分支，其中包括系统上所有硬件的完整列表），网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。
HKEY_PERFORMANCE_DATA	从属于此注册表项的注册表项允许您访问性能数据。 数据实际上没有存储在注册表中； 注册表功能使系统从其源收集数据。
HKEY_PERFORMANCE_NLSTEXT	从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。 这些条目不适用于Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_PERFORMANCE_TEXT	属于此键的注册表项引用了描述美式英语计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

cKeyName

nRegKey 中传递的键的子键的路径。

nAccessRights (可选)

默认值：KEY_ALL_ACCESS

以下值之一或组合。

常量	描述
KEY_ALL_ACCESS	合并 STANDARD_RIGHTS_REQUIRED, KEY_QUERY_VALUE, KEY_SET_VALUE, KEY_CREATE_SUB_KEY, KEY_ENUMERATE_SUB_KEYS, KEY_NOTIFY 和 KEY_CREATE_LINK 访问权限。
KEY_CREATE_LINK	保留供系统使用。
KEY_CREATE_SUB_KEY	创建注册表项的子项所必需。
KEY_ENUMERATE_SUB_KEYS	枚举注册表项的子项所必需。
KEY_EXECUTE	等效于 KEY_READ。
KEY_NOTIFY	要求为注册表项或注册表项的子项请求更改通知。
KEY_QUERY_VALUE	查询注册表项的值所必需。

KEY_READ	合并 STANDARD_RIGHTS_READ, KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS 和 KEY_NOTIFY 值。
KEY_SET_VALUE	创建, 删除或设置注册表值所必需。
KEY_WOW64_32KEY	表示 64 位 Windows 上的应用程序应在 32 位注册表视图上运行。 有关更多信息, 请参见访问备用注册表视图。 必须使用 OR 运算符将此标志与该表中查询或访问注册表值的其他标志组合。 Windows 2000: 不支持此标志。
KEY_WOW64_64KEY	表示 64 位 Windows 上的应用程序应在 64 位注册表视图上运行。 有关更多信息, 请参见访问备用注册表视图。 必须使用 OR 运算符将此标志与该表中查询或访问注册表值的其他标志组合。 Windows 2000: 不支持此标志。
KEY_WRITE	合并 STANDARD_RIGHTS_WRITE, KEY_SET_VALUE 和 KEY_CREATE_SUB_KEY 访问权限。

注意

有关更多信息, 请参见[注册表项安全性和访问权限](#)。

返回值

注册表项的句柄。

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[ReadRegistryKey](#)

[RegistryHiveToObject](#)

[RegistryValuesToObject](#)

[WriteRegistryKey](#)

使用的 WinApi

[RegOpenKeyEx](#)

ReadRegistryKey

检索指定注册表值的数据。

ReadRegistryKey(nRegKey [, cValueName [, cKeyName [, cValueType]]])

参数

nRegKey

从 [OpenRegistryKey](#) 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	<p>从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。</p> <p>该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。</p> <p>HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。 有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。 这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。 使用此键可以更轻松地建立当前用户的设置。 密钥映射到 HKEY_USERS 中当前用户的分支。 在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。 例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。 该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。 如果此安全上下文没有在 HKEY_USERS 中加载注册</p>

	<p>表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项 (HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外) 都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包括在漫游用户配置文件的按用户注册表部分中。</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000：从 Windows 7 和 Windows Server 2008 R2 开始支持此键。</p>
HKEY_LOCAL_MACHINE	<p>属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。它包含用于保存当前配置数据的子项，这些子项包括即插即用信息（Enum 分支，其中包括系统上所有硬件的完整列表），网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。</p>
HKEY_PERFORMANCE_DATA	<p>从属于此注册表项的注册表项允许您访问性能数据。数据实际上没有存储在注册表中；注册表功能使系统从其源收集数据。</p>
HKEY_PERFORMANCE_NLSTEXT	<p>从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。这些条目不适用于 Regedit.exe 和 Regedt32.exe。</p> <p>Windows 2000：不支持此键。</p>
HKEY_PERFORMANCE_TEXT	<p>属于此键的注册表项引用了描述美式英语计数器的文本字符串。这些条目不适用于 Regedit.exe 和 Regedt32.exe。</p> <p>Windows 2000：不支持此键。</p>
HKEY_USERS	<p>从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。</p>

cValueName

要为其返回数据的值的名称。

如果传递空字符串，则返回指定键的默认值。

cKeyName

nRegKey 中传递的键的子键的路径。

cValueType (可选)

注册表值应转换为的 VFP 数据类型。

如果省略 cValueType，则将从注册表中检索它。

下表列出了注册表到 VFP 类型的默认转换。

注册表类型	VFP 类型
REG_SZ, REG_MULTI_SZ, REG_EXPAND_SZ, REG_LINK and REG_NONE	C
REG_BINARY	Q **
REG_DWORD, REG_DWORD_BIG_ENDIAN, REG_DOUBLE	N
REG_INTEGER	I
REG_LOGICAL	L
REG_DATE	D
REG_DATETIME	T
REG_QWORD, REG_MONEY	Y

如果您传递 cValueType，则下表列出了允许的转换。

注册表类型	cValueType
REG_SZ	C, Q
REG_EXPAND_SZ	C, Q
REG_MULTI_SZ	C, Q
REG_BINARY	C, Q
REG_DWORD	N, I, L, Q
REG_DWORD_BIG_ENDIAN	N, I, L, Q
REG_QWORD	Y, Q, C, N
REG_INTEGER	I, N, L, Q
REG_LOGICAL	I, N, L, Q
REG_DOUBLE	N, D, T, Q
REG_DATE	D, T, Q
REG_DATETIME	D, T, Q
REG_MONEY	Y, Q

注意

** 由于无法从 FLL 返回 varbinary (Q) 值，因此该值将作为普通字符串返回。

您可以使用 CREATEBINARY () 将其转换为 varbinary 值。

返回值

注册表值的数据。

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)

[RegistryHiveToObject](#)

[RegistryValuesToObject](#)

[WriteRegistryKey](#)

使用的 WinApi

[RegOpenKeyEx](#)

[RegQueryValueEx](#)

[RegCloseKey](#)

RegistryHiveToObject

将包括所有子项的注册表项存储到对象中。

RegistryHiveToObject(nRegKey, cKeyName, oObject)

参数

nRegKey

从 [OpenRegistryKey](#) 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。

	<p>该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。使用此键可以更轻松地建立当前用户的设置。密钥映射到 HKEY_USERS 中当前用户的分支。在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项（HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外）都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包括在漫游用户配置文件的按用户注册表部分中。

	Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000: 从 Windows 7 和 Windows Server 2008 R2 开始支持此键。
HKEY_LOCAL_MACHINE	属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。 它包含用于保存当前配置数据的子项，这些子项包括即插即用信息（Enum 分支，其中包括系统上所有硬件的完整列表），网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。
HKEY_PERFORMANCE_DATA	从属于此注册表项的注册表项允许您访问性能数据。 数据实际上没有存储在注册表中； 注册表功能使系统从其源收集数据。
HKEY_PERFORMANCE_NLSTEXT	从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_PERFORMANCE_TEXT	属于此键的注册表项引用了描述美式英语计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

cKeyName

nRegKey 中传递的键的子键的路径。

oObject

用与读取的注册表值相对应的属性扩展的对象。

注意

由于注册表项和值名称可以包含对于 VFP 属性名称无效的字符，因此该函数将转换名称。

转换方案如下：

- 1.如果值名称以数字开头，则下划线（_）开头
- 2.每个无效字符（均预期为 0-9, a-z, A-Z 和_）都替换为下划线

例如 “ 2.SomeName” -> “ _2_SomeName” , “ {HelloWorld}” -> “ _HelloWorld_”

返回值

.T.

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)

[DeleteRegistryKey](#)

[FindRegistryChange](#)

[OpenRegistryKey](#)

[ReadRegistryKey](#)

[RegistryValuesToObject](#)

[WriteRegistryKey](#)

使用的 WinApi

[RegOpenKeyEx](#)

[RegQueryInfoKey](#)

[RegEnumValue](#)

[RegEnumKeyEx](#)

[RegCloseKey](#)

RegistryValuesToObject

将注册表项的所有值存储到一个对象中。

RegistryValuesToObject(nRegKey, cKeyName, oObject)

参数

nRegKey

从 [OpenRegistryKey](#) 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。

	<p>该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。文件查看器和用户界面扩展将其 OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。使用此键可以更轻松地建立当前用户的设置。密钥映射到 HKEY_USERS 中当前用户的分支。在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项 (HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外) 都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包括在漫游用户配置文件的按用户注册表部分中。

	Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000: 从 Windows 7 和 Windows Server 2008 R2 开始支持此键。
HKEY_LOCAL_MACHINE	属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。 它包含用于保存当前配置数据的子项，这些子项包括即插即用信息（Enum 分支，其中包括系统上所有硬件的完整列表），网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。
HKEY_PERFORMANCE_DATA	从属于此注册表项的注册表项允许您访问性能数据。 数据实际上没有存储在注册表中； 注册表功能使系统从其源收集数据。
HKEY_PERFORMANCE_NLSTEXT	从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。 这些条目不适用于Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_PERFORMANCE_TEXT	属于此键的注册表项引用了描述美式英语计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

cKeyName

nRegKey 中传递的键的子键的路径。

oObject

用与读取的注册表值相对应的属性扩展的对象。

注意

由于注册表项和值名称可以包含对于 VFP 属性名称无效的字符，因此该函数将转换名称。

转换方案如下：

1.如果值名称以数字开头，则下划线（_）开头

2.每个无效字符（均预期为 0-9, a-z, A-Z 和_）都替换为下划线

例如 “ 2.SomeName” -> “ _2_SomeName” , “ {HelloWorld}” -> “ _HelloWorld_”

返回值

添加到对象的注册表值的数量。

参考

[ARegistryKeys](#)[ARegistryValues](#)[CancelRegistryChange](#)[CloseRegistryKey](#)[CreateRegistryKey](#)[DeleteRegistryKey](#)[FindRegistryChange](#)[OpenRegistryKey](#)[ReadRegistryKey](#)[RegistryHiveToObject](#)[WriteRegistryKey](#)

使用的 WinApi

[RegOpenKeyEx](#)[RegQueryInfoKey](#)[RegEnumValue](#)[RegCloseKey](#)

WriteRegistryKey

在注册表项下设置数据和指定值的类型。

WriteRegistryKey(nRegKey, vValue [, cValueName [, cKeyName [, nValueType]]])

参数

nRegKey

从 [OpenRegistryKey](#) 返回的注册表句柄或以下键常量之一。

常量	描述
HKEY_CLASSES_ROOT	从属于此键的注册表项定义了文档的类型（或类）以及与这些类型关联的属性。 Shell 和 COM 应用程序使用此密钥下存储的信息。 该密钥还通过存储 DDE 和 OLE 支持的信息来提供与 Windows 3.1 注册数据库的向后兼容性。 文件查看器和用户界面扩展将其

	<p>OLE 类标识符存储在 HKEY_CLASSES_ROOT 中，并且进程内服务器已在此密钥中注册。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。</p>
HKEY_CURRENT_CONFIG	<p>包含有关本地计算机系统的当前硬件配置文件的信息。HKEY_CURRENT_CONFIG 下的信息仅描述当前硬件配置和标准配置之间的差异。有关标准硬件配置的信息存储在 HKEY_LOCAL_MACHINE 的软件和系统键下。</p> <p>HKEY_CURRENT_CONFIG 是 HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Hardware Profiles \ Current 的别名。</p>
HKEY_CURRENT_USER	<p>从属于此注册表项的注册表项定义了当前用户的首选项。这些首选项包括环境变量的设置，有关程序组，颜色，打印机，网络连接和应用程序首选项的数据。使用此键可以更轻松地建立当前用户的设置。密钥映射到 HKEY_USERS 中当前用户的分支。在 HKEY_CURRENT_USER 中，软件供应商存储要在其应用程序中使用的当前特定于用户的首选项。例如，Microsoft 创建 HKEY_CURRENT_USER \ Software \ Microsoft 密钥供其应用程序使用，每个应用程序都在 Microsoft 密钥下创建自己的子密钥。</p> <p>HKEY_CURRENT_USER 和 HKEY_USERS 之间的映射是针对每个进程的，并且是在该进程首次引用 HKEY_CURRENT_USER 时建立的。该映射基于引用 HKEY_CURRENT_USER 的第一个线程的安全上下文。如果此安全上下文没有在 HKEY_USERS 中加载注册表配置单元，则使用 HKEY_USERS \ .Default 建立映射。建立此映射后，即使线程的安全上下文发生更改，它也将保留。</p> <p>HKEY_CURRENT_USER 中的所有注册表项 (HKEY_CURRENT_USER \ Software \ Classes 下的注册表项除外) 都包含在漫游用户配置文件的按用户注册表部分中。要从漫游用户配置文件中排除其他条目，请将它们存储在 HKEY_CURRENT_USER_LOCAL_SETTINGS 中。</p> <p>不应在模拟其他用户的服务或应用程序中使用此句柄。而是，调用 RegOpenCurrentUser 函数。</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	从属于此注册表项的注册表项定义了计算机本地的当前用户的首选项。这些条目不包括在漫游用户配置文件的按用户注册表部分中。

	Windows Server 2008, Windows Vista, Windows Server 2003 和 Windows XP / 2000: 从 Windows 7 和 Windows Server 2008 R2 开始支持此键。
HKEY_LOCAL_MACHINE	属于此键的注册表项定义了计算机的物理状态，包括有关总线类型，系统内存以及已安装的硬件和软件的数据。 它包含用于保存当前配置数据的子项，这些子项包括即插即用信息（Enum 分支，其中包括系统上所有硬件的完整列表），网络登录首选项，网络安全信息，与软件相关的信息（例如（如服务器名称和服务器位置），以及其他系统信息。
HKEY_PERFORMANCE_DATA	从属于此注册表项的注册表项允许您访问性能数据。 数据实际上没有存储在注册表中； 注册表功能使系统从其源收集数据。
HKEY_PERFORMANCE_NLSTEXT	从属于此注册表项的注册表项引用了以计算机系统运行所在区域的本地语言描述计数器的文本字符串。 这些条目不适用于Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_PERFORMANCE_TEXT	属于此键的注册表项引用了描述美式英语计数器的文本字符串。 这些条目不适用于 Regedit.exe 和 Regedt32.exe。 Windows 2000: 不支持此键。
HKEY_USERS	从属于此注册表项的注册表项定义本地计算机上新用户的默认用户配置以及当前用户的用户配置。

vValue:

要写入注册表的值。

可以是除 NULL 之外的任何类型。

cValueName (可选)

值名称，应在其下存储值。

如果省略 cValueName 或传递空字符串，则该值将存储到指定键的默认值中。

cKeyName (可选)

nRegKey 中传递的键的子键的路径。

例如

```
&& write to HKEY_LOCAL_MACHINE\SOFTWARE\YourFirm\YourApp\Settings
WriteRegistryKey(HKEY_LOCAL_MACHINE, 'SomeOption', 'NameOfOption',
'SOFTWARE\YourFirm\YourApp\Settings')
```

nValueType (可选)

注册表值的数据类型。

如果省略数据类型，则由 vValue 参数的类型确定。

下表显示了 VFP 到注册表类型的默认映射。

FoxPro 类型	注册表类型
C	REG_SZ
C binary **	REG_BINARY
I	REG_INTEGER ***
N	REG_DOUBLE ***
D	REG_DATE ***
T	REG_DATETIME ***
L	REG_LOGICAL ***
Y	REG_MONEY ***

注意

** 来自 C NOCPTRANS, M NOCPTRANS, Q 或 W 字段, 或者来自使用 CREATEBINARY ()

创建的表达式。

注意

*** 这些类型是自定义的。 如果您在调用 [ReadRegistryKey](#) 和 [WriteRegistryKey](#) 时始终忽略 nValueType 参数, 则将始终获取确切的 VFP 值。

FoxPro 类型	注册表类型
C, Q	All
I	REG_DWORD, REG_QWORD, REG_DOUBLE, REG_INTEGER, REG_BINARY
N	REG_DWORD, REG_QWORD, REG_DOUBLE, REG_INTEGER, REG_BINARY
D	REG_DATE, REG_DOUBLE, REG_BINARY
T	REG_DATETIME, REG_DOUBLE, REG_BINARY
L	REG_LOGICAL, REG_DWORD, REG_INTEGER, REG_BINARY
Y	REG_MONEY, REG_QWORD, REG_BINARY

返回值

.T.

参考

[ARegistryKeys](#)

[ARegistryValues](#)

[CancelRegistryChange](#)

[CloseRegistryKey](#)

[CreateRegistryKey](#)[DeleteRegistryKey](#)[FindRegistryChange](#)[OpenRegistryKey](#)[ReadRegistryKey](#)[RegistryHiveToObject](#)[RegistryValuesToObject](#)

使用的 WinApi

[RegSetValueEx](#)[RegOpenKeyEx](#)[RegCloseKey](#)

资源信息

检索有关模块 (dll 和可执行文件) 中资源的信息。

AResourceLanguages	将与二进制模块关联的指定类型和名称的特定于语言的资源的信息存储到数组中。
AResourceNames	将有关二进制模块中指定类型的资源的信息存储到数组中。
AResourceTypes	将有关二进制模块内资源类型的信息存储到数组中。

AResourceLanguages

将与二进制模块关联的指定类型和名称的特定于语言的资源的信息存储到数组中。

AResourceLanguages(cArrayName, nModule, cnResourceType, cnResourceName)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	资源语言 ID。.	N

nModule

要搜索的模块的句柄。 从 Windows Vista 开始，如果这是与语言无关的可移植可执行文件 (LN

文件), 则搜索中将包括适当的.mui 文件 (如果存在)。 如果这是一个特定的.mui 文件, 则仅在该文件中搜索资源。

如果此参数为 0, 则等效于将句柄传递给用于创建当前进程的模块。

cnResourceType

被枚举的语言的资源类型。

或者, 此参数可以是表示预定义资源类型的整数值, 而不是字符串。

cnResourceName

为其枚举语言的资源的名称。

或者, 此参数可以是资源的整数标识符, 而不是字符串。

返回值

资源语言的数量。

参考

[AResourceNames](#)

[AResourceTypes](#)

使用的 WinApi

[EnumResourceLanguages](#)

AResourceNames

将有关二进制模块中指定类型的资源的信息存储到数组中。

AResourceNames(cArrayName, nModule, cnResourceType)

参数

cArrayName

返回时, 数组包含以下信息。

列	内容	数据类型
1	资源名称或 ID。	C or N

nModule

要搜索的模块的句柄。 从 Windows Vista 开始, 如果这是与语言无关的可移植可执行文件 (LN 文件), 则搜索中将包括适当的.mui 文件 (如果存在)。 如果这是一个特定的.mui 文件, 则仅在该文件中

搜索资源。

如果此参数为 0，则等效于将句柄传递给用于创建当前进程的模块。

cnResourceType

被枚举的语言的资源类型。

或者，此参数可以是表示预定义资源类型的整数值，而不是字符串。

返回值

资源名称的数量。

参考

[AResourceLanguages](#)

[AResourceTypes](#)

使用的 WinApi

[EnumResourceNames](#)

AResourceTypes

将有关二进制模块内资源类型的信息存储到数组中。

AResourceTypes(cArrayName, nHMODULE)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	资源名称或 ID。	C or N

nModule

要搜索的模块的句柄。 从 Windows Vista 开始，如果这是与语言无关的可移植可执行文件 (LN 文件)，则搜索中将包括适当的.mui 文件（如果存在）。 如果这是一个特定的.mui 文件，则仅在该文件中搜索资源。

如果此参数为 0，则等效于将句柄传递给用于创建当前进程的模块。

返回值

资源类型的数量。

参考

[AResourceLanguages](#)

[AResourceNames](#)

[AResourceTypes](#)

使用的 WinApi

[EnumResourceTypes](#)

服务管理

检索信息并控制 Windows 服务。

ADependentServices	将所有依赖于传入服务的服务存储到一个数组中。
AServiceConfig	将指定的 Windows 服务的配置参数存储到数组中。
AServices	将有关 Windows 服务的信息存储到数组中。
AServiceStatus	将有关指定服务的当前状态的信息存储到数组中。
CloseServiceHandle	关闭提供的服务句柄 (SC_HANDLE)。
ContinueService	将继续请求发送到指定的 Windows 服务。
ControlService	将自定义控制请求发送到指定的 Windows 服务。
CreateService	安装 Windows 服务。
OpenService	打开现有服务。
PauseService	将暂停请求发送到指定的服务。
StartService	启动 Windows 服务。
StopService	将停止请求发送到指定的服务。
WaitForServiceStatus	监视 Windows 服务以达到指定状态。

ADependentServices

将所有依赖于传入服务的服务存储到一个数组中。

```
ADependentServices(cArrayName, cServiceName | nServiceHandle [, cServer [, cDatabase]])
```

参数

cArrayName

数组的名称，作为函数应将信息存储在其中的字符串。

返回时，该数组包含来自 [ENUM SERVICE STATUS](#) 和 [SERVICE STATUS](#) 结构的以下信息。

列	描述
1	服务名
2	显示名称
3	服务类型
4	当前状态
5	Win32ExitCode
6	ServiceSpecificExitCode
7	CheckPoint
8	ControlsAccepted

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

依赖服务的数量。

参考

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[OpenService](#)

[PauseService](#)
[StartService](#)
[StopService](#)
[WaitForServiceStatus](#)

使用的 WinApi

[EnumDependentServices](#)
[OpenSCManager](#)
[OpenService](#)
[CloseServiceHandle](#)

AServiceConfig

将指定的 Windows 服务的配置参数存储到数组中。

AServiceConfig(cArrayName, cServiceName | nServiceHandle [, cServer [, cDatabase]])

参数

cArrayName

返回时，该数组包含来自 QUERY_SERVICE_CONFIG 结构的以下信息。

元素	内容
1	服务类型
2	启动类型
3	错误控制
4	二进制路径名
5	负载顺序组
6	TagId
7	依赖
8	服务启动名称
9	显示名称

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

.T.

参考

[ADependentServices](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[OpenService](#)

[PauseService](#)

[StartService](#)

[StopService](#)

[WaitForServiceStatus](#)

使用的 WinApi

[QueryServiceConfig](#)

[OpenSCManager](#)

[OpenService](#)

[CloseServiceHandle](#)

AServices

将有关 Windows 服务的信息存储到数组中。

AServices(cArrayName, [cServer [, cDatabase [, nServiceState [, nServiceType]]]])

参数

cArrayName

返回时，该数组包含来自 [ENUM_SERVICE_STATUS_PROCESS](#) 或 [ENUM_SERVICE_STATUS](#) 结构的以下信息。

列	内容
1	服务名称
2	显示名称
3	服务类型
4	当前状态
5	Win32ExitCode
6	ServiceSpecificExitCode
7	CheckPoint
8	ControlsAccepted
9	服务标记
10	ProcessId

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

nServiceState (可选)

默认值: SERVICE_STATE_ALL

可以是以下值之一。

ServiceState	描述
SERVICE_ACTIVE	仅枚举运行的服务。
SERVICE_INACTIVE	仅枚举已停止的服务。
SERVICE_STATE_ALL	枚举所有服务

nServiceType (可选, 附加)

默认值: SERVICE_WIN32

可以是以下值之一或组合。

ServiceType	描述
SERVICE_DRIVER	枚举类型为 SERVICE_KERNEL_DRIVER 和 SERVICE_FILE_SYSTEM_DRIVER 的服务。
SERVICE_WIN32	枚举类型为 SERVICE_WIN32_OWN_PROCESS 和 SERVICE_WIN32_SHARE_PROCESS 的服务。

返回值

服务的数量。

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[OpenService](#)

[PauseService](#)

[StartService](#)

[StopService](#)

[WaitForServiceStatus](#)

使用的 WinApi

[EnumServicesStatusEx \(如果支持\)](#)

[EnumServicesStatus](#)

[OpenSCManager](#)

[CloseServiceHandle](#)

AServiceStatus

将有关指定服务的当前状态的信息存储到数组中。

AServiceStatus(cArrayName, cServiceName | nServiceHandle [, cServer [, cDatabase]])

参数

cArrayName

返回时，该数组包含来自 [SERVICE_STATUS](#) 结构的以下信息。

元素	内容
1	CheckPoint
2	ControlsAccepted
3	Currentstate
4	ServiceSpecificExitCode
5	Servicetype
6	WaitHint
7	Win32ExitCode

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

.T.

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[OpenService](#)

[PauseService](#)
[StartService](#)
[StopService](#)
[WaitForServiceStatus](#)

使用的 WinApi

[QueryServiceStatus](#)
[OpenSCManager](#)
[OpenService](#)
[CloseServiceHandle](#)

CloseServiceHandle

关闭提供的服务句柄 (SC_HANDLE)。

CloseServiceHandle(nServiceHandle)

参数

nServiceHandle

从 [OpenService](#) 检索的 Windows 服务的句柄。

返回值

,T,

参考

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[ContinueService](#)
[ControlService](#)
[CreateService](#)
[OpenService](#)
[PauseService](#)

[StartService](#)[StopService](#)[WaitForServiceStatus](#)

使用的 WinApi

[CloseServiceHandle](#)

ContinueService

将继续请求发送到指定的 Windows 服务。

ContinueService(cServiceName | nServiceHandle [, nTimeout [, cServer [, cDatabase]]])

参数

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

nTimeout (可选)

服务处于 SERVICE_START_PENDING 状态时等待的最长时间 (以秒为单位)。

如果您将超时值设为 0，则函数不会等到服务启动后才开始，而是在发送启动请求后立即返回。

如果省略此参数或传递 NULL，则将超时设置为服务报告的默认超时。

请参阅 MSDN 帮助以获取 [SERVICE_STATUS_PROCESS](#) 结构的 “dwWaitHint” 成员。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

如果服务已经处于运行状态或在指定的超时时间内继续运行，则返回 1；如果超时，则返回 0。

参考

[ADependentServices](#)[AServiceConfig](#)

[AServices](#)[AServiceStatus](#)[CloseServiceHandle](#)[ControlService](#)[CreateService](#)[OpenService](#)[PauseService](#)[StartService](#)[StopService](#)[WaitForServiceStatus](#)

使用的 WinApi

[ControlService](#)[QueryServiceStatus](#)[OpenSCManager](#)[OpenService](#)[CloseServiceHandle](#)

ControlService

将自定义控制请求发送到指定的 Windows 服务。

ControlService(cServiceName | nServiceHandle, nControlCode [, cServer [, cDatabase]])

参数

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

nControlCode

要发送到服务的自定义控制代码。

有效的控制代码在 128 到 255 之间。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

.T.

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[CreateService](#)

[OpenService](#)

[PauseService](#)

[StartService](#)

[StopService](#)

[WaitForServiceStatus](#)

使用的 WinApi

[ControlService](#)

[OpenSCManager](#)

[OpenService](#)

[CloseServiceHandle](#)

CreateService

安装 Windows 服务。

CreateService(cServiceName, cDisplayName, cExecutable [, nServiceType [, nStartType [,

nErrorControl [, cLoadOrderGroup [, cDependencies [, cServiceAccount [, cAccountPassword [, cMachine [, cDatabase]]]]]]])

参数

cServiceName

服务的简称。

cDisplayName

服务的描述性名称。

cExecutable

服务可执行文件的完整路径。

nServiceType (可选)

默认值: SERVICE_WIN32_OWN_PROCESS

服务类型。

此参数可以是以下值之一。

服务类型	描述
SERVICE_FILE_SYSTEM_DRIVER	文件系统驱动程序服务。
SERVICE_KERNEL_DRIVER	驱动服务
SERVICE_WIN32_OWN_PROCESS	在自己的进程中运行的服务。
SERVICE_WIN32_SHARE_PROCESS	与一个或多个其他服务共享一个进程的服务。
SERVICE_INTERACTIVE_PROCESS	如果您指定 SERVICE_WIN32_OWN_PROCESS 或 SERVICE_WIN32_SHARE_PROCESS, 并且该服务在 LocalSystem 帐户的上下文中运行, 则也可以指定此值。

nStartType (可选)

默认值: SERVICE_AUTO_START

服务启动选项。

此参数可以是以下值之一。

启动类型	描述
SERVICE_AUTO_START	在系统启动期间, 服务控制管理器会自动启动服务。
SERVICE_BOOT_START	由系统加载程序启动的设备驱动程序。 此值仅对驱动程序服务有效。
SERVICE_DEMAND_START	当进程调用 StartService 函数时, 由服务控制管理器启动的服务。
SERVICE_DISABLED	无法启动的服务。 尝试启动服务会导致错误代码 ERROR_SERVICE_DISABLED。
SERVICE_SYSTEM_START	由 IoInitSystem 函数启动的设备驱动程序。 此值仅对驱动程序服务有效。

nErrorControl

默认值: SERVICE_ERROR_NORMAL

如果此服务无法启动，则错误的严重性和采取的措施。

参数可以是以下值之一。

错误控制	描述
SERVICE_ERROR_CRITICAL	如果可能，启动程序将错误记录在事件日志中。 如果正在启动最后一个正确的配置，则启动操作将失败。 否则，将使用最新的正确配置重新启动系统。
SERVICE_ERROR_IGNORE	启动程序将忽略该错误并继续启动操作。
SERVICE_ERROR_NORMAL	启动程序将错误记录在事件日志中，但继续启动操作。
SERVICE_ERROR_SEVERE	启动程序将错误记录在事件日志中。 如果正在启动最后一个正确的配置，则启动操作将继续。 否则，将使用最新的正确配置重新启动系统。

cLoadOrderGroup (可选)

默认值: NULL

cDependencies (可选)**cServiceAccount (可选)****cAccountPassword (可选)****cMachine (可选)****cDatabase (可选)****返回值**

.T.

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[OpenService](#)

[PauseService](#)
[StartService](#)
[StopService](#)
[WaitForServiceStatus](#)

使用的 WinApi

[OpenSCManager](#)
[CreateService](#)
[CloseServiceHandle](#)

OpenService

打开现有服务。

OpenService(cServiceName [, nAccess [, cServer [, cDatabase]]])

参数

cServiceName | nServiceHandle

服务的名称或 OpenService 函数返回的数字句柄。

nAccess (可选, 可添加)

默认值: SERVICE_ALL_ACCESS

所需的服务访问权限。

以下值之一或组合。

访问权限	描述
SERVICE_ALL_ACCESS	除此表中的所有访问权限外, 还包括 STANDARD_RIGHTS_REQUIRED。
SERVICE_CHANGE_CONFIG	调用 ChangeServiceConfig 或 ChangeServiceConfig2 函数来更改服务配置是必需的。因为这授予调用方更改系统运行的可执行文件的权限, 所以应仅将其授予管理员。
SERVICE_ENUMERATE_DEPENDENTS	调用 ADependentServices 函数以枚举依赖于该服务的所有服务所必需的。
SERVICE_INTERROGATE	要求调用 ControlService 函数以要求服务立即报告其状态。
SERVICE_PAUSE_CONTINUE	是调用 PauseService 和 ContinueService 函数必需的。
SERVICE_QUERY_CONFIG	调用 AServiceConfig 函数以查询服务配置所必需。
SERVICE_QUERY_STATUS	要求调用 AServiceStatus 函数向服务控制管理器询问服务状态。

	态。
SERVICE_START	调用 StartService 函数以启动服务所必需。
SERVICE_STOP	调用 ControlService 函数以停止服务是必需的。
SERVICE_USER_DEFINED_CONTROL	调用 ControlService 函数以指定用户定义的控制代码是必需的。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

Windows 服务句柄。

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[PauseService](#)

[StartService](#)

[StopService](#)

[WaitForServiceStatus](#)

使用的 WinApi

[OpenService](#)

[OpenSCManager](#)

PauseService

将暂停请求发送到指定的服务。

PauseService(cServiceName | nServiceHandle [, nTimeout [, cServer [, cDatabase]]])

参数

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

nTimeout (可选)

服务处于 SERVICE_START_PENDING 状态时等待的最长时间 (以秒为单位)。

如果您将超时值设为 0，则函数不会等到服务启动后才开始，而是在发送启动请求后立即返回。

如果省略此参数或传递 NULL，则将超时设置为服务报告的默认超时。

请参阅 MSDN 帮助以获取 [SERVICE_STATUS_PROCESS](#) 结构的“dwWaitHint”成员。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

如果服务在超时时间内暂停，则为 1，否则为 0。

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[OpenService](#)[StartService](#)[StopService](#)[WaitForServiceStatus](#)

使用的 WinApi

[ControlService](#)[OpenSCManager](#)[OpenService](#)[CloseServiceHandle](#)

StartService

启动 Windows 服务。

StartService(cServiceName | nServiceHandle [, @aArguments [, nTimeout [, cServer [, cDatabase]]]])

参数

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

@aArguments (可选)

作为参数传递给服务的字符串数组。

如果要省略此参数，并传递后面的参数，则可以传递 NULL。

nTimeout (可选)

服务处于 SERVICE_START_PENDING 状态时等待的最长时间（以秒为单位）。

如果您将超时值设为 0，则函数不会等到服务启动后才开始，而是在发送启动请求后立即返回。

如果省略此参数或传递 NULL，则将超时设置为服务报告的默认超时。

请参阅 MSDN 帮助以获取 [SERVICE_STATUS_PROCESS](#) 结构的“dwWaitHint”成员。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

如果服务已成功启动或已经在运行，则为 1；如果在服务切换到运行状态之前已超过超时间隔，则为 0。

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[OpenService](#)

[PauseService](#)

[StopService](#)

[WaitForServiceStatus](#)

使用的 WinApi

[StartService](#)

[QueryServiceStatus](#)

[OpenSCManager](#)

[OpenService](#)

StopService

将停止请求发送到指定的服务。

StopService(cServiceName | nServiceHandle [, nTimeout [, bStopDependantServices [,

cServer [, cDatabase]]])

参数

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

nTimeout (可选)

服务处于 SERVICE_START_PENDING 状态时等待的最长时间 (以秒为单位)。

如果您将超时值设为 0，则函数不会等到服务启动后才开始，而是在发送启动请求后立即返回。

如果省略此参数或传递 NULL，则将超时设置为服务报告的默认超时。

请参阅 MSDN 帮助以获取 [SERVICE_STATUS_PROCESS](#) 结构的 “ dwWaitHint” 成员。

bStopDependantServices (可选)

默认值： .F.

如果为.T.，依赖于传递的服务的服务将首先停止。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值

如果服务成功停止或已经停止，则为 1；如果服务切换为停止状态之前已超过超时间隔，则为 0。

参考

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)[OpenService](#)[PauseService](#)[StartService](#)[WaitForServiceStatus](#)

使用的 WinApi

[ControlService](#)[EnumDependentServices](#)[QueryServiceStatus](#)[OpenSCManager](#)[OpenService](#)

WaitForServiceStatus

监视 Windows 服务以达到指定状态。

WaitForServiceStatus(cServiceName | nServiceHandle, nStatus [, nTimeout [, cServer [, cDatabase]]])

参数

cServiceName | nServiceHandle

服务的名称或 [OpenService](#) 函数返回的数字句柄。

nStatus

等待的服务状态。

可以是以下值之一。

状态	描述
SERVICE_STOPPED	服务未运行。
SERVICE_START_PENDING	服务正在启动。
SERVICE_STOP_PENDING	服务正在停止。
SERVICE_RUNNING	服务正在运行。
SERVICE_CONTINUE_PENDING	服务继续挂起。
SERVICE_PAUSE_PENDING	服务暂停未决。

SERVICE_PAUSED	服务已暂停。
----------------	--------

nTimeout (可选)

服务处于 SERVICE_START_PENDING 状态时等待的最长时间 (以秒为单位)。

如果您将超时值设为 0，则函数不会等到服务启动后才开始，而是在发送启动请求后立即返回。

如果省略此参数或传递 NULL，则将超时设置为服务报告的默认超时。

请参阅 MSDN 帮助以获取 [SERVICE_STATUS_PROCESS](#) 结构的 “ dwWaitHint” 成员。

cServer (可选)

在其上运行服务的服务器名称。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

cDatabase (可选)

服务注册所在的数据库。

请参阅 MSDN 帮助中的 [OpenSCManager](#)。

返回值**参考**

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandle](#)

[ContinueService](#)

[ControlService](#)

[CreateService](#)

[OpenService](#)

[PauseService](#)

[StartService](#)

[StopService](#)

使用的 WinApi

[QueryServiceStatus](#)

[OpenSCManager](#)

[OpenService](#)

Shell

封装 Windows Shell 程序的一些功能。

SHCopyFiles	复制一个或几个文件。
SHDeleteFiles	删除一个或几个文件。
SHMoveFiles	移动一个或几个文件。
SHRenameFiles	重命名文件。
SHSpecialFolder	检索特殊文件夹的路径。

SHCopyFiles

复制一个或几个文件。

SHCopyFiles(cSource, cTarget, nFlags [, cTitle])

参数

cSource

一个或多个文件名。这些名称应为完全限定的路径，以防止出现意外结果。

仅在文件名位置允许使用标准 MS-DOS 通配符，例如 “*” 。在字符串的其他位置使用通配符将导致不可预测的结果。

每个文件名都应以单个 CHR (0) 终止。

cTarget

目标文件或目录名称。不允许使用通配符。它们的使用将导致不可预测的结果。

cTarget 必须满足以下规格：

不支持通配符。

它可以指定不存在的目标目录。在这些情况下，系统会尝试创建它们，并且通常会显示一个对话框，询问用户是否要创建新目录。若要取消显示此对话框并以静默方式创建目录，请在 nFlags 中设置 FOF_NOCONFIRMMKDIR 标志。

如果 nFlags 参数指定 FOF_MULTIDESTFILES，则它可以包含多个目标文件名。

使用标准路径。不禁止使用相对路径，但是会产生不可预测的结果。

nFlags (附加)

可以为以下任何一项：

Flag	描述
FOF_ALLOWUNDO	如果可能，保留撤消信息。 只能从执行原始操作的同一进程撤消操作。 如果 pFrom 不包含标准路径和文件名，则忽略此标志。
FOF_FILES ONLY	仅当指定了通配符文件名 (*.*) 时，才对文件执行操作。
FOF_MULTIDESTFILES	cTarget 参数指定多个目标文件（每个源文件一个），而不是要存放所有源文件的目录。
FOF_NOCONFIRMATION	对于显示的任何对话框，以“全部同意”答复。
FOF_NOCONFIRMMKDIR	如果操作需要创建一个新目录，请不要确认新目录的创建。
FOF_NO_CONNECTED_ELEMENTS	Shell 版本 5.0。不要将连接的文件成组移动。仅移动指定的文件。
FOF_NOCOPYSECURITYATTRIBS	Shell 版本 4.71。不要复制文件的安全属性。
FOF_NOERRORUI	如果发生错误，请不要显示用户界面。
FOF_NORECURSION	仅在本地目录中操作。不要对子目录进行递归操作。
FOF_RENAMEONCOLLISION	如果具有目标名称的文件已经存在，请在移动，复制或重命名操作中使用新名称给正在操作的文件。
FOF_SILENT	不显示进度对话框。
FOF_SIMPLEPROGRESS	显示进度对话框，但不显示文件名。如果传递 cTitle 参数，则会自动设置 FOF_SIMPLEPROGRESS。

cTitle (可选)

进度对话框的标题。

如果传递此参数，则会自动设置 FOF_SIMPLEPROGRESS 标志，因为如果不设置此标志就无法指定标题。

返回值

如果成功复制了所有文件，则为 1；如果中止操作，则为 0；否则，返回 [SHFileOperation](#) API 的错误代码。

参考

[SHCopyFiles](#)

[SHDeleteFiles](#)

[SHMoveFiles](#)

[SHRenameFiles](#)

[SHSpecialFolder](#)

SHDeleteFiles

删除一个或几个文件。

SHDeleteFiles(cFiles, nFlags [, cTitle])

参数

cSource

一个或多个文件名。这些名称应为完全限定的路径，以防止出现意外结果。

仅在文件名位置允许使用标准 MS-DOS 通配符，例如“*”。在字符串的其他位置使用通配符将导致不可预测的结果。

每个文件名都应以单个 CHR (0) 终止。

nFlags (附加)

可以为以下任何一项：

Flag	描述
FOF_ALLOWUNDO	如果可能，保留撤消信息。 只能从执行原始操作的同一进程撤消操作。 如果 pFrom 不包含标准路径和文件名，则忽略此标志。
FOF_FILES ONLY	仅当指定了通配符文件名 (*.*) 时，才对文件执行操作。
FOF_MULTIDESTFILES	cTarget 参数指定多个目标文件（每个源文件一个），而不是要存放所有源文件的目录。
FOF_NOCONFIRMATION	对于显示的任何对话框，以“全部同意”答复。
FOF_NOCONFIRMMKDIR	如果操作需要创建一个新目录，请不要确认新目录的创建。
FOF_NO_CONNECTED_ELEMENTS	Shell 版本 5.0。不要将连接的文件成组移动。仅移动指定的文件。
FOF_NOCOPYSECURITYATTRIBS	Shell 版本 4.71。不要复制文件的安全属性。
FOF_NOERRORUI	如果发生错误，请不要显示用户界面。
FOF_NORECURSION	仅在本地目录中操作。不要对子目录进行递归操作。
FOF_RENAMEONCOLLISION	如果具有目标名称的文件已经存在，请在移动，复制或重命名操作中使用新名称给正在操作的文件。
FOF_SILENT	不显示进度对话框。
FOF_SIMPLEPROGRESS	显示进度对话框，但不显示文件名。如果传递 cTitle 参数，则会自动设置 FOF_SIMPLEPROGRESS。

cTitle (可选)

进度对话框的标题。

如果传递此参数，则会自动设置 FOF_SIMPLEPROGRESS 标志，因为如果不设置此标志就无法指

定标题。

返回值

参考

[SHCopyFiles](#)

[SHDeleteFiles](#)

[SHMoveFiles](#)

[SHRenameFiles](#)

[SHSpecialFolder](#)

SHMoveFiles

移动一个或几个文件。

SHMoveFiles(cSource, cTarget, nFlags [, cTitle])

参数

cSource

一个或多个文件名。这些名称应为完全限定的路径，以防止出现意外结果。

仅在文件名位置允许使用标准 MS-DOS 通配符，例如 “*”。在字符串的其他位置使用通配符将导致不可预测的结果。

每个文件名都应以单个 CHR (0) 终止。

cTarget

目标文件或目录名称。不允许使用通配符。它们的使用将导致不可预测的结果。

cTarget 必须满足以下规格：

不支持通配符。

它可以指定不存在的目标目录。在这些情况下，系统会尝试创建它们，并且通常会显示一个对话框，询问用户是否要创建新目录。若要取消显示此对话框并以静默方式创建目录，请在 nFlags 中设置 FOF_NOCONFIRMMKDIR 标志。

如果 nFlags 参数指定 FOF_MULTIDESTFILES，则它可以包含多个目标文件名。

使用标准路径。不禁止使用相对路径，但是会产生不可预测的结果。

nFlags (附加)

可以为以下任何一项：

Flag	描述
FOF_ALLOWUNDO	如果可能，保留撤消信息。 只能从执行原始操作的同一进程撤消操作。 如果 pFrom 不包含标准路径和文件名，则忽略此标志。
FOF_FILES ONLY	仅当指定了通配符文件名 (*.*) 时，才对文件执行操作。
FOF_MULTIDESTFILES	cTarget 参数指定多个目标文件（每个源文件一个），而不是要存放所有源文件的目录。
FOF_NOCONFIRMATION	对于显示的任何对话框，以“全部同意”答复。
FOF_NOCONFIRMMKDIR	如果操作需要创建一个新目录，请不要确认新目录的创建。
FOF_NO_CONNECTED_ELEMENTS	Shell 版本 5.0。不要将连接的文件成组移动。仅移动指定的文件。
FOF_NOCOPYSECURITYATTRIBS	Shell 版本 4.71。不要复制文件的安全属性。
FOF_NOERRORUI	如果发生错误，请不要显示用户界面。
FOF_NORECURSION	仅在本地目录中操作。不要对子目录进行递归操作。
FOF_RENAMEONCOLLISION	如果具有目标名称的文件已经存在，请在移动、复制或重命名操作中使用新名称给正在操作的文件。
FOF_SILENT	不显示进度对话框。
FOF_SIMPLEPROGRESS	显示进度对话框，但不显示文件名。如果传递 cTitle 参数，则会自动设置 FOF_SIMPLEPROGRESS。

cTitle (可选)

进度对话框的标题。

如果传递此参数，则会自动设置 FOF_SIMPLEPROGRESS 标志，因为如果不设置此标志就无法指定标题。

返回值

如果成功复制了所有文件，则为 1；如果中止操作，则为 0；否则，返回 [SHFileOperation API](#) 的错误代码。

参考

[SHCopyFiles](#)

[SHDeleteFiles](#)

[SHMoveFiles](#)

[SHRenameFiles](#)

[SHSpecialFolder](#)

调用的 WinApi

[SHFileOperation](#)

SHRenameFiles

重命名文件。

SHRenameFiles(cSource, cTarget, nFlags [, cTitle])

参数

cSource

文件名。 该名称应为完全限定的路径，以防止出现意外结果。

cTarget

目标文件名。

cTitle (可选)

进度对话框的标题。

如果传递此参数，则会自动设置 FOF_SIMPLEPROGRESS 标志，因为如果不设置此标志就无法指定标题。

返回值

如果成功重命名了文件，则为 1；如果中止操作，则为 0；否则，返回 [SHFileOperation](#) API 的错误代码。

备注

您不能使用此函数重命名多个文件。

请改用 [SHMoveFiles](#)。

参考

[SHCopyFiles](#)

[SHDeleteFiles](#)

[SHMoveFiles](#)

[SHRenameFiles](#)

[SHSpecialFolder](#)

调用的 WinApi

SHFileOperation

SHSpecialFolder

检索特殊文件夹的路径。

SHSpecialFolder(nFolderId, @cFolder [, bCreateFolder])

参数

nFolderId (附加)

标识目标文件夹的 CSDL。

可以是以下值之一。

常量	描述
CSIDL_ADMINTOOLS	用于存储单个用户的管理工具的文件系统目录。 MMC 将自定义的控制台保存到此目录，并且将与用户漫游。
CSIDL_ALTSTARTUP	与用户的非本地化启动程序组相对应的文件系统目录。 Windows Vista 可以识别此值以实现向后兼容，但是文件夹本身不再存在。
CSIDL_APPDATA	文件系统目录，用作特定于应用程序的数据的公共存储库。典型的路径是 C:\Documents and Settings\用户名\Application Data。对于未安装 Microsoft Internet Explorer 4.0 集成命令行管理程序的系统，可再发行的 Shfolder.dll 支持此 CSDL。
CSIDL_BITBUCKET	包含用户回收站中对象的虚拟文件夹。
CSIDL_CDBURN_AREA	文件系统目录，用作等待文件写入 CD 的暂存区。典型路径为 C:\Documents and Settings\用户名\Local Settings\Application Data\Microsoft\CD Burning。
CSIDL_COMMON_ADMINTOOLS	包含计算机所有用户的管理工具的文件系统目录。
CSIDL_COMMON_ALTSTARTUP	与所有用户的非本地化启动程序组相对应的文件系统目录。 Windows Vista 可以识别此值以实现向后兼容，但是文件夹本身不再存在。
CSIDL_COMMON_APPDATA	包含所有用户的应用程序数据的文件系统目录。典型路径是 C:\Documents and Settings\All Users\Application Data。该文件夹用于非特定于用户的应用程序数据。例如，应用程序可以在 CSDL_COMMON_APPDATA 文件夹中存储拼写检查字

	典，剪贴画数据库或日志文件。此信息将不会漫游，并且使用计算机的任何人都可以使用。
CSIDL_COMMON_DESKTOPDIRECTORY	文件系统目录，其中包含为所有用户显示在桌面上的文件和文件夹。典型路径是 C:\Documents and Settings\All Users\Desktop。
CSIDL_COMMON_DOCUMENTS	包含所有用户通用文档的文件系统目录。典型的路径是 C:\Documents and Settings\All Users\Documents。
CSIDL_COMMON_FAVORITES	文件系统目录，用作所有用户共有的收藏夹项的通用存储库。
CSIDL_COMMON_MUSIC	用作所有用户通用音乐文件的存储库的文件系统目录。典型的路径是 C:\Documents and Settings\All Users\Documents\My Music。
CSIDL_COMMON_OEM_LINKS	Windows Vista 可以识别此值以实现向后兼容，但是不再使用该文件夹本身。
CSIDL_COMMON_PICTURES	用作所有用户通用图像文件的存储库的文件系统目录。典型的路径是 C:\Documents and Settings\All Users\Documents\My Pictures。
CSIDL_COMMON_PROGRAMS	文件系统目录，其中包含出现在“开始”菜单上的所有用户的通用程序组的目录。典型的路径是 C:\Documents and Settings\All Users\Start Menu\Programs。
CSIDL_COMMON_STARTMENU	文件系统目录，其中包含所有用户在“开始”菜单上显示的程序和文件夹。典型的路径是 C:\Documents and Settings\All Users\Start Menu。
CSIDL_COMMON_STARTUP	文件系统目录，其中包含出现在所有用户的“启动”文件夹中的程序。典型的路径是 C:\Documents and Settings\All Users\开始菜单\Programs\Startup。
CSIDL_COMMON_TEMPLATES	包含可供所有用户使用的模板的文件系统目录。典型的路径是 C:\Documents and Settings\All Users\Templates。
CSIDL_COMMON_VIDEO	文件系统目录，用作所有用户通用的视频文件的存储库。典型的路径是 C:\Documents and Settings\All Users\Documents\My Videos。
CSIDL_COMPUTERSNEARME	代表工作组中其他计算机的文件夹。
CSIDL_CONNECTIONS	表示网络连接的虚拟文件夹，其中包含网络和拨号连接。
CSIDL_CONTROLS	包含控制面板应用程序图标的虚拟文件夹。
CSIDL_COOKIES	用作 Internet cookie 通用存储库的文件系统目录。典型的路径是 C:\Documents and Settings\用户名\ Cookies。
CSIDL_DESKTOP	表示 Windows 桌面的虚拟文件夹，名称空间的根。

CSIDL_DESKTOPDIRECTORY	用于将文件对象物理存储在桌面上的文件系统目录（不要与桌面文件夹本身混淆）。典型路径是 C:\Documents and Settings\username\Desktop。
CSIDL_DRIVES	代表“我的电脑”的虚拟文件夹，其中包含本地计算机上的所有内容：存储设备，打印机和控制面板。该文件夹还可以包含映射的网络驱动器。
CSIDL_FAVORITES	文件系统目录，用作用户喜欢的项目的公共存储库。典型的路径是 C:\Documents and Settings\username\Favorites。
CSIDL_FONTS	包含字体的虚拟文件夹。典型路径是 C:\Windows\Fonts。
CSIDL_HISTORY	用作 Internet 历史记录项目的公共存储库的文件系统目录。
CSIDL_INTERNET	Internet Explorer 的虚拟文件夹。
CSIDL_INTERNET_CACHE	用作 Internet 临时文件的公共存储库的文件系统目录。典型的路径是 C:\Documents and Settings\用户名\Local Settings\Temporary Internet Files。
CSIDL_LOCAL_APPDATA	用作本地（非漫游）应用程序的数据存储库的文件系统目录。典型的路径是 C:\Documents and Settings\用户名\Local Settings\Application Data。
CSIDL_MYDOCUMENTS	代表“我的文档”桌面项目的虚拟文件夹。此值等效于 CSIDL_PERSONAL。
CSIDL_MYMUSIC	用作音乐文件的公共存储库的文件系统目录。典型的路径是 C:\Documents and Settings\User\My Documents\My Music。
CSIDL_MYPICTURES	用作映像文件的公共存储库的文件系统目录。典型的路径是 C:\Documents and Settings\用户名\My Documents\My Pictures。
CSIDL_MYVIDEO	用作视频文件的公共存储库的文件系统目录。典型的路径是 C:\Documents and Settings\用户名\My Documents\My Videos。
CSIDL_NETHOOD	一个文件系统目录，其中包含“网上邻居”虚拟文件夹中可能存在的链接对象。它与 CSIDL_NETWORK 不同，后者代表网络名称空间的根。典型的路径是 C:\Documents and Settings\username\NetHood。
CSIDL_NETWORK	一个虚拟文件夹，代表网络邻居，网络命名空间层次结构的根。
CSIDL_PERSONAL	代表“我的文档”桌面项目的虚拟文件夹。这等效于 CSIDL_MYDOCUMENTS。

	6.0 版之前的版本。 用于物理存储用户公用文档存储库的文件系统目录。 典型的路径是 C: \ Documents and Settings \ username \ My Documents。 这应该与名称空间中的虚拟“我的文档”文件夹区分开。 要访问该虚拟文件夹，请使用 SHGetFolderPath，它返回虚拟位置的 ITEMIDLIST，或参考“管理文件系统”中描述的技术。
CSIDL_PRINTERS	包含已安装打印机的虚拟文件夹。
CSIDL_PRINTHOOD	包含可以在“打印机”虚拟文件夹中存在的链接对象的文件系统目录。 典型路径是 C: \ Documents and Settings \ 用户名\ PrintHood。
CSIDL_PROFILE	用户的配置文件文件夹。 典型的路径是 C: \ Users \ username。 应用程序不应在此级别创建文件或文件夹；他们应将其数据放在 CSIDL_APPDATA 或 CSIDL_LOCAL_APPDATA 所指的位置下。 但是，如果要创建新的已知文件夹，则 CSIDL_PROFILE 引用的配置文件根目录是合适的。
CSIDL_PROGRAM_FILES	程序文件文件夹。 典型的路径是 C: \ Program Files。
CSIDL_PROGRAM_FILES_COMMON	跨应用程序共享的组件的文件夹。 典型路径是 C: \ Program Files \ Common。 仅对 Windows XP 有效。
CSIDL_PROGRAMS	包含用户程序组的文件系统目录（它们本身就是文件系统目录）。 典型路径是 C: \ Documents and Settings \ 用户名\ Start Menu \ Programs。
CSIDL_RECENT	文件系统目录，其中包含用户最近使用的文档的快捷方式。 典型路径是 C: \ Documents and Settings \ username \ My Latest Documents。 要在此文件夹中创建快捷方式，请使用 SHAddToRecentDocs。 除了创建快捷方式之外，此功能还可以更新命令行管理程序的近期文档列表，并将快捷方式添加到“开始”菜单的“我的近期文档”子菜单中。
CSIDL_RESOURCES	Windows Vista。 包含资源数据的文件系统目录。 典型路径是 C: \ Windows \ Resources。
CSIDL_SENDTO	包含“发送到”菜单项的文件系统目录。 典型路径是 C: \ Documents and Settings \ username \ SendTo。
CSIDL_STARTMENU	包含“开始”菜单项的文件系统目录。 典型路径是 C: \ Documents and Settings \ 用户名\开始菜单。
CSIDL_STARTUP	与用户的启动程序组相对应的文件系统目录。 只要有任何用户登录，系统就会启动这些程序。 典型路径是 C: \ Documents and Settings \ username \ Start Menu \ Programs \ Startup。
CSIDL_SYSTEM	Windows 系统文件夹。 典型路径是 C: \ Windows \ System32。

CSIDL_TEMPLATES	用作文档模板的公共存储库的文件系统目录。 典型路径是 C:\Documents and Settings\username\Templates。
CSIDL_WINDOWS	Windows 目录或 SYSROOT。 这对应于%windir%或%SYSTEMROOT%环境变量。 典型路径是 C:\Windows。

@cFolder

通过引用的变量，用于存储请求的文件夹路径。

bCreateFolder (可选)

默认值：.F.

指示是否要创建该文件夹（如果尚不存在）。

如果此值为.T.，则创建文件夹。

返回值

操作成功返回.T.；否则返回.F.。

参考

[SHCopyFiles](#)

[SHDeleteFiles](#)

[SHMoveFiles](#)

[SHRenameFiles](#)

[SHSpecialFolder](#)

调用的 WinApi

[SHGetSpecialFolderPath](#)

系统信息

检索系统信息

ADesktopArea	将可见桌面（不包括系统任务栏或任何应用程序桌面工具栏）区域的尺寸存储到数组中。
ADesktops	将与调用过程的指定窗口站关联的所有桌面存储到一个数组中。
ADisplayDevices	将有关当前会话中的显示设备的信息存储到数组中。
AResolutions	将有关显示设备的所有图形模式（分辨率）的信息存储到数组中。
AWindowStations	将当前会话中所有窗口站的名称存储到一个数组中。

ExpandEnvironmentStrings	在传入的字符串中扩展环境变量。
GetLocaleInfoEx	检索有关语言环境的信息。
GetSystemDirectory	检索 Windows 系统目录的路径。
GetWindowsDirectory	检索 Windows 目录的路径。
OsEx	检索当前操作系统。

ADesktopArea

将可见桌面（不包括系统任务栏或任何应用程序桌面工具栏）区域的尺寸存储到数组中。

ADesktopArea(cArrayName)

参数

cArrayName

返回时，该数组包含来自 [RECT](#) 结构的以下值。

元素	内容
1	左
2	右
3	上
4	下

返回值

.T.

参考

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStrings](#)

[GetLocaleInfoEx](#)

[GetSystemDirectory](#)

[GetWindowsDirectory](#)

[OsEx](#)

使用的 WinAPI

[SystemParametersInfo](#) (使用参数 SPI_GETWORKAREA)

ADesktops

将与调用过程的指定窗口站关联的所有桌面存储到一个数组中。

ADesktops(cArrayName [, nHWINSTA])

参数

cArrayName

列	内容
1	桌面名称

nHWINSTA (可选)

默认值:从 [GetProcessWindowStation](#) 返回的 windowstation。

HWINSTA api 句柄。

返回值

桌面数量.

参考

[ADesktopArea](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStrings](#)

[GetLocaleInfoEx](#)

[GetSystemDirectory](#)

[GetWindowsDirectory](#)

[OsEx](#)

使用的 WinAPI

[EnumDesktops](#)

[GetProcessWindowStation](#)

ADisplayDevices

将有关当前会话中的显示设备的信息存储到数组中。

ADisplayDevices(cArrayName [, cDeviceName])

参数

cArrayName

数组的名称，作为函数应将信息存储在其中的字符串。

返回时，该数组包含来自 DISPLAY_DEVICE 结构的以下值。

列	内容	数据类型
1	设备字符串	C
2	设备名称	C
3	设备 ID	C
4	设备 KEY	C
5	状态标记	N

cDeviceName (可选)

默认值: NULL

设备名称。如果为 NULL，则函数返回机器上显示适配器的信息。

有关更多信息，请参见备注。

返回值

显示设备数量。

备注

查看 EnumDisplayDevices API 的备注部分，了解如何为监视器等获取信息...

参考

[ADesktopArea](#)

[ADesktops](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStrings](#)[GetLocaleInfoEx](#)[GetSystemDirectory](#)[GetWindowsDirectory](#)[OsEx](#)

使用的 WinAPI

[EnumDisplayDevices](#)

AResolutions

将有关显示设备的所有图形模式（分辨率）的信息存储到数组中。

AResolutions(cArrayName [, cDeviceName])

参数

cArrayName

返回时，数组包含以下信息。

列	内容
1	指定分辨率的宽度(以像素为单位)
2	指定分辨率的高度(以像素为单位)
3	指定颜色分辨率,以每像素位数为单位.
4	指定频率,以赫兹(每秒周期)为单位.

cDeviceName (可选)

默认值:空

此参数为空或从 [ADisplayDevices](#) 返回的显示设备名称。

一个空值指定运行调用线程的计算机上的当前显示设备。

返回值

分辨率

参考

[ADesktopArea](#)[ADesktops](#)

[ADisplayDevices](#)[AWindowStations](#)[ExpandEnvironmentStrings](#)[GetLocaleInfoEx](#)[GetSystemDirectory](#)[GetWindowsDirectory](#)[OsEx](#)

使用的 WinAPI

[EnumDisplaySettings](#)

AWindowStations

将当前会话中所有窗口站的名称存储到一个数组中。

AWindowStations(cArrayName)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	Windowstation 的名称。	C

返回值

参考

[ADesktopArea](#)[ADesktops](#)[ADisplayDevices](#)[AResolutions](#)[ExpandEnvironmentStrings](#)[GetLocaleInfoEx](#)[GetSystemDirectory](#)[GetWindowsDirectory](#)

[OsEx](#)**使用的 WinAPi**[EnumWindowStations](#)

ExpandEnvironmentStrings

在传入的字符串中扩展环境变量。

ExpandEnvironmentStrings(cEnvironmentString)**参数****cEnvironmentString**

一个字符串，包含一个或多个格式为%variableName%的环境变量字符串。

对于每个此类引用，将%variableName%部分替换为该环境变量的当前值。

查找环境变量名称时，忽略大小写。如果未找到名称，则%variableName%部分将保持未展开状态。

返回值**参考**[ADesktopArea](#)[ADesktops](#)[ADisplayDevices](#)[AResolutions](#)[AWindowStations](#)[GetLocaleInfoEx](#)[GetSystemDirectory](#)[GetWindowsDirectory](#)[OsEx](#)**使用的 WinAPi**[ExpandEnvironmentStrings](#)

GetLocaleInfoEx

检索有关语言环境的信息。

GetLocaleInfoEx(nType [, nLocaleID])

参数

nType

nLocaleID (可选)

默认值: LOCALE_USER_DEFAULT

要检索其信息的语言环境标识符。 您可以使用 MAKELCID 宏来创建区域设置标识符，也可以使用以下预定义值之一。

LOCALE_CUSTOM_DEFAULT

LOCALE_CUSTOM_UI_DEFAULT

LOCALE_CUSTOM_UNSPECIFIED

LOCALE_INVARIANT

LOCALE_SYSTEM_DEFAULT

LOCALE_USER_DEFAULT

返回值

从 API [GetLocaleInfo](#) 获取的信息 (字符串)。

参考

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStrings](#)

[GetSystemDirectory](#)

[GetWindowsDirectory](#)

[OsEx](#)

使用的 WinAPI

[GetLocaleInfo](#)

GetSystemDirectory

检索 Windows 系统目录的路径。

GetSystemDirectory()

返回值

系统目录。

参考

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStrings](#)

[GetLocaleInfoEx](#)

[GetWindowsDirectory](#)

[OsEx](#)

使用的 WinAPI

[GetSystemDirectory](#)

GetWindowsDirectory

检索 Windows 目录的路径。

GetWindowsDirectory()

返回值

Windows 目录。

参考

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStrings](#)

[GetLocaleInfoEx](#)

[GetSystemDirectory](#)

[OsEx](#)

使用的 WinAPI

[GetWindowsDirectory](#)

OsEx

检索当前操作系统。

OsEx()

返回值

表示正在使用的操作系统的整数。

值	OS
1	Windows 95
2	Windows 95 OSR2
3	Windows 98
4	Windows 98 Second Edition
5	Windows Millennium
6	Windows NT 3.5
7	Windows NT 4.0
8	Windows NT 4.0 Server
9	Windows 2000
10	Windows XP
11	Windows XP Professional x64

12	Windows Home Server
13	Windows Vista
14	Windows Server 2003
15	Windows Server 2003 R2
16	Windows 7
17	Windows Server 2008
18	Windows Server 2008 R2
19	具有更高版本号的未知 Windows

备注

如果兼容模式生效，则 OsEx 函数会在标识自身时报告操作系统，该操作系统可能不是已安装的操作系统。例如，如果兼容模式生效，则 OsEx 将报告为应用程序兼容性选择的操作系统。

参考

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStrings](#)

[GetLocaleInfoEx](#)

[GetSystemDirectory](#)

[GetWindowsDirectory](#)

使用的 WinAPI

[GetVersionEx](#)

[GetNativeSystemInfo](#)

[GetSystemInfo](#)

磁盘卷信息

检索有关卷的信息。

[AVolumeInformation](#)

将有关与指定根目录关联的文件系统和卷的信息存储到阵列中。

AVolumeMountPoints	将指定卷上已安装文件夹的名称存储到阵列中。
AVolumePaths	将指定卷的驱动器号和卷 GUID 路径存储到阵列中。
AVolumes	将计算机上的卷名称存储到阵列中。

AVolumeInformation

将有关与指定根目录关联的文件系统和卷的信息存储到阵列中。

AVolumeInformation(cArrayName, cRootPath)

参数

cArrayName

返回时，数组包含以下信息。

元素	内容	数据类型
1	指定卷的名称.	C
2	卷序列号.	N
3	指定文件系统支持的文件名的最大字符长度.	N
4	与指定文件系统关联的标志.有关可能的值的列表,参看: GetVolumeInformation .	N
5	文件系统.例如 FAT 或 NTFS	C

cRootPath

要描述的卷的根目录。

返回值

.T.

参考

[AVolumeMountPoints](#)

[AVolumePaths](#)

[AVolumes](#)

使用的 WinApi

[GetVolumeInformation](#)

[SetErrorMode](#)

AVolumeMountPoints

将指定卷上已安装文件夹的名称存储到阵列中。

AVolumeMountPoints(cArrayName, cVolume)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	挂载文件夹的名称。	C

cVolume

要扫描已装入文件夹的卷 GUID 路径。可以使用 [AVolumes](#) 检索卷 GUID 路径的列表。

返回值

卷装载点的数量。

参考

[AVolumeInformation](#)

[AVolumePaths](#)

[AVolumes](#)

使用的 WinApi

[FindFirstVolumeMountPoint](#)

[FindNextVolumeMountPoint](#)

[FindVolumeMountPointClose](#)

AVolumePaths

将指定卷的驱动器号和卷 GUID 路径存储到阵列中。

AVolumePaths(cArrayName, cVolume)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	驱动器号或卷 GUID 路径。	C

cVolume

卷的 GUID 路径。 可以使用 [AVolumes](#) 获取卷 GUID 路径的列表。

返回值

卷路径的数量。

参考

[AVolumeInformation](#)

[AVolumeMountPoints](#)

[AVolumes](#)

使用的 WinApi

[GetVolumePathNamesForVolumeName](#)

AVolumes

将计算机上的卷名称存储到阵列中。

AVolumes(cArrayName)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	卷 GUID 路径。	C

返回值

卷的数量.

参考

[AVolumeInformation](#)

[AVolumeMountPoints](#)

[AVolumePaths](#)

使用的 WinApi

[FindFirstVolume](#)

[FindNextVolume](#)

[FindVolumeClose](#)

窗体信息

检索关于窗体的信息

AWindowProps	将有关窗口的属性列表中条目的信息存储到数组中。
AWindows	将窗口句柄 (HWND) 存储到数组中。
AWindowsEx	将有关窗口的信息存储到数组中。
CenterWindowEx	在特定窗口，父窗口或桌面上将窗口居中。
GetWindowRectEx	获取指定窗口的边界矩形的尺寸。
GetWindowTextEx	获取与窗口相对应的文本。

AWindowProps

将有关窗口的属性列表中条目的信息存储到数组中。

AWindowProps(cArrayName, nHwnd)

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	属性名	C
2	数据的句柄	N

nHwnd

需要获取属性的窗体的句柄。

返回值

窗体属性的数量。

参考

[AWindows](#)[AWindowsEx](#)[CenterWindowEx](#)[GetWindowRectEx](#)[GetWindowTextEx](#)

使用的 WinApi

[EnumPropsEx](#)

AWindows

将窗口句柄 (HWND) 存储到数组中。

AWindows(cArrayName, nType [, nParam])

参数

cArrayName

返回时，数组包含以下信息。

列	内容	数据类型
1	窗体句柄 (HWND) .	N

nType (附加)

以下值之一。

类型	描述
AWINDOWS_TOPLEVEL	返回顶层窗体。
AWINDOWS_CHILD	返回在 nParam 中传递的窗体的子窗体。
AWINDOWS_THREAD	返回 nParam 中传递的线程所拥有的窗体。
AWINDOWS_DESKTOP	返回 nParam 中传递的桌面中包含的窗体。
AWINDOWS_CALLBACK	将第一个参数解释为回调函数（上述内容的补充）。

nParam

根据 nType 参数，此参数应为以下值之一：

类型	参数含义
AWINDOWS_TOPLEVEL	不要传递此参数。
AWINDOWS_CHILD	为其获取子窗体的窗体句柄。
AWINDOWS_THREAD	为其获取拥有的窗口的线程句柄。

AWINDOWS_DESKTOP	为其获取包含的窗口的桌面句柄。
------------------	-----------------

返回值

如果 nType 包含 AWINDOWS_CALLBACK，则返回 1，否则返回窗口数。

示例

将顶级窗口枚举到数组中：

```
AWindows('laArray', 1)
```

通过为每个找到的窗口调用带有 hHwnd 参数的 SomeFunction 来枚举顶级窗口：

```
AWindows('SomeFunction', 1+16)
```

枚举_SCREEN 的子窗口：

```
AWindows('laArray', 2, _SCREEN.hWnd)
```

枚举线程的窗口：

```
AWindows('laArray', 4, _VFP.ThreadId)
```

参考

[AWindowProps](#)

[AWindowsEx](#)

[CenterWindowEx](#)

[GetWindowRectEx](#)

[GetWindowTextEx](#)

使用的 WinApi

[EnumWindows](#) (if nType = AWINDOWS_TOPLEVEL)

[EnumChildWindows](#) (if nType = AWINDOWS_CHILD)

[EnumThreadWindows](#) (if nType = AWINDOWS_THREAD)

[EnumDesktopWindows](#) (if nType = AWINDOWS_DESKTOP)

AWindowsEx

将有关窗口的信息存储到数组中。

AWindowsEx(cArrayName, cFlags, nType [, nParam])

参数

cFlags (附加)

字符的位置确定结果数组中的列位置

Char	Information
W	HWND
C	Class
T	Text
S	Style
E	ExStyle
H	HInstance
P	HWND of parent window
D	Userdata
I	ID
R	ThreadID
O	ProcessID
V	Visible
N	Iconic
M	Maximized
U	Unicode

nType (附加)

以下值之一。

类型	描述
AWINDOWS_TOPLEVEL	返回顶层窗体。
AWINDOWS_CHILD	返回在 nParam 中传递的窗体的子窗体。
AWINDOWS_THREAD	返回 nParam 中传递的线程所拥有的窗体。
AWINDOWS_DESKTOP	返回 nParam 中传递的桌面中包含的窗体。
AWINDOWS_CALLBACK	将第一个参数解释为回调函数（上述内容的补充）。

nParam

根据 nType 参数，此参数应为以下值之一：

类型	参数含义
AWINDOWS_TOPLEVEL	不要传递此参数。
AWINDOWS_CHILD	为其获取子窗体的窗体句柄。
AWINDOWS_THREAD	为其获取拥有的窗口的线程句柄。
AWINDOWS_DESKTOP	为其获取包含的窗口的桌面句柄。

返回值

窗体数量。

参考

[AWindowProps](#)

[AWindows](#)

[CenterWindowEx](#)

[GetWindowRectEx](#)

[GetWindowTextEx](#)

使用的 WinApi

[EnumWindows](#) (if nType = AWINDOWS_TOPLEVEL)

[EnumChildWindows](#) (if nType = AWINDOWS_CHILD)

[EnumThreadWindows](#) (if nType = AWINDOWS_THREAD)

[EnumDesktopWindows](#) (if nType = AWINDOWS_DESKTOP)

[GetClassName](#)

[GetWindowText](#)

[GetWindowLong](#)

[GetParent](#)

[GetWindowThreadProcessId](#)

[IsWindowVisible](#)

[IsIconic](#)

[IsZoomed](#)

[IsWindowUnicode](#)

CenterWindowEx

在特定窗口，父窗口或桌面上将窗口居中。

CenterWindowEx(nHwnd, nCenterInHwnd)

参数

nHwnd

应居中的窗口句柄。

nCenterInHwnd (可选)

窗口居中位置的窗口句柄。

如果省略 nCenterinHwnd，则该窗口在其父窗口中居中；如果没有父窗口，则该窗口在桌面中居中。

返回值

.T.

参考

[AWindowProps](#)

[AWindows](#)

[AWindowsEx](#)

[GetWindowRectEx](#)

[GetWindowTextEx](#)

使用的 WinApi

[SetWindowPos](#)

[GetWindowRect](#)

[GetParent](#)

[GetSystemMetrics](#) with parameter SM_CMONITORS

[SystemParametersInfo](#) with parameter SPI_GETWORKAREA

[MonitorFromWindow](#)

GetWindowRectEx

获取指定窗口的边界矩形的尺寸。

GetWindowRectEx(nHwnd, cArrayName)

参数

nHwnd

窗口的句柄。

cArrayName

返回时，该数组包含来自 [RECT](#) 结构的以下值。

元素	内容
1	左
2	右
3	上
4	下

返回值

.T.

参考

[AWindowProps](#)

[AWindows](#)

[AWindowsEx](#)

[CenterWindowEx](#)

[GetWindowTextEx](#)

使用的 WinApi

[GetWindowRect](#)

GetWindowTextEx

获取与窗口相对应的文本。

GetWindowTextEx(nHwnd)

参数

nHwnd

窗体句柄。

返回值

窗体的 Caption.

参考

[AWindowProps](#)

[AWindows](#)

[AWindowsEx](#)

[CenterWindowEx](#)

[GetWindowRectEx](#)

使用的 WinApi

[SendMessageTimeout](#)

附录

VFP2C32 Ver:2.0.0.16 新增函数

AMonitors

FindFileChangeEx

译者写在最后

- 本文档未标注“机翻”的大部分内容参考了机器翻译，甚至直接采用其结果。
- 因译者能力有限，如果您对此文档中的内容感到难以理解，请参阅原版英文帮助。
- [C 结构仿真](#)一节，请参阅英文帮助，本文档未对其进行翻译。
- 如果您有更好的翻译建议，请及时反馈，不胜感激！