When I ask the GetIpAddrTable function to sort the results, how are they sorted?

devblogs.microsoft.com/oldnewthing/20200415-00

April 15, 2020

Raymond Chen

A customer had a question about the way the `GetIpAddrTable` function sorts the results. The documentation says that if you pass `TRUE` as the `bOrder` parameter, then the mapping table will be sorted upon return.

What exactly is this sorting order?

The customer observed that in practice, they got the IP addresses in this order: Public IP addresses, then internal addresses, and then local addresses. They were interested in obtaining the public IP address, so they just asked for the results to be sorted and the grabbed the first one.

That worked great until one day, they grabbed the first sorted address and got the local address 127.0.0.1. Did this mean that the system didn't have any public IP addresses? The customer is trying to figure out why there was no public address, or at least no public address that the `GetIpAddrTable` function could find.

The problem is that their assumption wasn't supported by the documentation. The documentation says that the `bOrder` parameter controls "whether the returned mapping table should be sorted in ascending order by IPv4 address."

The sorting is done by IPv4 address, not by scope or availability or subnet or routing or broadcast. Specifically, the sorting is done in lexicographical order by the IPv4 address in network byte order.

The following table lists the IPv4 addresses in sorted order (not to scale):

Dotted notation	Network byte order	`dwAddress`	Notes
0.0.0.0	`00 00 00 00`	`0x00000000`	Local
0.0.0.1	`00 00 00 01`	`0x01000000`	

⋮	⋮	⋮	
0.255.255.254	`00 FF FF FE`	`0xFEFFFF00`	
0.255.255.255	`00 FF FF FF`	`0xFFFFFF00`	
1.0.0.0	`01 00 00 00`	`0x00000001`	Public
1.0.0.1	`01 00 00 01`	`0x01000001`	
⋮	⋮	⋮	
9.255.255.255	`09 FF FF FF`	`0xFFFFFF09`	
10.0.0.0	`0A 00 00 00`	`0x0000000A`	Private
10.0.0.1	`0A 00 00 01`	`0x0100000A`	
⋮	⋮	⋮	
10.255.255.255	`0A FF FF FF`	`0xFFFFFF0A`	
11.0.0.0	`0B 00 00 00`	`0x0000000B`	Public (mostly)
11.0.0.1	`0B 00 00 01`	`0x0100000B`	
⋮	⋮	⋮	
126.255.255.255	`7E FF FF FF`	`0xFFFFFF7E`	
127.0.0.0	`7F 00 00 00`	`0x0000007F`	Loopback
127.0.0.1	`7F 00 00 01`	`0x0100007F`	
⋮	⋮	⋮	
127.255.255.255	`7F FF FF FF`	`0xFFFFFF7F`	
128.0.0.0	`80 00 00 00`	`0x00000080`	Public (mostly)
128.0.0.1	`80 00 00 01`	`0x01000080`	
⋮	⋮	⋮	
255.255.255.255	`FF FF FF FF`	`0xFFFFFFFF`	

Note that the areas marked “Public (mostly)” contain islands of private or other special addresses within them. The purpose of this list was not to break down the entire IPv4 address range. It was to highlight that lexicographical ordering by IPv4 address in network byte order has no relation to the nature of the address.

I suspect what happened is that the company's public IP address assignment moved from an address less than `127.0.0.0` to one greater than `128.0.0.0`, which means that `127.0.0.1` is now the numerically lowest IP address.

The sorting performed by the `GetIpAddrTable` is purely numerical by IPv4 address. If you want to fish out your system's public IP address, you'll have to do your own filtering.

Bonus chatter: I listed IPv4 addresses like 0.0.0.1, even though 0.0.0.1 is strictly speaking not a valid IPv4 address. The `IP_MULTICAST_IF` socket option uses values of this form to mean "Not an address, but an interface index."

Raymond Chen

Follow