Passing data from a program to a TKN and returning a result back to the calling program.

By Mike Bradbury, December 2005.

Environment Variables

The usual way of passing data to/from TKNs was to use a disk file link. I think you will agree, after trying it, that the method described here is simpler to manage and faster in operation. I have only been able to test with LB4.02/4.03b3 and Windows XP.

The MS SDK states that each running process has reserved space for what are known as Environment variables, which can be created during runtime and destroyed when the program is closed. So if a program comprises a number of TKN modules, those modules can pass data between themselves and the calling module, by means of CommandLine$ and Environment variables. Each Environment variable is unique to the process, even though the variable name may be the same.

Creation and reading of Environment variables is accomplished by means of Windows API, but is quite a simple process.

To create and set an Environment variable:

```
    calldll #kernel32,
"SetEnvironmentVariableA",lpName$ as ptr,lpValue$ as ptr,
result as long
```

where lpName$ is any name you choose to give to the variable and lpValue$ is the value you wish to assign to the variable.

To read an Environment variable:

It is necessary to reserve space in a buffer for the string returned from the variable. (It appears to be no longer necessary in LB4.02/4.03b3 to null terminate strings which are passed by pointer.)

```
    lpName$="YourEnvName"
```

```
    lpBuffer$=space$(maxEnvarLen)
    nSize=len(lpBuffer$)

    calldll #kernel32,
"GetEnvironmentVariableA",lpName$ as ptr,lpBuffer$ as ptr,nSize As
Long,_
                        result as Long
```

The examples below can be found in the zipped archive of this newsletter in files.zip, as source code which you should extract and copy to a temporary folder. They can then be compiled as TKNs in the same folder.

In each case Prog 2 and Prog 4 must be in TKN format, Prog 1 and Prog 3 can either be run from the LB IDE or as TKNs.

files.zip

- Details
- Download
- 45 KB

The contents of the zip are:

- Prog 1, callPassdataByEnvar1.bas
- Prog 2, passdataByEnvar1.bas
- Prog 3, callPassdataByEnvar2.bas
- Prog 4, passdataByEnvar2.bas

Programs 1 & 2:

The first program uses two Environment variables to pass two numeric values to a TKN and the same Environment variables to return a numeric result and a string. The maximum size for the Environment variables is set by maxEnvarLen to be 32 characters. The data is stored in the Environment variables and the second program (in TKN format) is run, whereupon the data is read from the Environment variables. The user can then enter a name in a textbox and when the Return button is clicked, the product of the two numbers and the entered name is stored in the Environment variables and control returns to the first program. The returned results are then read again from the Environment variables and displayed in the first window.

NOTE: in this demo the values of var1 and var2 need to be numerical only and validation should be

included in any practical application.

```
'Prog 1 start

nomainwin
global maxEnvarLen, MyEnvarName1$, MyEnvarName2$, var1, var2
maxEnvarLen=32
MyEnvarName1$="MyEnvarVariable1"
MyEnvarName2$="MyEnvarVariable2"
'next two values will be passed to the TKN and
'the product of them, returned to the calling prog
var1=123.456
var2=3.142

statictext #1.st1, ""                          ,10 ,10 ,450,40
statictext #1.st2, ""                          ,10 ,55 ,400,20
statictext #1.st3, "var1=";var1;", var2=";var2,10 ,95 ,200,20
statictext #1.st4, "(var1) x (var2) ="         ,10 ,130,90 ,20
statictext #1.st5, "?"                         ,105,130,400,20
statictext #1.st5a, "Name = ?"                 ,10 ,155,300,30
button #1.b1, "Run TKN",runTKN,ul          ,200,250,100,20
WindowWidth=500
open "Run TKN and capture returned data" for window_nf as #1
#1, "trapclose quit"
#1, "font arial 10"
#1.st1,
"Place two numbers into Environment Variables, run passdataByEnvar.tkn
 "+_

"and show product of those variables and a name, returned from TKN."
#1.st2,"Max string length for returned data (set by this program)= 32"
#1.st5a, "!hide"
wait

sub quit h$
    close #1
    END
end sub

sub runTKN h$
    call setEnvVar MyEnvarName1$,str$(var1)
    call setEnvVar MyEnvarName2$,str$(var2)
    #1.b1, "!disable"
    run "passdataByEnvar1.tkn "
    #1.b1, "!enable"
```

```
    #1.st5a, "!show"
    lpBuffer$=space$(maxEnvarLen+1)
    nSize=len(lpBuffer$)

    funcRet=getEnvVar(MyEnvarName1$,lpBuffer$,nSize)
    retValue=val(trim$(lpBuffer$))
    lpBuffer$=space$(maxEnvarLen+1)
    funcRet=getEnvVar(MyEnvarName2$,lpBuffer$,nSize)
    name$=trim$(lpBuffer$)
    #1.st5, retValue
    #1.st5a, "Name = ";name$

    'you may wish to clear the buffer and Environment variable here
    call setEnvVar MyEnvarName1$,""
    call setEnvVar MyEnvarName2$,""
end sub

sub setEnvVar envName$,envVal$
    calldll #kernel32, "SetEnvironmentVariableA", _
    envName$ as ptr, _
    envVal$ as ptr, _
    result as long
'If the function succeeds, the return value is nonzero.
end sub

function getEnvVar(lpName$,lpBuffer$,nSize)
    calldll #kernel32, "GetEnvironmentVariableA", _
    lpName$ As PTR, _
    lpBuffer$ As PTR, _
    nSize As Long, _
    getEnvVar as Long
'num of chars returned, or size of buffer required if buffer too small
.
end function

'Prog 1 end
```

```
'Prog 2 start

    nomainwin
    maxEnvarLen=32
    MyEnvarName1$="MyEnvarVariable1"
    MyEnvarName2$="MyEnvarVariable2"
```

```
    lpBuffer$=space$(maxEnvarLen+1)
    nSize=len(lpBuffer$)

'LB4.02 no need to pass lpBuffer$ byref as dll passes pointer referenc
e
    'later versions may change.
    funcRet=getEnvVar(MyEnvarName1$,lpBuffer$,nSize)
    var1=val(trim$(lpBuffer$))
    lpBuffer$=space$(maxEnvarLen+1)
    funcRet=getEnvVar(MyEnvarName2$,lpBuffer$,nSize)
    var2=val(trim$(lpBuffer$))
    WindowWidth = 555
    WindowHeight = 280
    statictext #1,
"Values received from calling program:",  60,  15,  300,  20
    statictext #1, "var1=";var1,  108,  45,  100,  20
    statictext #1, "var2=";var2,  108,  85,  100,  20
    statictext #1, "Enter your name...",108,120,200,20
    textbox #1.tb1, 108,145,200,25
    statictext #1,
"(Returned values = var1*var2 and Name)",  210,  205,  250,  20

    button #1.b1,"Return",[ok],ul,150,200,50,25
    stylebits #1, _DS_CENTER,0,0,0
    open
"Return data to calling program via ";MyEnvarName$ for window_nf as #1
    print #1, "font arial 10"
    print #1, "trapclose [quit]"
    #1.tb1, "!setfocus"
    wait

    [ok]
    #1.tb1, "!contents? name$"
    prod$=str$(var1*var2)
    if len(prod$)>maxEnvarLen then prod$=left$(prod$,maxEnvarLen)
    call setEnvVar MyEnvarName1$,prod$
    if len(name$)>maxEnvarLen then name$=left$(name$,maxEnvarLen)
    call setEnvVar MyEnvarName2$,name$
    [quit]
    close #1
    end


sub setEnvVar e$,d$
    calldll #kernel32, "SetEnvironmentVariableA", _
    e$ As ptr, _
```

```
    d$ As ptr, _
    result as long
end sub

function getEnvVar(lpName$,lpBuffer$,nSize)
    calldll #kernel32, "GetEnvironmentVariableA", _
    lpName$ As PTR, _
    lpBuffer$ As PTR, _
    nSize As Long, _
    result as Long
end function

'Prog 2 end
```

Programs 3 & 4:

The second pair of programs are very similar to the first pair, but I have attempted to indicate how a large amount of data could be passed to and returned from a TKN, using just one Environment variable, by assembling the data into a string and parsing the string to recover the data.

Althought tested up to 512 chars, I have not attempted to find the maximum string length which can be handled by kernal32.

```
'Prog 3 start

nomainwin
global maxEnvarLen, MyEnvarName$, var1, var2
maxEnvarLen=512
MyEnvarName$="MyEnvarVariable1"
'next two values will be passed to the TKN and
'the product of them, returned to the calling prog
var1=123.456
var2=3.142

statictext #1.st1, ""                          ,10 ,10 ,450,40
statictext #1.st2, ""                          ,10 ,55 ,400,20
statictext #1.st3, "var1=";var1;", var2=";var2,10 ,95 ,200,20
statictext #1.st4, "(var1) x (var2) ="         ,10 ,130,90 ,20
statictext #1.st5, "?"                         ,105,130,400,20
statictext #1.st5a, "Name = ?"                 ,10 ,155,450,70
statictext #1.st6, "Returned chars = 0"        ,10 ,230,400,20
button #1.b1, "Run TKN",getProduct,ul          ,200,260,100,20
```

```
WindowWidth=500
open "Run TKN and capture returned data" for window_nf as #1
#1, "trapclose quit"
#1, "font arial 10"
#1.st1,
"Place two numbers into an Environment Variable, run passdataByEnvar.t
kn "+_

"and show product of those numbers and a name, returned from TKN."
#1.st2,
"Max string length for returned data (set by this program)= 512"
#1.st5a, "!hide"
wait

sub quit h$
    close #1
    END
end sub

sub getProduct h$
    data$=var1;" ";var2
    call setEnvVar MyEnvarName$,data$

    #1.b1, "!disable"
    run "passdataByEnvar2.tkn "
    #1.b1, "!enable"
    #1.st5a, "!show"
    lpBuffer$=space$(maxEnvarLen+1)
    nSize=len(lpBuffer$)

    funcRet=getEnvVar(MyEnvarName$,lpBuffer$,nSize)
    retData$=trim$(lpBuffer$)
    retValue=val(word$(retData$,1,":"))
    #1.st5, retValue
    #1.st5a, "Name = ";word$(retData$,2,":")
    #1.st6, "Returned chars = ";funcRet

    'you may wish to clear the buffer and Environment variable here
    call setEnvVar MyEnvarName$,""
end sub

sub setEnvVar envName$,envVal$
    calldll #kernel32, "SetEnvironmentVariableA", _
    envName$ as ptr, _
    envVal$ as ptr, _
    result as long
```

```
'If the function succeeds, the return value is nonzero.
end sub

function getEnvVar(lpName$,lpBuffer$,nSize)
    calldll #kernel32, "GetEnvironmentVariableA", _
    lpName$ As PTR, _
    lpBuffer$ As PTR, _
    nSize As Long, _
    getEnvVar as Long
end function

'Prog 3 end
```

```
'Prog 4 start

    nomainwin
    maxEnvarLen=512
    MyEnvarName$="MyEnvarVariable1"
    lpName$=MyEnvarName$
    lpBuffer$=space$(maxEnvarLen+1)
    nSize=len(lpBuffer$)

    funcRet=getEnvVar(lpName$,lpBuffer$,nSize)

    data$=trim$(lpBuffer$)
    var1=val(word$(data$,1))
    var2=val(word$(data$,2))

    WindowWidth = 555
    WindowHeight = 280
    statictext #1,
"Values received from calling program:",  60,  15,  300,  20
    statictext #1, "var1=";var1,  108,  45,  100,  20
    statictext #1, "var2=";var2,  108,  85,  100,  20
    statictext #1, "Enter your name...",108,120,200,20
    textbox #1.tb1, 108,145,200,25
    statictext #1,
"(Returned values = var1*var2 and Name)",  210,  205,  250,  20

    button #1.b1,"Return",[ok],ul,150,200,50,25
    stylebits #1, _DS_CENTER,0,0,0
    open
"Return data to calling program via ";MyEnvarName$ for window_nf as #1
    #1, "font arial 10"
```

```
    #1, "trapclose [quit]"
    #1.tb1, "!setfocus"
    wait

    [ok]
    #1.tb1, "!contents? name$"
    data$=str$(var1*var2);":";name$

    if len(data$)>maxEnvarLen then
        data$=left$(data$,maxEnvarLen)
        notice "Data string truncated.";chr$(13);
"Exceeded number of characters permitted by calling program."
    end if
    call setEnvVar MyEnvarName$,data$

    [quit]
    close #1
    end


sub setEnvVar e$,d$
    calldll #kernel32, "SetEnvironmentVariableA", _
    e$ As ptr, _
    d$ As ptr, _
    result as long
end sub

function getEnvVar(lpName$,lpBuffer$,nSize)
    calldll #kernel32, "GetEnvironmentVariableA", _
    lpName$ As PTR, _
    lpBuffer$ As PTR, _
    nSize As Long, _
    result as Long
end function

'Prog 4 end
```

Command Line Variables

These demos are only intended to give a basic outline on the use of Environment variables in the application of data transfer. By combination of CommandLine$ and Environment variables, flexibility can be obtained in a number of ways. For example, the called TKN need not have the name and size of the Environment variables hard coded. The calling program can send that information as commandline

variables:

```
run "mytkn.tkn ";MyEnvarName$;" ";maxEnvarLen

or

envarData$=MyEnvarName$+":"+str$(maxEnvarLen)
run "mytkn.tkn ";envarData$
```

and parse the string within the TKN to obtain the Environment variable name and size. Any returned data would need to be tested for length and truncated to fit. It is even possible to include a flag in the returned string, to indicate that truncation had occurred and that a second attempt should be made made to call the TKN with an enlarged buffer size to receive the returned data.

Passing data from a program to a TKN and returning a result back to the calling program. | Environment Variables | To read an Environment variable: | Programs 1 & 2: | Programs 3 & 4: | Command Line Variables