This article originally appeared in the August 2001 issue of FoxTalk magazine.
Pinnacle Publishing.

What’s my (IP) Address?

Paul Mrozowski

Every so often the question: “How do I get the IP address of my system?” comes up. For me, up until now, the standard answer has always been to create the WinSock object, and use the LocalIP property. While this works, it has one glaring limitation – it only returns one IP address. In this article I’ll show you another method that doesn’t have this limitation – it will return all of the IP addresses your system is currently using and their associated subnet masks.

The heart of this code that makes this possible is the DECLARE in the Init() method of this class. I happened to come across this one in the October 2000 issue of Visual Basic Developer (VB Explorations: Your IP Addresses, Dan Appleman). I’m not much of a VB programmer so I immediately moved the DECLARE code over to VFP and started experimenting with it. My initial code wasn’t very pretty – it essentially just displayed the raw structure returned by GetIPAddrTable. It took me a while to work out the details of the structure and figure out how to convert the information into something useable. The numbers in the structure started looking familiar once I added the ConvertDWORDToNum() method. Plugging that routine into my shell code started giving me some recognizable results so I decided it was time to put everything together into a class.

The code itself is actually pretty simple and well documented, so I won’t go into detail here. One important thing to note though, the first IP address is number “0”. The second one is “1”, and so on. So if .nEntries contains a 2, you will need to ask for 0, and 1. The sample code demonstrates this.

Here’s some code showing how you might use this class:

```
SET CLASSLIB TO IpAddress
oIP = CreateObject(“IPAddress”)

lnEntries = oIP.nEntries - 1

FOR lnIndex = 0 TO lnEntries
  ? oIP.GetIPAddress(lnIndex)
  ? oIP.SubnetMask
ENDFOR
```

Simple, isn’t it?

I’ll be the first to admit, getting the IP address of your machine isn’t the type of thing that comes up everyday so this is one chunk of code you’ll want to file away until you need it.

Old technique for retrieving IP address

```
oIP = CreateObject(“MSWinSock.Winsock.1”)
?oIP.LocalIP
```

IP Address Code

```
*-- IPAddress : Allows developer to retrieve the
*--             IP Addresses and subnet masks on
*--             a system.
*-
*-- Copyright © 2001 – RCS Solutions, Inc.
*-- Written by: Paul Mrozowski
*-- 11/4/2000

DEFINE CLASS IPAddress AS CUSTOM

*-- # of entries in the IP Address table
nEntries = 0
IPAddress = (SPACE(0))
SubnetMask = (SPACE(0))
```

```
Name = "IPAddress"

PROCEDURE Init()

* This is the “magic” declare

DECLARE INTEGER GetIpAddrTable IN Iphlpapi ;
        STRING @ pIpAddrTable, ;
        INTEGER @ pdwSize, ;
        INTEGER BORDER

ENDPROC

*-- Retrieve IP address.
PROCEDURE GetIPAddress
LPARAMETER tnEntry
LOCAL pdwSize, pIpAddrTable, lnStructSize, ;
      lnOffset, lnIPNum, lnSubNum, lnEntries, ;
      lcAddr

lnOffset = 5
lnStructSize = 24

pdwSize = 0
pIpAddrTable = " "

IF VARTYPE(tnEntry) = "N"

   * Call the Win function to get IP addresses
   * We pass it the pIPAddrTable variable by
   * reference and it passes back the structure
   * size back into pdwSize.

   GetIpAddrTable(@pIpAddrTable, @pdwSize, 1)

   * Make sure we set aside enough room for
   * the info.

   pIpAddrTable = SPACE(pdwSize)

   * Call GetIPAddrTable again, this time we
   * can get the information that we’re
   * really interested in.

   GetIpAddrTable(@pIpAddrTable, @pdwSize, 1)

   lcAddr = SUBSTR(pIpAddrTable, 1, 4)

   lnEntries = THIS.ConvertDWORDtoNum(lcAddr)
   THIS.nEntries = lnEntries

   * If they asked for a specific entry, get
   * it as long as it exists

   IF tnEntry <= lnEntries – 1

      * We calculate the offset in the structure
      * to get the IP Address, which is 4 bytes
      * long (a DWORD). We also add in an offset
      * from the entry value.

      lnIPNum = SUBSTR(pIpAddrTable, ;
                (tnEntry * lnStructSize) ;
                + lnOffset, 4)

      THIS.IPAddress = THIS.ConvertToDot(lnIPNum)

      * Now get the subnet mask

      lnSubNum = SUBSTR(pIpAddrTable, ;
                 (tnEntry * lnStructSize) ;
                 + lnOffset + 8, 4)
      THIS.SubnetMask = THIS.ConvertToDot(lnSubNum)
    ENDIF
ENDIF

RETURN THIS.IPAddress
ENDPROC

*-- Convert the passed in # to IP address "dot" format
PROCEDURE ConvertToDot
LPARAMETER tcDWORD
LOCAL lcByte0, lcByte1, lcByte2, lcByte3, lcIP

lcIP = ""

IF VARTYPE(tcDWORD) = "C"

* Same code as ConvertDWORDToNum, we just don’t bother
* to convert it into a decimal since it’s practically
* in “dot” format already. Duplicated here for simplicity
* sake.

   lcByte0 = ASC(tcDWORD)
   lcByte1 = ASC(SUBSTR(tcDWORD, 2, 1))
   lcByte2 = ASC(SUBSTR(tcDWORD, 3, 1))
   lcByte3 = ASC(SUBSTR(tcDWORD, 4, 1))

   lcIP = ALLTRIM(STR(lcByte0)) + "." ;
        + ALLTRIM(STR(lcByte1)) + "." ;
        + ALLTRIM(STR(lcByte2)) + "." ;
        + ALLTRIM(STR(lcByte3))

ENDIF

RETURN lcIP
ENDPROC


PROCEDURE ConvertDWORDtoNum
LPARAMETER tcDWORD
LOCAL lcByte0, lcByte1, lcByte2, lcByte3, lnResult

IF VARTYPE(tcDWORD) = "C"
   lnResult = 0

   lcByte0 = ASC(tcDWORD)
   lcByte1 = ASC(SUBSTR(tcDWORD, 2, 1))
   lcByte2 = ASC(SUBSTR(tcDWORD, 3, 1))
   lcByte3 = ASC(SUBSTR(tcDWORD, 4, 1))

   lnResult = (((lcByte3 * 256 + lcByte2) * ;
              256 + lcByte1) * 256 + lcByte0)
ENDIF

RETURN lnResult
ENDPROC


PROCEDURE nEntries_access

THIS.GetIPAddress(0)
RETURN THIS.nEntries

ENDPROC


ENDDEFINE
```

Paul Mrozowski is a Senior Applications Developer for Kirtland Associates in Troy, Michigan.You can contact him at paulm@rcs-solutions.com.