September, 2001

Advisor Answers

Checking DLL functions

VFP 7/6/5/3

Q: Is there a way to find out if a DLL function you want to declare has already been declared?

–Dmitry Litvak (via Compuserve.COM)

A: In VFP 7, this one is really easy. The new ADLLS() function fills an array with a list of the currently declared functions. The function returns the number of rows in the resulting array. The array has three columns: the first contains the name of the function, the second contains the alias with which it was declared, and the third contains the name of the library (DLL) containing the function.

Once you fill the array with ADLLS(), you can use ASCAN() to see whether the specified function is included. ASCAN() has a new parameter in VFP 7 that lets you specify the column to search, so you can search only the appropriate column. In addition, another new ASCAN() parameter lets you indicate that you want an exact search, so that you can find only the exact function name or alias without worrying about the SET EXACT setting.

Here's a function to do the trick. You pass it the name or alias of the DLL function you're interested in. If you're passing the alias, pass .T. for the second parameter; otherwise, you can omit that parameter. The function returns .T. if the DLL is already declared and .F. if the function isn't declared. If there's a problem with the parameters, an error is fired.

```
* IsDLLDeclared.PRG
* This function determines whether a particular
* DLL function has been declared.
* Author: Tamar E. Granor
* Date: 22-May-2001

* Parameters:
*  cFunction - the name or alias of the function for
*              which to search
*  lSearchAlias - indicates whether to search for the
*                 alias or the name

LPARAMETERS cFunction, lSearchAlias

* Check parameters
```

```
ASSERT VARTYPE(cFunction) = "C" ;
  MESSAGE "IsDLLDeclared: First parameter " + ;
        "(cFunction) is required"

IF VARTYPE(cFunction) <> "C"
  ERROR 11
  RETURN .f.
ENDIF

ASSERT VARTYPE(lSearchAlias) = "L" ;
  MESSAGE "IsDLLDeclared: Second parameter " + ;
        "(lSearchAlias) must be logical"

IF VARTYPE(lSearchAlias) <> "L"
  ERROR 11
  RETURN .F.
ENDIF

LOCAL aDLLList[1], nDLLCount, lReturn, nSearchColumn

nDLLCount = ADLLS( aDLLList )
IF nDLLCount = 0
  lReturn = .F.
ELSE
  * Search for the right one
  IF lSearchAlias
    nSearchColumn = 2
  ELSE
    nSearchColumn = 1
  ENDIF

  * Since DLL functions are case-sensitive,
  * do a case-sensitive search.
  * In addition, do an exact search.
  IF ASCAN(aDLLList, cFunction, -1, -1, ;
        nSearchColumn, 6) > 0
    lReturn = .T.
  ELSE
    lReturn  = .F.
  ENDIF
ENDIF

RETURN lReturn
```

In VFP 6, this task is more difficult because there's nothing like the ADLLS() function. The only way to get this information is to parse the output of the LIST STATUS command. The function below actually creates the same array as ADLLS(), then searches it. However, searching in the array is also more difficult because we have to check that the result is in the right column and we have to make sure we're doing an exact search. Here's the code:

```
* IsDLLDeclared6.PRG
```

```
* This function determines whether a particular
* DLL function has been declared.
* This version is designed to work in the English
* version of VFP 6, only.
* Author: Tamar E. Granor
* Date: 24-May-2001

* Parameters:
*  cFunction - the name or alias of the function
*              for which to search
*  lSearchAlias - indicates whether to search for
*                 the alias or the name

LPARAMETERS cFunction, lSearchAlias

* Check parameters
ASSERT VARTYPE(cFunction) = "C" ;
  MESSAGE "IsDLLDeclared: First parameter " + ;
        "(cFunction) is required"

IF VARTYPE(cFunction) <> "C"
  ERROR 11
  RETURN .f.
ENDIF

ASSERT VARTYPE(lSearchAlias) = "L" ;
  MESSAGE "IsDLLDeclared: Second parameter " + ;
        "(lSearchAlias) must be logical"

IF VARTYPE(lSearchAlias) <> "L"
  ERROR 11
  RETURN .F.
ENDIF

ASSERT VERSION(3) = "00" ;
  MESSAGE "IsDLLDeclared: For English VFP only"

IF VERSION(3) <> "00"
  ERROR "Resource file mismatch"
  RETURN .F.
ENDIF

LOCAL cStatusFile, cStatusList, nDLLsStart
LOCAL aDLLOriginalList[1], aDLLList[1], nDLLCount
LOCAL lReturn, nSearchColumn, cOldExact, nStartPos
LOCAL nFoundPos, lDone, cLine, nComponents

* Get the status listing
cStatusFile = SYS(2015) + ".Txt"
LIST STATUS TO (cStatusFile) NOCONSOLE OVERWRITE

* Read the listing into a variable
cStatusList = FileToStr( cStatusFile )
ERASE (cStatusList)
```

```
* See whether there are any DLLs declared
nDLLsStart = AT("Declared DLLs:", cStatusList)
IF nDLLsStart = 0
  * No DLL functions declared
  RETURN .F.
ELSE
  * Get rid of unneeded info and parse DLL info
  cStatusList = SUBSTR( cStatusList, nDLLsStart )
  nDLLCount = ALines( aDLLOriginalList, cStatusList )
  * Get rid of the header line
  ADEL( aDLLOriginalList, 1)
  nDLLCount = nDLLCount - 1
  DIMENSION aDLLOriginalList[ nDLLCount  ]

  * Now break up the lines
  DIMENSION aDLLList[ nDLLCount , 3]
  aDLLList = ""
  FOR nItem = 1 TO nDLLCount
    * Convert items on the line to an array.
    * However, every space gets its own line
    nLines = ALINES( aLineItems, ;
                STRTRAN(aDLLOriginalList[ nItem ], ;
                " ", CHR(13)) )
    nColumn = 0

    * Loop through resulting array, and put
    * non-empty items into actual array.
    FOR nLine = 1 TO nLines
      IF NOT EMPTY(aLineItems[ nLine ])
        nColumn = nColumn + 1
        IF nLine = nLines
          * If we're on the last item, it goes in
          * column 3, no matter what
          nColumn = 3
        ENDIF
        aDLLList[ nItem, nColumn] = ;
          aLineItems[ nLine ]
      ENDIF
    ENDFOR
  ENDFOR

  * aDLLList now contains the name in column 1,
  * the alias in column 2,
  * and the library in column 3.

  * Search for the right one
  IF lSearchAlias
    nSearchColumn = 2
  ELSE
    nSearchColumn = 1
  ENDIF

  * Search until find result in right column or
  * reach end
  cOldExact = SET("EXACT")
```

```
  SET EXACT ON

  nStartPos = 1
  nFoundPos = ASCAN(aDLLList, cFunction, nStartPos)
  lDone = .F.
  DO WHILE nFoundPos > 0 AND NOT lDone

    IF ASUBSCRIPT(aDLLList,nFoundPos,2) = nSearchColumn
      lReturn = .T.
      lDone = .T.
    ELSE
      * Found in wrong column
      nStartPos = nFoundPos + 1
      nFoundPos = ASCAN(aDLLList, cFunction, nStartPos)
    ENDIF
  ENDDO

  IF NOT lDone
    lReturn  = .F.
  ENDIF

  SET EXACT &cOldExact
ENDIF

RETURN lReturn
```

There are several things you might want to do to improve this code. First, you may want to extract the portion of the second function that mimics ADLLS() and create your own version of that function for use in VFP 6 and earlier versions. For earlier versions you have to replace several functions, like ALINES(), as well. Second, you may want to combine the two functions into a single function that checks the VFP version and performs accordingly.

Finally, you should take into account that issuing a DECLARE command raises no error if the function has already been declared and typically is much faster than parsing a file. Neither ADLLS(), nor the replacement function, tell you with which parameters an API function has been declared, though.

The two functions (IsDLLDeclared.PRG and IsDLLDeclared6.PRG) are on this month's PRD.

–Tamar