

# NetworkWorld

THE CONNECTED ENTERPRISE

# INSIDER

➔ Instant Expert

## IPV6 & WINDOWS

How to implement IPv6 in a Microsoft Windows environment



### INSIDE

- 2 **Getting serious with IPv6 in a Windows network environment**
- 5 **IPv6 Addressing, Subnets, Private Addresses**
- 7 **IPv6 Static Addressing, DNSv6**
- 11 **Setting up DHCPv6 to Dynamically Issue IPv6 Addresses**
- 14 **Configuring IPv6 Routing through IPv4 in Windows**
- 16 **Best Practices for Configuring Applications for IPv6**
- 17 **Configuring Active Directory to Support IPv6**

# Getting Serious with IPv6 in a Windows Network Environment

BY RAND MORIMOTO

**I**n February 2011, the Internet Assigned Numbers Authority (IANA) that issues IP addresses issued the last five blocks of public IP addresses, thus starting a firestorm of news stories that IPv6 is the next Y2K. As one of the twelve Y2K Advisors to the White House under President Clinton, I figured I'm compelled to write an article on IPv6, so here goes... (oh, and by the way, my stance on Y2K was that planes weren't going to fall out of the sky and the world wasn't going to come to an end at the stroke of midnight on January 1, 2000. The Daylight Savings Time issue in 2007 when the US changed the start and end dates of DST was a much more disruptive event for everyone in the computer industry...)

And by the way, unlike pretty much every other article on IPv6 on the Internet, I won't stop at just giving you the theoretical background of what IPv6 is. This is a first of several blog postings I'll be writing that'll actually give you hands-on, step-by-step guidance on how to actually implement IPv6 in your Windows/Active Directory environment.

## The IPv4 Problem

As a background for those who are just coming up to speed on the whole "IPv4 Problem", when IP addressing was developed for the Internet, it was expected that 232 (or 4.3-billion) IP addresses would be plenty, however by the late 1990s and the rise of the dotcom era, Internet service providers were forced to implement Network Address Translation (NAT) so that internet network users could use private (internal) IP addresses thus minimizing the need for every Internet connected device to have a public IP address. So today, when you plug your computer into the network at your place of work, or you plug your desktop into your home network, you go to a WiFi hotspot or connect and surf the Internet from your mobile phone, you are almost always assigned a private (internal) IP address. Your private

## Your guide to IPv6 in Windows networks

**B**y now, you've heard that ARIN has (more or less) run out of IPv4 addresses and the time has come for mass migration to IPv6. If you are a Windows shop, the good news is that Microsoft has been preparing for IPv6 for years and (almost) all of the latest Microsoft wares support it. Some, such as Windows Server 2008 R2, depend heavily on IPv6 for certain features.

But that doesn't mean firing up IPv6 in a Windows environment will be effortless. For now, most networks will need to build a dual-protocol network that will support both Internet Protocols, old and new, at least until the time when most of the world moves to support IPv6, likely a couple of years away.

No worries. Microsoft guru Rand Morimoto has come to the rescue with a comprehensive series of articles originally published on his Network World Microsoft Subnet blog, *Secrets of Windows Back Office Servers*. He covers everything on setting up an IPv6 network with Windows and other Microsoft products, from the basics and the details. This edition of Instant Expert has compiled his series into an easy-to-read PDF.

Morimoto has been in the computer industry for more than 30 years and has authored, co-authored, or been a contributing writer for a couple dozen books on Microsoft Windows, Security, Exchange email, BizTalk Server, and remote and mobile computing. Rand is the president of Convergent Computing, an IT consulting firm that has been one of the early adopter partners with Microsoft, implementing beta versions of Microsoft technologies two to three years before the product is released. This provides Morimoto with extensive knowledge on the technologies, even before they are on the market.

Besides speaking at more than 50 conferences and conventions worldwide on tips, tricks, and best practices on planning, migrating, and implementing technologies, Rand is also head judge for the worldwide Imagine Cup competition. All that is to say that Morimoto is well equipped to help you bring your Microsoft Network into alignment with the next version of the Internet.

*Follow Rand Morimoto and all the Microsoft Subnet bloggers on Twitter @microsoftsubnet*

address communications are routed out to the Internet through a router so that many private internally addressed users in a single site communicate through a single public IP address.

However, every publicly addressable Webserver needs to have a unique public IP address for Internet users to access the Webserver. Every NAT router also needs to have a public IP address for the internal users to access the Internet. With a world of 7-billion people and businesses around the world connecting their companies to the Internet

and hosting a Website, 4.3-billion public IP addresses isn't a lot.

## IPv6

To address this problem with IPv4 addresses being depleted, IPv6 was introduced to provide 2128 (or 340-undecillion (that's 340 with 36 zeros after it) IPv6 addresses. While the RFC on this is over a decade old, organizations have chosen to not spend time to implement IPv6 in their networks. The time has come to begin.

## Implementing IPv6 in a Windows Network

Fortunately IPv6 has been available in Windows since Windows 2008 and Windows Vista, so organizations that have migrated their servers and workstations to the latest Windows server and/or are on the path to implement Windows 7 on the client already have IPv6 available for their systems. So it is a matter of setting up the addressing scheme, IPv6 DNS, and routers to support IPv6.

Unfortunately an IPv4 device cannot route or access an IPv6 server system, so all endpoints need to support IPv6. However, once configured with IPv6, the endpoint can route over existing IPv4 networks, this is through IPv6 translation technologies called 6to4, Teredo, or IP-HTTPS. The 6to4 and Teredo are official IPv6 translation standards, IP-HTTPS is something Microsoft put together in Windows 2008 R2 and Windows 7 because most routers on the Internet don't route 6to4 or Teredo. So, while those are great standards, they don't work in the real world. With IP-HTTPS, an IPv6 system translates its IPv6 traffic over IPv4 through HTTPS, basically tunneling IPv6 through HTTPS like we've been doing RPC over HTTPS for Outlook to Exchange for years.

To implement (and really use) IPv6 in a Windows environment, an organization needs to have:

- Active Directory Servers on IPv6, so Active Directory 2008 or Active Directory 2008 R2, preferably Active Directory 2008 R2 so that global catalog servers support IP-HTTPS
  - DNS needs to support IPv6 (ie: DNS on a Windows 2008 R2 server) so that DNS will resolve IPv6 addresses, and again, on Windows 2008 R2 so that DNS supports IP-HTTPS
  - DHCP needs to be setup to support issuing IPv6 addresses (ie: DHCP setup on a Windows 2008 R2 server) just like the DHCP servers today issue IPv4 addresses
  - Client Systems should be upgraded to Windows 7 with an IPv6-enabled so that client systems are issued IPv6 addresses from DHCP servers, authenticate to Active Directory, and can do name resolution of IPv6 systems through the IPv6 DNS server(s).
- With this basic setup, the organization now has the basis for IPv6, however with just this, users still can't access servers in the network nor access the Internet.
- So the next step is to:
- Setup internal servers to support IPv6 (such as Exchange 2010, SharePoint 2010, SQL 2008 R2, and the like) on Windows 2008 R2 servers with IPv6 enabled. (Most companies that have been implemented Exchange 2010, SharePoint 2010, etc., have been implementing those systems

on Windows 2008 or Windows 2008 R2 which is great, those organizations are already one step closer to being on IPv6!)

- Setup Internet routing to support IPv6 so that IPv6 internal devices can communicate to and through the Internet

This last step is one of the more challenging steps as your Internet provider needs to support IPv6 as well as all of your Internet-working equipment (ie: switches, routers, gateways, etc) need to support IPv6 as well.

## Frequently Asked Questions (FAQs)

The following are frequently asked questions I get on IPv6:

### Is this IPv4 problem just hype (like Y2K) or is it real?

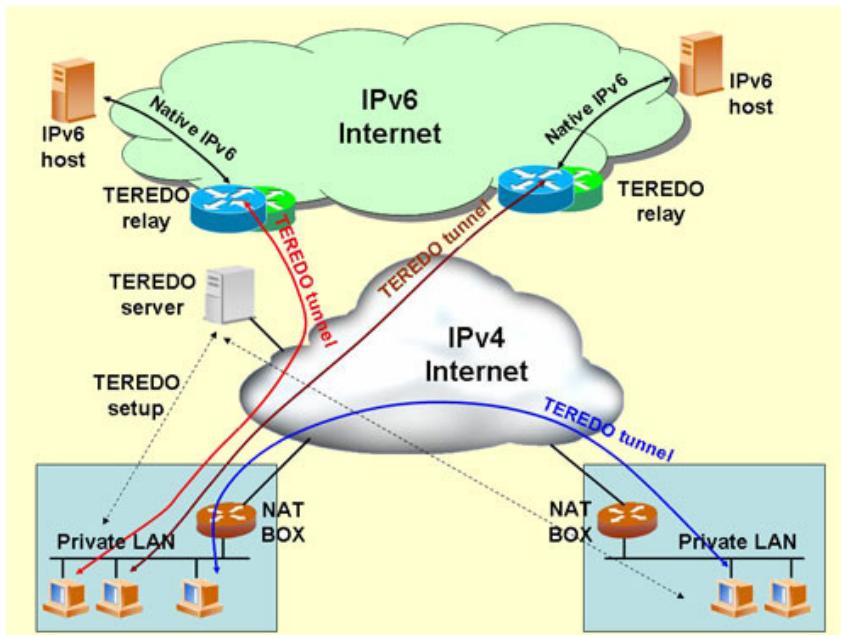
A: The IPv4 problem is real, and the Internet WILL run out of IPv4 addresses to issue likely before the end of 2011 in some regions such as Asia where Internet Service Providers cannot keep up with the number of public Internet connections being requested. However, no planes will fall from the sky, and if you already have all of your Web servers and public servers addressed and have a few spare public IP addresses available, you'll be fine, the Internet won't go away any time soon. However, new sites and new Internet hotspots may have a hard time coming online if there are no public IPv4 addresses available for them to bring up their sites

### So the problem is somebody else's problem and not mine, right?

A: Yeah, kind of, but Internet providers who make their money bringing on new customers won't be able to bring on new customers without public IP addresses available to host their customer's servers and connection points. When you make it an economic problem for Internet Service Providers, they will move faster to support IPv6 on their networks. Once they start the ball to provide IPv6 to endpoints (and not IPv4), then a whole wave of IPv6-only providers and devices will drive an accelerated need to have IPv6 support.

### So it's possible for hotspots to only support IPv6 then?

A: Absolutely, that's the issue. You'll have a corporate executive traveling to Asia with a laptop that plugs into a hotspot. The hotspot only supports IPv6, but the executive's laptop only supports IPv4 (either you haven't



upgraded the exec to Windows 7 yet, or you did but didn't setup IPv6 to work properly on the laptop). Now somebody else's problem becomes the problem of the person at the top of your company's food chain who can't access the Internet when traveling.

### So IPv6 is only an impact on convenience?

A: Actually, no, IPv6 can hit a company's bottom line too... Say for example your company sells stuff over the Internet, either directly as an e-tailer, or just has a global presence. If say Asia runs out of IPv4 addresses and then all new connections there get IPv6 addresses only for their mobile phones, laptops, hotspots, and businesses but your company has no IPv6 presence or support on the Internet, then your company's Website can't be accessed by all of those devices and now has an economic downside caused by your lack of getting IPv6 going on your network.

Your VP of Sales and the CEO will come rushing into your office wondering why millions of users in China can't access your company Website to buy your products, or Google doesn't resolve your IPv4 address for an IPv6-only connected searcher. Now not having IPv6 support is just like having your company Website down. Do people complain when your entire www public Website is down? If so, then you need to do something with IPv6 sooner than later.

### You mean Google has a different search engine for IPv6 sites than it has for IPv4 sites?

A: Google has a <http://ipv6.google.com> site that is specifically for IPv6 searches. It's unclear how Google will handle mixed IPv4 and IPv6 searches, but again, if an end point is only running IPv6 because they couldn't get an IPv4 address for their site, then that end point can only access other IPv6 servers. It is unlikely that Google will redirect an IPv6 user to hit an IPv4 site and have the user repeatedly get a "site not found" error.

### But I thought you said there was IPv6 to IPv4 translation?

A: Yes, there is 6to4, Teredo, and IP-HTTPS translation, but it is for ROUTING of IPv6 traffic through an IPv4 network, not a conversion algorithm that'll allow an IPv6 device to access an IPv4 device. There are ways to Proxy IPv6 traffic to access an IPv4 server, Microsoft has a product Unified Access Gateway (UAG) 2010 that does such a thing for DirectAccess IPv6 users to

access IPv4 servers. But this is intended to be a Proxy, meaning it is good for a business to allow employees access to selected servers, but not something that will handle day to day public website traffic.

### Isn't the whole idea of IPv6 to get rid of firewalls and NAT and have all devices just openly plug into the Internet, thus every one of our internal servers is exposed to the Internet with no security?

A: Well, kind of but not completely. The idea of every IPv6 being uniquely and directly addressable is true so that devices "can" communicate directly with one another over the Internet, but when you think about it, it's not like each of your servers will have a private connection to the Internet (ie: DSL line to the Internet for each server...). No, each of your IPv6 servers will connect to your company backbone just like it does today, and then your backbone will have to connect through some router that then connects to the Internet. And since you will be proxying or routing communications through a single Internet point (still), you can still put in firewalls and all that type of stuff like you have today.

What IPv6 does "is" provide the "ability" to have every device directly addressable on the Internet just like it was when the Internet was first founded and not have everything routed through a NAT server. There are pros and cons, but at least with IPv6 you have the "option" of having all devices available on the Internet without bottlenecks if you choose.

### Wait, so I need to put in a whole new network, firewall, and stuff and move all my devices to that network?

A: No, your existing Ethernet cabling and everything remains as is, but the devices that route your traffic, act as firewalls or gateways need to be able to support the routing of both IPv4 and IPv6. Remember when you created subnets on your routers to isolate traffic to specific subnets and then you setup routing between subnets? Well that was all for IPv4, right? If you put an IPv6 computer on that subnet, it won't use an IPv4 subnet routing protocol to get that IPv6 to "hop" to the next segment.

You need to make sure all of your Internetworking devices support routing of IPv6 traffic, and you need to make sure your firewall will allow IPv6 traffic through it. By default, an IPv4 firewall will block 100% of all IPv6 traffic. So you need a new firewall or

to upgrade your firewall, proxy server, ISA server, etc., so that they recognize and are configured to support your IPv6 scheme just like you've configured everything to support your IPv4 scheme.

### Can't I just wait until I really need IPv6 and deal with it then?

A: Uh, yeah you can, just like you can wait to start using a seatbelt in your car until you really need the seatbelt when you're in an accident ... IPv6 is not an "if" but more of a "when." It will happen, and unless you plan to retire in the next two years, it'll happen in your career span.

IPv6 is built in to Windows 2008 and higher, and Windows Vista/Windows 7. It doesn't take a lot to setup IPv6 DNS and IPv6 DHCP, and once those are setup, all of your current Windows (2008, 2008R2, Vista, and Windows 7) devices now support IPv6. Since it's not hard to do on the Windows side of things, you might as well put it in place so when you need to be on IPv6, you'll already be ready to do so.

### So just my Windows systems, right?

A: Windows systems are a start, but obviously you need to start the process to get all of your systems to IPv6 (Mac, Linux, Unix), and make sure that ALL of the routers, switches, appliances, gateways, anything you buy is IPv6 capable. If it is not IPv6 capable, you should NOT buy it. IPv6 will be here before your accounting department depreciates the hardware and software they are buying today, so make sure everything is IPv6 ready.

### Okay, I'm convinced, where do I start?

A: So 99% of the stuff you find on the Internet on IPv6 seems like the same article and information rehashed a million times telling you about IPv6, theoretically what it means, but there's not a lot of stuff out there that'll help an IT professional actually plan, architect, and implement an IPv6 scheme in their environment.

I've written several guides on actual IPv6 implementation (my book "Windows Server 2008 R2 Unleashed" by Sams Publishing has a whole chapter on IPv6 in a Windows environment), however I will be posting more on my blog here over the next few weeks on step by step guides. Come back to this site <http://www.networkworld.com/community/morimoto> and I'll have more hands-on resources available.

# IPv6 Addressing, Subnets, Private Addresses

Understanding the Basic Addressing of IPv6 in your Windows IPv6 Architectures

BY RAND MORIMOTO

**T**his is the first of many technical blog posts I'm going to post on IPv6 architecture and implementation for a Microsoft Windows-based environment. I started off with a basic introduction of the IPv4 problem and covered a handful of FAQs in my initial post; now on to the "How to" guides.

In this blog post, I'm going to cover:

- Explanation of IPv6 in terms and terminology for those of you familiar with IPv4
- How you officially get a block of IPv6 address
- What the equivalent of private (internal) network addressing is in IPv6
- Understand IPv6 addressing
- How to subnet IPv6
- How the concept of gateways and routing works in IPv6

## Explanation of IPv6 in terms and terminology for those of you familiar with IPv4

IPv6 is similar in many ways to IPv4 addressing, basically every device has to have an IP address, there is name resolution of IPv6 addresses to common names, dynamic addressing, static addressing, routing, etc. However when drawing up the specification for IPv6, rather than doing things "exactly" like IPv4 (good and bad), IPv6 improved upon a handful of things (that I'll explain here) to simplify addressing, routing, improve security, and improve performance and efficiency of IPv6 communications compared to IPv4. So as much as IPv6 addresses are really long and you might assume that would put a huge increase in traffic and payload on IPv6 over IPv4, what was done "inside" IPv6 actually makes it more efficient in many ways.

So the following are terminology in IPv4 terms and how they are addressed in IPv6:

- **IP Address:** Each device will have an IP address still, but instead of an

IPv4 address, they will have an IPv6 address. Other than the length and slightly different look, this concept is identical.

- **Subnet Mask:** We used to do subnet masks in IPv4 with notation like 255.255.255.0, but in IPv6, while we still do subnetting, the notation is different in two ways. We now write subnets using a slash and a number that denotes the masking. So it'll look like IPV6ADDRESS/64 or IPV6ADDRESS/112. But when you actually key in the IPv6 address on a system, that /64 or /112 will convert to a hexadecimal number that will be in the middle of the IPv6 Address. So when you see an IPv6 address, while it is really long, it actually includes the Network Address: Subnet: Device IP Address in that long address string. More on this in the "Understand IPv6 Addressing" section below.

- **Gateway Address:** The concept of the network gateway in IPv6 is the same as in IPv4, a gateway address will be designated noting how traffic can be routed out of the current subnet (technically the IPv6 Gateway address is not a formalized standard in IPv6, however Microsoft has included a Gateway setting in their IP Configuration properties page).

So all of the concepts remain the same, but you'll see when we get to the IPv6 addressing section that the long IPv6 address includes the Network Address, Subnet, and Unique Device Address all together.

## How you officially get a block of IPv6 addresses

So the next question everyone always asks is "How do I get an official block of IPv6 addresses?" That's kind of simple, "How'd you get your official public IPv4 addresses that you have today?" Usually the answer is that you got them from your Internet Service Provider (ISP) such as ATT, Sprint, Comcast, or the like when you had your Internet connection line pulled into your building. That same concept still applies as the big Internet knows generally where to find you by knowing what region you are in, and what ISP

you are connected to by the general range of addresses you are using.

Of course some of you were the lucky ones that actually got a block of IP Addresses early on when IP addresses were being given away just by writing and asking for a block of addresses. For those of you spoiled by owning your own IPv4 block, you're now at the mercy of your ISP to "loan you" a block of their addresses, you no longer own IP addresses (for IPv6) even if you owned IPv4 addresses before.

However, you CAN get ARIN to issue you your own block of IPv6 addresses if you are an ISP or will be acting as an ISP, see this link for a document on how to request IP Addresses from ARIN directly.

## What the equivalent of private (internal) network addressing is in IPv6

So the next question that I'm typically asked is "How about private (internal) network addresses? Do they exist in IPv6?" And the answer is yes. If you are just fiddling around with IPv6 in your lab, or you want to do the equivalent of network address translation where you have private addresses for your internal servers and systems, then you can use IPv6 private addressing, or what is called Unique Local Addresses (ULA). In the IPv4 world, private addresses include 10.0.0.0-10.255.255.255, and 172.16.0.0-172.31.255.255, and 192.168.0.0-192.168.255.255. In the IPv6 world, the ULA space is fc00::/7, or basically anything that starts with fd in the IPv6 address, so fdxx:xxxx:xxxx...

Do note though, if you use Unique Local Addressing in IPv6, these addresses cannot be routed on the Internet. These devices will always have to remain behind a router. The good part about that is that you control these devices like you do IPv4 devices on the "inside" of your network. So some may say using an ULA is more secure because the device cannot be accessed externally. However, if everything is on the inside of a firewall, no one can access the device anyway. And

because there are so many IPv6 addresses, it's not like someone will "guess" the address of the destination devices either.

Another argument against Unique Local Addressing is the whole concept of IPv6 is to be able to have IPv6 devices globally routable so that in the future, you want to have your IPv6 systems talk to other IPv6 systems directly without having to translate addresses through a router (from private to public addressing), having publicly accessible IPv6 addresses on internal devices is planning for the future of what will come in IPv6 communications.

This is a tough one. We got convinced by ISPs to setup Network Address Translation (NAT) and hide everything behind a firewall with non-routable private addresses and we think we have security. But if we simply use routable IPv6 addresses and create secured subnets protected by routers and firewalls, we're effectively getting the same security without having to have the overhead of address translation. I highly encourage organizations to consider implementing publicly addressable IPv6 addresses for all devices.

But if you are fiddling with IPv6 in your lab, rather than using a legally assignable public address (illegally), you might as well use the ULAs and do private internal addressing. When you are fiddling with ULAs, you can pick anything that starts with fd and pick anything you want after that, or if you want a truly randomly generated ULA, there's this Website that will generate a unique group of ULAs for you. This site will grab 1 of the 72-trillion possible Network/Subnet addresses for you to work with in your lab, that'll then give you your open realm to use any of the 18-quintillion (18 with 18 zeros) devices connected to your network/subnet. Again, while ULAs are not routable, if you were to put a ULA addressed device on the Internet, it would likely be unique.

## Understand IPv6 addressing

Okay, so I've gone through the concepts of IP Addressing in IPv6, in which I explained that the same concepts we've used and have

gotten familiar with in IPv4 still remain in IPv6, but with slightly different notation. Now to take a step back and actually provide you details on how IPv6 addressing is done.

In IPv4, as a 32-bit address, we separated the 32-bits into 4 octets separated by periods (or dots), so it looks like 10.12.2.200. We'd give the address a Subnet mask like 255.255.0.0 which means the network is 10.12 and the device address is 2.200.

In IPv6, as a 128-bit address, rather than breaking into dot-separated octets that would end up being 16 numbers (separated by periods), IPv6 uses long hexadecimals in a double-octet format separated by a colon. Effectively it is written out as 8 sets of "numbers" (since this is hexadecimal, it is 0 thru 9, and a thru f)

so something like fd30:0000:0000:0001:ff4e:003e:0009:000e

IPv6 addressing allows you to drop preceding zeros in the format, so the above could be simplified as: fd30:0000:0000:1:ff4e:3e:9:e

And when you have a double-octet group that is nothing but zeros (0000) you can replace the entire grouping with a ::, so this further simplifies the above to look like fd30::1:ff4e:3e:9:e (Note: You can only have 1 set of :: in an IPv6 address, so if you have two groupings of zeros, you would put the :: on one set but not on the second set to truncate.)

## How to subnet IPv6

So I told you I'd explain how subnetting works in IPv6. For the above address, it's not just one massively long IP address. It's actually broken down into 3 parts, the network address, the subnet address, and the device address.

The network address is the first 48-bits of the address, or since they are grouped in 16-bit groupings, effectively the first 3 groups of numbers designate the network. For the above example, the network address is fd30:0000:0000. For those getting their IPv6 addresses from an ISP, the first part of this Network Address will be the same for all of the customers of the ISP, which will designate the region and ISP. If you are doing Unique Local Addressing (ie: IPv6 private addressing), you could effectively

just address it as fd00:0000:0000 where fd designates this as a ULA, and that you are working with a single common network.

The subnet address is the next 16-bits of the address, or as addresses are grouped in 16-bit groupings, the next group in the IPv6 string. For the above example, that would be 0001. Instead of a subnet address, in IPv6, you just note the network address and the subnet address, and that'll give this address a specific designation of the network that this device is on, and the subnet that this device is on. This is where I noted IPv6 is more efficient than IPv4 as each packet has everything a router needs to route the information along, instead of having to add or append routing information, or look to a completely separate subnet mask parameter to work backwards into the address.

The last 64-bits (or four groupings) is the unique device address, in this case, the device is specifically ff4e:003e:0009:000e.

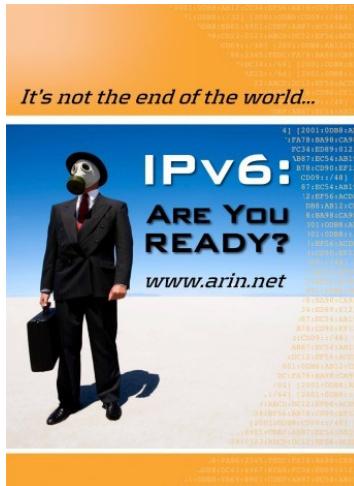
With 16-bit allocated to subnets, and 64-bits allocated to devices on a subnet, that means a single network address can have 65,535 subnets each with over 18-quintillion (18 with 18 zeros after it) devices. And with 48-bits allocated to the network address header of the IPv6 address, that's 281-trillion networks (with 65,535 subnets, with 18-quintillion devices).

## How the concept of gateways and routing works in IPv6

So for a gateway address in IPv6, it works exactly like IPv4, you'll have some IPv6 address that'll be the route out of your subnet. Just like in IPv4, that gateway address needs to be an IP address ON the subnet you are on so that your traffic hits that gateway address, and presumably that gateway address will then be configured to route your communications to a device outside of your subnet.

Routing works the same too, there will be IPv6 routes that will be dynamically configured, or you will statically configure routes between subnets.

Hopefully now the super long IPv6 address makes a little sense. From your ISP, you will likely be given the first four groupings of numbers (the network and subnet) and you will have the last four groupings to address as you please. You will define your gateway address that will take you out of your subnet to other subnets. In the case of a simple router between your subnet and the public Internet, that gateway address will route from your subnet out through your router to your ISP and on to the public Internet.



# IPv6 Static Addressing and DNSv6

Configuring IPv6 addresses on Windows 2008 and 2008 R2 Servers and Windows 7 Workstations, and configuring DNSv6

BY RAND MORIMOTO

**M**y last blog post covered basic terminology on IPv6 addressing, including concepts around subnets, gateways, private (internal) addresses, and how to read a long IPv6 address and understand the address. Now it's time to roll up your sleeves, take what you've learned, and start configuring Windows servers and workstations with IPv6.

In this blog post on IPv6, I'm going to cover:

- How to statically address a Windows 2008 / Windows 2008 R2 Server
- How to statically address a Windows 7 client
- How to setup DNS for IPv6 on a Windows 2008 R2 Server to do name resolution of IPv6 systems

I'm going to assume you have your network/subnet architecture figured out; if anything just make your network 1 and subnet 1 so (fd00:0000:0001:0001) but you can get more creative than that since you have a quintillion numbers to choose from. And I'm assuming you have some idea of how you will assign device addresses (again out of the 18-quintillion – 18 with 18 zeros – possible addresses) for addressing your systems. You could always start your device addressing with 0000:0000:0000:0001, and 0000:0000:0000:0002, and so forth, or you can use an addressing scheme I started using when I was playing with this stuff in the lab a while back that helped me know what type of device I was working with. So I put a 1, 2, or 3 in the first address place to designate a router/networking device, computer, or printer. I put a 1 or a 2 in the second address place to designate a server or workstation. I put a 1, 2, or 3 in the third address place to designate Windows, Mac, or Linux. I put a 1, 2, or 3 in the fourth address place if the device is a desktop, laptop, or tablet, etc. So if I have a device address 2212:0000:0000:0001 I know that this is a computer, workstation, Windows, laptop. There are variations that include domain attached or not, intended for public access or not, physical or virtual server, even the type of application the server primarily runs like Exchange, SharePoint, Global Catalog, etc. This actually helped me do my inventory of devices through the IP addresses when I was coming up to speed on IPv6.

For this example, here are the following assumptions:

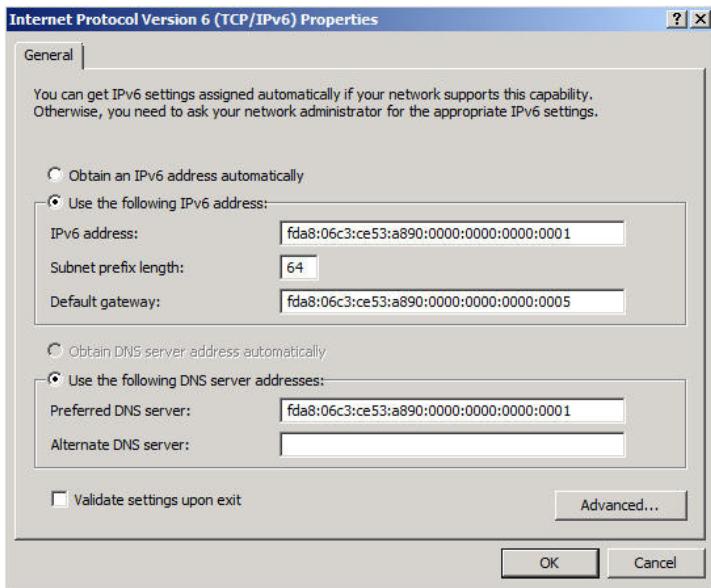
- I'm going to be using Unique Local Addresses (ie: private IPv6 addressing), thus my prefix starts with fd.
- I'm going to use a randomly selected GlobalID (a8:06c3:ce53) and SubnetID (a890) so that all of my devices will start with fda8:06c3:ce53:a890. Again, the only thing that needs to be there is starting off with fd to make this an officially private – ULA – address, after that, the rest of the “numbers” – 0-9, a-f – can be anything you want them to be.
  - For simplicity, I'm going to number my devices sequentially, so that the server will be fda8:06c3:ce53:a890:0000:0000:0000:0001 and the workstation will be fda8:06c3:ce53:a890:0000:0000:0000:0002 (remember, I can use the truncation method and get rid of extra zeros so it'll look like fda8:6c3:ce53:a890::1 and fda8:6c3:ce53:a890::2, but I'll write out the full address so as to not confuse everyone.
  - For the gateway to another subnet, I'm going to make that simply: fda8:06c3:ce53:a890:0000:0000:0000:0005.
  - For the DNS address, in my case, the server we're configuring is the DNS server, so the DNS address is fda8:06c3:ce53:a890:0000:0000:0000:0001.

## How to statically address a Windows 2008 / Windows 2008 R2 Server

With the static IP address fda8:06c3:ce53:a890:0000:0000:0000:0001 selected for the server and the gateway fda8:06c3:ce53:a890:0000:0000:0000:0005, to statically address a Windows 2008 R2 server, I will do the following:

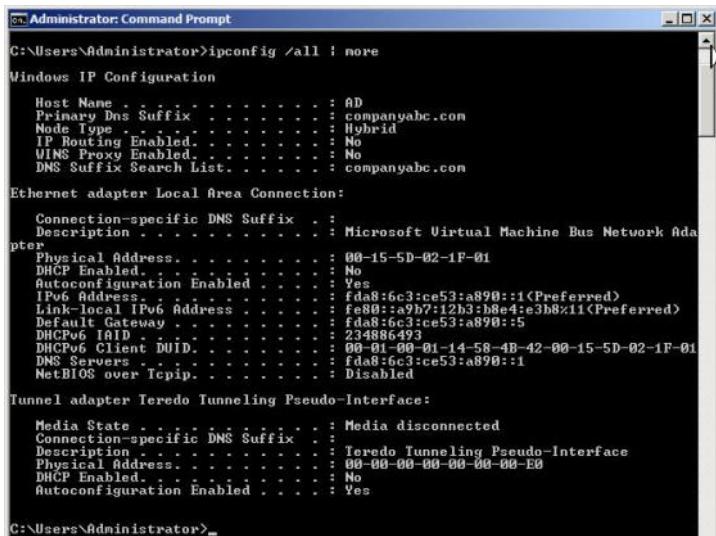
- 1) Logon to the server with administrator rights
- 2) Click on Start | Network | Network and Sharing Center | Change Adapter Setting
- 3) Right-click on the Local Area Connection of the network adapter I want to set IPv6 and choose Properties.
- 4) Highlight Internet Protocol Version 6 (TCP/IPv6) and click on Properties.
- 5) Click on “Use the following IPv6 address” and in the IPv6 Address field, type in the IP address you want to use (in my case, I typed in fda8:06c3:ce53:a890:0000:0000:0000:0001)
- 6) Press the Tab key and it will automatically populate the Subnet prefix length to 64.
- 7) Press the Tab key and in the Default Gateway field, type in the IP address you want to use for your gateway (in my case, I typed in fda8:06c3:ce53:a890:0000:0000:0000:0005).
- 8) For Preferred DNS Server, type in the IP address of your DNS server (which since I'm sitting on my DNS server, I'm going to type in the fda8:06c3:ce53:a890:0000:0000:0000:0001)

The screen will look something like the following:



9) Click OK, OK, Close to save and exit

Running IPConfig /All to see what the configuration looks like, you'll notice the IPv6 Address, Default Gateway address, and DNS server address have all been truncated down to the simplified format. In my case, I disabled IPv4 so there is no IPv4 address reporting on this system.

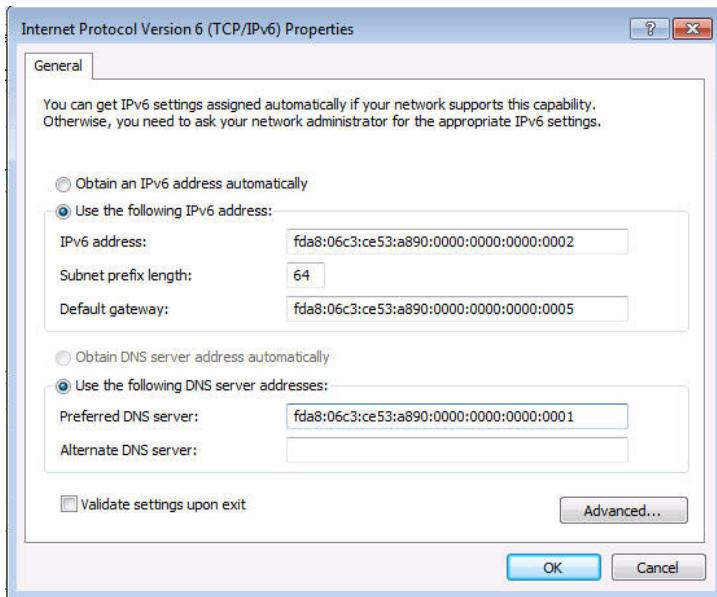


### How to statically address a Windows 7 Client System

For a Windows 7 client, the configuration process works exactly the same. With the static IP address fda8:06c3:ce53:a890:0000:0000:0000:0002 selected for the workstation and the gateway fda8:06c3:ce53:a890:0000:0000:0000:0005, to statically address a Windows 7 workstation, I will do the following:

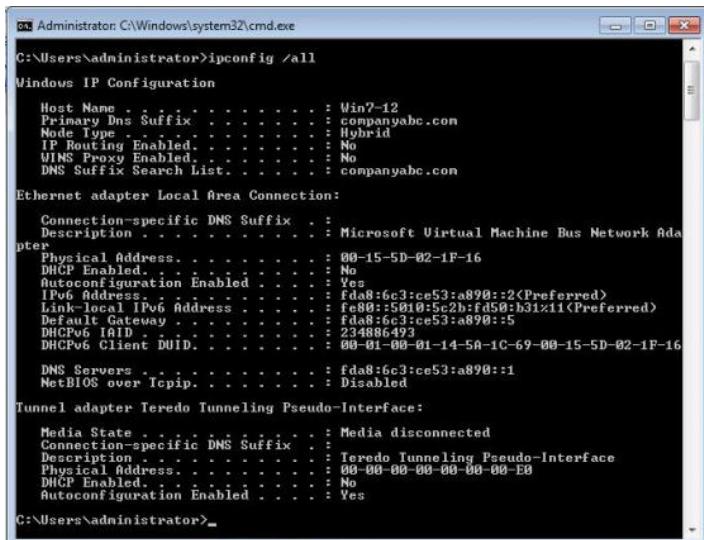
- 1) Logon to the workstation with administrator rights.
- 2) Click on Start | Control Panel | Network and Internet | Network and Sharing Center | Change Adapter Setting.
- 3) Right-click on the Local Area Connection of the network adapter I want to set IPv6 and choose Properties.

- 4) Highlight Internet Protocol Version 6 (TCP/IP6) and click on Properties.
  - 5) Click on "Use the following IPv6 address" and in the IPv6 Address field, type in the IP address you want to use (in my case, I typed in fda8:06c3:ce53:a890:0000:0000:0000:0002).
  - 6) Press the Tab key and it'll automatically populate the Subnet prefix length to 64.
  - 7) Press the Tab key and in the Default Gateway field, type in the IP address you want to use for your Gateway (in my case, I typed in fda8:06c3:ce53:a890:0000:0000:0000:0005).
  - 8) For Preferred DNS Server, type in the IP address of your DNS server (which is the server I configured above which is fda8:06c3:ce53:a890:0000:0000:0000:0001).
- The screen will look something like the following:

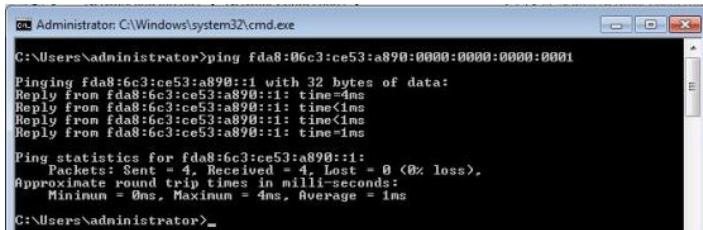


- 9) Click OK, OK, Close to save and exit

Running IPConfig /All to see what the configuration looks like, you'll notice the IPv6 Address, Default Gateway address, and DNS Server address have all been truncated down to the simplified format. In my case, I disabled IPv4 so there is no IPv4 address reporting on this system.



What you'll find when you ping from the workstation to the server, just like pinging an IPv4 address (but longer) is:



```

Administrator: C:\Windows\system32\cmd.exe
C:\Users\administrator>ping fda8:06c3:ce53:a890:0000:0000:0000:0001
Pinging fda8:6c3:ce53:a890::1 with 32 bytes of data:
Reply from fda8:6c3:ce53:a890::1: time=1ms
Reply from fda8:6c3:ce53:a890::1: time<1ms
Reply from fda8:6c3:ce53:a890::1: time<1ms
Reply from fda8:6c3:ce53:a890::1: time=1ms
Ping statistics for fda8:6c3:ce53:a890::1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 4ms, Average = 1ms
C:\Users\administrator>_

```

This is where DNS really helps out in IPv6. You'll go crazy trying to remember the IPv6 addresses, so you'll rely more and more on DNS, which I'll cover now...

### How to set up DNS for IPv6 on a Windows 2008 R2 Server to do name resolution of IPv6 systems

As you probably know, DNS is the name resolution that translates a common server name (ie: AD.companyabc.com) with its IP address (ie: fda8:06c3:ce53:a890:0000:0000:0000:0001). Obviously it is a lot easier to remember the common name than the IPv6 address, thus DNS becomes very valuable in an IPv6 environment.

To setup and configure DNS, two major things need to be done:

- 1) You need to configure a Windows 2008 R2 Server with an IPv6 address (like we did in the server configuration step above).
- 2) You need to install the Windows Server DNS role to the server. Quite frankly, if you've ever setup DNS in a Windows environment before, there is nothing different other than manually creating an IPv6 reverse lookup zone. So DNS setup is just simple DNS setup but on a Windows 2008 R2 server to support IPv6 addressing.

For the first step, build a Windows 2008 R2 server and give it an IPv6 address like you did above. Your DNS server should have a good IPv6 address before you proceed so that your server will be listening properly to the right IPv6 network/subnet for DNS requests. But if you already have DNS, you can still just update the IPv6 address and DNS will now start working properly for IPv6.

For the second step of making it a DNS server, do the following:

- 1) Logon to the server with administrator rights
- 2) Click on Start | Administrative Tools | Server Manager
- 3) Click to highlight Roles, then click on Add Roles
- 4) For the Before you Begin, click Next
- 5) Select DNS Server (so it is check-marked), click Next
- 6) For the Introduction to DNS Server, click Next
- 7) Click Install

This will install DNS on the Server.

Once DNS is installed on the system, to configure it as a DNS server (assuming it is part of an Active Directory domain), do the following:

- 1) Click on Start | Administrative Tools | DNS.
- 2) Highlight the computer name and choose Action | Configure a DNS Server.
- 3) Click Next, then choose either forward lookup zone, forward and reverse lookup zone, or root hints only (if in doubt, choose forward lookup zone).
- 4) Select Yes, create a forward lookup zone now, click Next.
- 5) Select Primary zone (if this is the first DNS server) or Secondary zone (if you already have a DNS server) or select Stub zone (if you know what you are doing and you want a stub zone configure), chose one of the 3 and then click Next.
- 6) Choose IPv4 Reverse Lookup Zone. (Assuming the server is running both IPv4 and IPv6 we'll set the IPv6 lookup zone later). Click Next.
- 7) Enter in the IPv4 network address in the NetworkID fields (this is the start of the IPv4 address of your network adapter), click Next
- 8) When prompted to create a new Zone file, click Next
- 9) For forwarders, click on Yes and enter in the DNS address of a public DNS server that you want to resolve non-AD addresses (this would typically be provided to you by your Internet Service Provider as a DNS address of a major public DNS server on the Internet). Click Next.
- 10) Click Finish to continue.

With DNS configured on your server, you will begin to see DNS records populated on the server either from devices if this is a primary DNS server, or from the other DNS server if this is a secondary DNS server. To force devices to register to DNS, go to other workstations or servers that have a proper IPv4 or IPv6 address and run the command `IPConfig/RegisterDNS` from a Command Prompt on the system. Go back to

the DNS server and refresh your DNS screen to see the devices start showing up.

Note: If you already have DNS working on an AD 2008 R2 server, and you have a proper IPv6 address configured for the server, DNS does not need to be reconfigured or rebuilt. DNS (if it was working for IPv4) will work for IPv6 natively on a Windows 2008 R2 server with IPv6 enabled.

To create an IPv6 Reverse Lookup Zone, since the configuration wizard only allowed you to create either an IPv4 or IPv6 reverse lookup zone, and we chose to create an IPv4 since likely you'll still be using IPv4 for a while, you have to configure the IPv6 Reverse Lookup Zone manually. To do so, right click on the "Reverse Lookup Zones" in the DNS manager and choose New Zone.

- 1) Click Next to start the New Zone wizard.
- 2) Choose Primary Zone (assuming this is the first DNS v6 server in your environment) and click Next.
- 3) Choose the option "To all DNS servers running on domain controllers in this domain" (or you can alternately choose forest if you have a single forest / single domain simple model). Click Next.
- 4) Select IPv6 Reverse Lookup Zone, click Next.
- 5) Enter in the IPv6 Prefix, which in our case would be fda8:06c3:ce53:a890::/64 (which will create a reverse lookup zone of 0.9.8.a.3.5.e.c.3.c.6.O.8.a.d.f.ipv6.arpa). Click Next.
- 6) Assuming this is on Active Directory, choose "Allow only secure dynamic updates." Click Next.
- 7) Click Finish.

That's it. You now have workstations and servers with your IPv6 addressing scheme setup, and you have DNS that is resolving not only your IPv4 names, but also now your IPv6 names.

Next up, I will cover how to setup DHCP for IPv6 to dynamically issue addresses in your block of IPv6 addresses. This is a little trickier. While the DHCPv6 tool is the same as the normal DHCP IPv4 tool, and the setup of DHCP for IPv6 is basically the same, there are a couple quirks that I bet have hung up anyone who has ever tried to setup DHCP for IPv6 in a Windows environment. I bet 99 percent of the people who tried gave up at that point with DHCP in IPv6. I'll share with you the workarounds to getting DHCPv6 to work the way you would expect it to work. Stay tuned ...

# Setting up DHCPv6 to Dynamically Issue IPv6 Addresses

Providing DHCP for IPv6 Devices in your Microsoft Windows Network

BY RAND MORIMOTO

**A**s promised, I am now going to cover how to set up DHCP for IPv6 to dynamically issue addresses in your block of IPv6 addresses. DHCPv6 is effectively doing the same thing as DHCP in that it dynamically issues IP addresses to systems. But instead of issuing just IPv4 addresses, we will be issuing IPv6 addresses as well. The concept is identical to issuing IPv4 addresses: you need to assign a block of IPv6 addresses you want to dynamically assign, you need to know the IPv6 address for your DNS server, and that's it.

In continuing on the example of static IP

addresses I used previously, I'm going to use similar IPv6 addresses and have made the following assumptions:

- I'm going to be using Unique Local Addresses (ie: private IPv6 addressing), thus my prefix starts with fd.
- I'm going to use a randomly selected GlobalID (a8:06c3:ce53) and SubnetID (a890) so that all of my devices will start with fda8:06c3:ce53:a890.
- My DNS server that I created in the last blog is addressed fda8:06c3:ce53:a890:0000:0000:0000:0001, the Gateway to another Subnet is fda8:06c3:ce53:a890:0000:0000:0000:0005 (which the truncation method gets rid of extra zeros so they look like fda8:6c3:ce53:a890::1 and fda8:6c3:ce53:a890::5).

- And for my DHCP server, I'm going to give it a static IP address of fda8:06c3:ce53:a890:0000:0000:0000:0004.

## How to set up DHCPv6 for IPv6 on a Windows 2008 R2 Server

As you probably know, DHCP issues IP addresses to systems when the system boots and needs an IP address, saving you from having to go to each system and statically addressing the systems (especially with the crazy long IPv6 addresses).

To set up and configure DHCP, three major things need to be done:

- 1) You need to configure a Windows 2008 R2 server with an IPv6 address. (I covered statically addressing IPv6 server addresses

in my previous post).

2) You need to install the Windows Server DHCP role to the server.

3) You need to configure the DHCP server role to issue IPv6 addresses.

Note: If you already have a fully working DHCP server running on Windows 2008 R2, you can skip the section on installing the basic DHCP role and just jump right to configuring the IPv6 scope. A fully working DHCP server on Windows 2008 R2 works fine for DHCPv6. It's installed/setup exactly the same. So all we are really doing is adding in an IPv6 "scope" to a working DHCP IPv4 server.

For the first step, build a Windows 2008 R2 server and give it an IPv6 address (as noted, we'll be using fda8:06c3:ce53:a890:0000:0000:0000:0004 as the IP address for this DHCP server, but your IP address can be anything as long as it is on the same subnet as your DNS server, domain controller, etc). Just make sure you can ping the other servers on your network and if you can, then your server is ready to go!

For the second step of making the server a DHCP server, do the following:

1) Logon to the server with administrator rights.

2) Click on Start | Administrative Tools | Server Manager.

3) Click to highlight Roles, then click on Add Roles.

4) For the Before you Begin, click Next.

5) Select DHCP Server (so it is check-marked), click Next.

6) For the Introduction to DHCP Server, click Next.

7) For Network Connection Binding, assuming you are issuing dynamic IPv4 addresses, select the IPv4 address for the DHCP server (which if you only have one IP address on the server, it will already be checked). Click Next.

8) For the DNS setting, enter your parent domain (this is the domain name of your network, ie: companyabc.com), and enter in the IPv4 address of your DNS server. Click Next.

9) Most orgs are no longer using WINS so you can likely say "WINS is not required." Click Next.

10) For your DHCP v4 scope, click Add and enter in the IPv4 range you want to issue IPv4 addresses including the Subnet Mask.

Click OK, then click Next.

11) For DHCP Stateless mode:

a) If your routers are setup to support IPv6 with the otherconfig=true, effectively that your router is configured to tell your IPv6 clients their routing information, then choose to Enable DHCPv6 Stateless Mode. Click Next. (Note: if you choose this mode, and your DHCP server issues IPv6 addresses and you get an error when trying to Ping IPv6 devices that look like "transmit failed. General failure," then configure your routers to support the otherconfig=true setting, --or-- reinstall DHCP to Disable Stateless mode, --or-- run the manual Add Route commands I note below (in that order of preference).

b) If your routers are not setup to support IPv6 (and cannot be configured to support otherconfig=true, then choose the option to Disable DHCPv6 Stateless Mode, click Next

12) For Parent Domain, enter in the name of your domain again (ie: companyabc.com in my case, same as step #8 above).

13) For preferred DNS server, enter in the DNS server we have configured for this scenario which is fda8:06c3:ce53:a890:0000:0000:0000:0001 (click Validate to make sure it resolves), clear the Alternate DNS (unless you have an alternate DNS, if you have an alternate DNS server, then enter it in and Validate that system). Click Next.

14) To authorize this DHCP server, choose "Use Current Credentials" assuming you are logged in as the domain administrator. Click Next.

15) Click Install.

This will install DHCP on the Server.

Once DHCP is installed on the system, it'll be ready to issue IPv4 dynamic addresses, but you need to configure it to support IPv6 dynamic addresses. To configure it as a DHCPv6 server, do the following:

1) Click on Start | Administrative Tools | DHCP.

2) Highlight (and Expand the computer name.

3) Highlight the IPv6 container, right click the container and choose New Scope.

4) Click Next through the Welcome screen.

5) Enter in IPv6 DHCP Scope (or whatever you want) for the Name. Click Next

6) For Prefix, in our case, we will enter in the Network and Subnet: fda8:06c3:ce53:a890::

(note the two : there before the default /64 on the screen). For Preference, leave the default at 0 (if you have multiple DHCP scopes, you can change the priority of which DHCP scope gets priority/preference for issuing addresses. Assuming this is the first and only DHCP server for now, leave it at 0). Click Next.

7) For exclusions, enter in any static IPv6 addresses you've already created. For our case we have issued static IPv6 addresses for our DNS server, our DHCP server, our gateway, so we would add fda8:06c3:ce53:a890:0000:0000:0000:0001, fda8:06c3:ce53:a890:0000:0000:0000:0002, fda8:06c3:ce53:a890:0000:0000:0000:0004, f d a 8 : 0 6 c 3 : c e 5 3 : a 8 9 0 : 0 0 0 0 : 0 0 0 0 : 0 0 0 5 (or more easily exclude the "range" from :0001 to say :00ff). Click next.

Note: You will see that with the IPv6 Scope, you cannot name a specific range (ie: :0100 to :01ff), it's going to pull from any of the available addresses in the entire device address range, thus you need to exclude your static servers.

One comment I get all the time is "then I should put my static servers in one subnet, and my dynamic devices in another." You could do that so you don't have to exclude all static addresses in the DHCPv6 scope, but Windows DHCP provides a concept called "DHCP Reservations" for setting pre-reserved static IP addresses in DHCP. It's a new way where "everything" (including domain controllers and servers) are dynamically addressed, but when DHCP sees the name of a specific server, it'll always assign that server a specific IP address you designate.

I will cover this in a separate blog post but I think it is due a mention here when discussing DHCP designs. So for now, just know that "all" IP addresses in the entire IP device range will be up for grabs, BUT if you want to narrow the scope, then just exclude "everything" except for a small range (ie: exclude 0000:0000:0000:0001 to 0000:0000:0000:ffff and exclude say 0000:0000:0010:0000 to ffff:ffff:ffff:ffff which will only give a very tight range (0000:0000:0001:000 to 0000:0000:000f:ffff) to be issued IP addresses for these dynamic devices)

8) Specify the life of the lease (default is fine for this case). Click Next.

9) Have Active Scope Now (Yes), then click Finish.

### Troubleshooting client problems (getting a “transmit failed. General failure.” when trying to ping other IPv6 clients after DHCPv6 is setup)

If you have a problem where after DHCPv6 is set up your client systems cannot access other IPv6 systems and you get a transmit failed error, as noted above in the configuration settings, you have three options. The best of the options is to run DHCPv6 in a Stateless autoconfiguration mode and set your routers with the otherconfig=true setting. But if your routers are not IPv6 supported (yet), you can reconfigure DHCPv6 to Disable Stateless mode, and that'll issue IPv6 addresses that will eliminate the Ping problem.

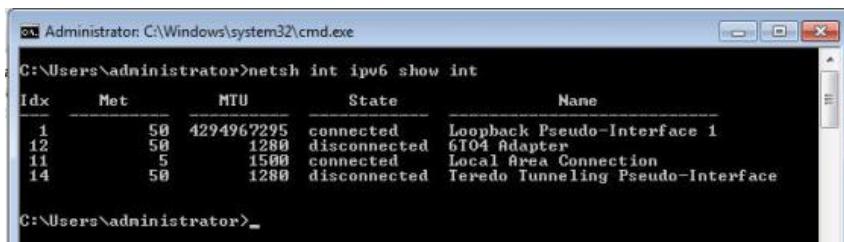
Or there is a workaround by manually setting Routes on your client systems. The appropriate way to address this is to add otherconfig=true on your routers on your network, which will have IPv6 devices looking for a route/gateway to get out of the subnet and automatically acknowledge the subnet that the device is on. If your router supports this configuration, then you do not need to proceed with the manual command configurations I will discuss in the next paragraph. If your routers do not support the otherconfig=true configuration setting, then upgrade your router firmware, or you may need to upgrade/replace your internetworking equipment to have IPv6 support so that this can be configured at the router.

If your router does not support the otherconfig=true configuration, or you won't be purchasing new internetworking equipment for a while but still want to get IPv6 working on your clients and servers, then proceed with the following manual settings. If you don't have supported routers, or if you don't do this workaround, you'll get an error when you try to Ping anything – the “transmit failed. General failure” error. You may just scratch your head forever and never figure it out. The reason you have to add these commands is that while DHCPv6 issues the IPv6 address to a client, it is missing the /64 route needed for the client system to access servers on the subnet. If you go to a freshly DHCPv6-addressed client and type netsh interface ipv6 show route you'll see /128 there, you'll see other routes, but no /64 route for your specific subnet, thus the DHCPv6 addressed client can't see any systems on its own network. If you statically address the client, it works fine (statically address a workstation and do the same netsh command and you'll see the /64 address show up). This

is what we are manually having to insert for ALL DHCPv6 issued clients.

The commands you need to run on a DHCPv6 issued client are as follows:

- 1) Run an elevated command prompt on the client system (cmd.exe).
- 2) Type Netsh int ipv6 show int to display a list of connected and disconnected network adapters. You're looking for your default adapter (on my system it is 11). (see figure below)



- 3) Type Netsh interface ipv6 set interface {# you identified in step 2} advertise=enabled
- For my example: Netsh interface ipv6 set interface 11 advertise=enabled

- 4) Type Netsh interface ipv6 add route 1024://64 {# you identified in step 2} publish=yes

For my example: Netsh interface ipv6 add route 1024://64 11 publish=yes

- 5) Type Netsh interface ipv6 add route {your prefix://64} {# you identified in step 2} publish=yes

For my example: Netsh interface ipv6 add route fda8:06c3:ce53:a890://64 11 publish=yes (see figure below)

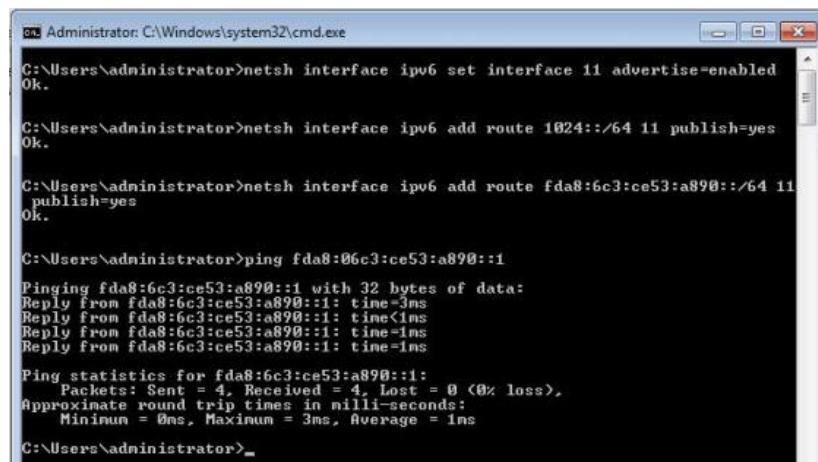
Test DHCP to see if it is working. Have a

server or workstation with DHCP selected for the IP address of the system and see if the system pulls a proper IPv4 address from the IPv4 scope, and a proper IPv6 address from the IPv6 scope. See if you can ping a server like your DNS server. In my example: ping fda8:06c3:ce53:a890:0000:0000:0000:0001 -6 (the -6 will ping over IPv6).

If you need to run these Netsh commands, you can run them in a batch file to execute when you configure the system. If you have

IPv4 available and can access an IPv4 share, then run the batch file off an available IPv4 share to get IPv6 running. If you are running IPv6 only, then you need to put the batch file on a USB stick and run it manually on the system (yeah, I know, lame...) You only need to do this once per system, so once you get this working, you don't have to deal with it again unless you change the network adapter of the system, and then you'll have a different “interface” (ie: mine was 11 above). Then you'll need to run the commands to the new interface as it responds to the system.

That is it for DHCP in IPv6. You'll find that doing dynamic addressing in IPv6 to be a preference over typing in IPv6 static addresses into systems.



# Configuring IPv6 Routing Through IPv4 in Windows

Providing support for IPv6 in a world still predominantly IPv4-based

BY RAND MORIMOTO

**C**onfiguring IPv6 Routing through IPv4 is a pretty important topic. As you set up IPv6 in your internal network environment as well as your client systems, you may come to a point where your IPv6-configured clients plug their computers into WiFi Hotspots that ONLY issue them IPv4 addresses. Their systems can communicate IPv4, but they can't get to any of the IPv6 servers in your environment. (Of course you will likely be on both IPv6 and IPv4 for a while so that your users will just communicate over IPv4 back to the office, but then why did you spend all the time to get to IPv6? Oh, that's right, IPv4 is running out of addresses and we all need to migrate to IPv6.) Here I will cover the step-by-step process of setting up "transitional technologies" that will allow you to have IPv6 clients that will route their traffic through an IPv4 network, back to your office that is running IPv6. Transitional routing is basically how you can set up IPv6 on your systems today, and route IPv6 over an IPv4 network.

I've also got a very good example of a valuable Windows 7 feature (DirectAccess) that will allow your remote endpoints to connect using IPv6 (over IPv4 networks) with encrypted and policy-based security controls back to your office.

In this blog post on IPv6, I'm going to cover:

- How to support transitional routing of IPv6 through IPv4
- Configuring 6to4 and Teredo
- Configuring IP-HTTPS and Microsoft DirectAccess
- Understanding Tunnel Brokers

## How to support transitional routing of IPv6 through IPv4

Transitional routing is needed because much of the public internet is still plain IPv4. So even if you set up your laptop with IPv6 and all of your servers at your office are IPv6, when you are at your office everything is completely IPv6, the minute you go to an Internet café, you can't access your office because the IP address you get from the Internet café is an IPv4 address. So transitional routing encapsulates your IPv6 traffic into IPv4 packets and tunnels your IPv6 traffic over the IPv4 network.

## Configuring 6to4 and Teredo routing

6to4 and Teredo are two of transitional routing standards for IPv6, which Windows 2008, 2008 R2, Vista, and Windows 7 fully support these standards. In fact when you setup IPv6 and IPv4 on your system(s), when you type `IPconfig /all`, you may notice the 6to4 and/or Teredo routing setup. But you may also notice that while it notes 6to4 or Teredo, many times it'll note "media disconnected" as shown below:

```
Administrator: C:\Windows\system32\cmd.exe
Tunnel adapter isatap.6667E509B-350D-4DE9-AA0C-A52DB2DF7DDF:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : companyabc.com
Description . . . . . : Microsoft ISA6to4 adapter #2
Physical Address. . . . . : 00-00-00-00-00-00-00-00
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes

Tunnel adapter Teredo Tunneling Pseudo-Interface:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Teredo Tunneling Pseudo-Interface
Physical Address. . . . . : 00-00-00-00-00-00-00-00
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
```

What that typically means is the "adapter" for Teredo and 6to4 have not been setup on your system. You need to install the "network adapter" drivers for Teredo and 6to4 for them to work. To do so, do the following:

- 1) Start | Control Panel | System and Security | System | Device Manager
- 2) Expand Network Adapters and see if 6to4 and Teredo are there, likely not (just your network adapter).
- 3) Add the 6to4 and Teredo by selecting Action | Add Legacy Hardware.
- 4) Click Next through the Welcome screen.
- 5) Choose to "Install the hardware that I manually select (Advanced)." Click Next.
- 6) Click on Network Adapters, then Next.
- 7) Click on Have Disk, then Cancel and wait a couple seconds. A list of Manufacturers will show up.
- 8) Choose Microsoft and select Microsoft 6to4 Adapter, Click Next, Next.
- 9) Click next to install.
- 10) Repeat this process to install Teredo and IP-HTTPS.

Now, after installing the adapter drivers, you might find that in Device Manager, the adapters still don't show up, basically just your physical network adapter. Or you might find that these new adapters installed, but they have a yellow triangle with an exclamation point noting that the adapters "did not start". The common reason for the Teredo and 6to4 adapters not to install or start is because your physical LAN adapter does not support Teredo or 6to4. Check with the hardware vendor that makes your network adapter and download and install the latest physical network adapter driver and see if that solves the problem. I've also seen users running virtualized Windows guest sessions on an Apple Mac to try to configure 6to4 or Teredo on the virtual guest session. That won't work. Being that your actual network adapter is the physical adapter of the host (Apple Mac) system, the Teredo and

6to4 setup needs to be done on the adapter of the host, not in the virtual guest session. I have successfully gotten 6to4 and Teredo to work within a HyperV host guest session, so if you wanted to test these transitional technologies on a HyperV guest in the lab, or your production routers are going to be virtualized guest sessions, this all works under HyperV.

(Hint: Before you waste a lot of time trying to get a Teredo or 6to4 driver to configure and install on your system, keep reading this article because in about six paragraphs down, I'm going to tell you that Teredo and 6to4 likely aren't going to be necessary and instead encourage you to work to configure IP-HTTPS instead).

So you might have to tinker with drivers to make sure you get the drivers installed and working.

Once you get the hardware stuff working, now you have to do the configuration.

To configure Teredo routing, on the "server side", you need to have a server with two consecutive public IP addresses (ie: x.x.x.100 and x.x.x.101). Teredo on a Windows server needs this as a listener adapter port, and a Teredo communications adapter port. On the server that'll be your Teredo endpoint, you would then type in `Netsh interface Teredo set state server {the first IP address}`, so it'll look like `Netsh interface Teredo set state server x.x.x.100` (with x.x.x.100 being the first of your two consecutive IP addresses).

On the client side, you would type in `netsh interface ipv6 set teredo client`. This will enable Teredo on your Windows 7 client (assuming your hardware supports the Teredo adapter and everything installed properly in Device Mgr). You now need the client to know where your Teredo server is, so on the client you would type in `Netsh interface teredo set state server=x.x.x.100` (with x.x.x.100 being the first of your two consecutive IP addresses of your Teredo server). With the Teredo server and the Teredo client configured, your client can communicate IPv6 (over IPv4) to the end server.

If you want the Teredo server to be a relay so that it accepts this remote client traffic and relay the IPv6 traffic on to another IPv6 server in your network, then on the Teredo server, you would type in `Netsh interface ipv6 set interface {interface #} forwarding=enabled` (where you can find the interface # by typing `Netsh int ipv6 show int` and look for the enabled

connection, usually the Local Connection, that corresponds to the working network adapter. When I covered this in the article on DCHPv6, I gave a screenshot that shows this command and that my network adapter was 11, so in that case the command would be `Netsh interface ipv6 interface 11 forwarding=enabled`. (See previous article: Providing DHCP for IPv6 Devices in your Microsoft Windows Network for more on finding the interface and running the netsh command).

So, all of the above will get Teredo to work so that your IPv6 client sitting at some Internet café can route IPv6 traffic through IPv4 back to a Teredo server (or relay server) that will then get the IPv6 traffic to your end destination. BUT, after all this, most of the time, the routing of IPv6 traffic still won't work! (Hopefully you read through the article and got to this point before you tried to configure it). 6to4 does not route through NAT and because when you go from hotspot to hotspot, you are almost always setup on the inside of NAT, 6to4 plain won't work. And in order for Teredo to work, IPv6 packets are encapsulated inside UDP packets, which are frequently not routed or specifically blocked by routers and firewalls. For transitional routing to work, the IPv4 WiFi hotspot or wherever your user is connecting to must have routers that support the routing of Teredo communications. For WiFi access points and routers that are at Internet cafes, hotels, airports, coffee shops, I rarely find Teredo will work.

So I've found it kind of useless trying to get 6to4 and Teredo to work as I travel around the world from hotspot to hotspot.

### Configuring IP-HTTPS & Microsoft DirectAccess

Okay, so since Teredo and 6to4 rarely work in the real world because most routers are not configured to allow Teredo or 6to4 traffic to route through the systems, what's one to do? This is where Microsoft came up with IP-HTTPS. IP-HTTPS is effectively taking IPv6 traffic and tunneling it within an HTTPS stream. This is exactly like RPC over HTTPS that we've been doing for years with Outlook to Exchange. Since all WiFi hotspots allow HTTPS (SSL) traffic, having your IPv6 traffic slip through this HTTPS tunnel is the way Microsoft accomplishes this, and any time I'm traveling with my laptop and communicating over IPv6, I'm either talking to my servers natively over IPv6, or my laptop tries Teredo (and fails), tries 6to4

(and fails) and then tries IP-HTTPS, and as long as I have a good basic Internet connection, my laptop is able to communicate back to my office using IPv6 over IP-HTTPS.

Rather than going into the step by step of configuring IP-HTTPS generically, what I'm going to do is refer you to content I wrote on Microsoft DirectAccess. DirectAccess is one of the first apps that Microsoft is providing that uses IPv6 natively and provides a VPN-less secured and encrypted tunnel from Windows 7 clients back to and into a company's network. DirectAccess uses IPv6. In the real world, since I find myself seeing my laptop connecting over IP-HTTPS for DirectAccess, it's a great tool to setup, configure, and experience IPv6 (and IP-HTTPS) firsthand.

For my content on DirectAccess, the information is as follows...

Deployment Guide for DirectAccess (link): This is a step by step guide I wrote on installing and configuring DirectAccess. I go through the process of configuring all of the server points, and configuring systems (Windows 7 end client and the DirectAccess server). The DirectAccess server is actually an IPv6 "router." It is configured to support Teredo, 6to4, and IP-HTTPS for incoming communications and the end client communicates with the DirectAccess server. Most of what I covered earlier in this article on setting up two consecutive IP addresses for the external NIC, setting up forwarding, etc., are all embedded in this configuration guide.

Video on Setting up DirectAccess (link): I actually led a 70-minute session at TechEd 2010 where I went through this deployment guide I reference and go through a step by step implementation of DirectAccess. Between the deployment guide, and watching the video, much of the basic IPv6 configuration stuff covered in the past four technical blog posts I've written on IPv6 now start to come together.

It's actually fun to fiddle with DirectAccess (although warning, it is not a simple whip-together-in-a-couple-hours solution. It's for those who really like to get technical with this stuff and like a challenge...) But what you'll find with DirectAccess is a real world solution based on IPv6 that utilizes IP addressing to prove that you can really use IPv6 for practical purposes!

### Understanding Tunnel Brokers

Okay, before I end this blog post on routing IPv6 through IPv4, I need to touch on Tunnel Brokers. There are "tunnel brokers" out there that are available to help you take your

IPv4 and IPv6 client systems, and access IPv6 content on the Internet. Effectively, you load on a driver / applet on your client system, and you establish a tunnel from your system, through the Tunnel Broker's network to access servers and services, things like <http://ipv6.google.com> over IPv6.

For the individual or small business that doesn't want to setup a DirectAccess server and a relay/routing setup, the Tunnel Broker can drastically simplify the process. Say for example you want to access the latest IPv6 Facebook site from your home office, but your DSL provider doesn't have IPv6 support yet and your company hasn't setup IPv6 routing on their end, you can simply download the Tunnel Broker driver(s), install them on your home system, and within about 10 minutes, your home system is now configured using IPv6 to Facebook's IPv6 site.

For larger businesses, you will likely want to setup your own IPv6 architecture and get ahead of the curve so that your employees are configuring their home systems to communicate IPv6 over a corporate tunnel back to your office using something like Microsoft DirectAccess. You can do all of the IPv6/IPv4 tunneling on your own straight from your end users to your corporate network without going through a middleman like a Tunnel Broker, but they serve a purpose for small businesses or individuals who may want or need IPv6 functionality without having to setup an entire IPv6 infrastructure.

There are a handful of Tunnel Brokers out there, one that serves the United States is Hurricane Electric. This Wikipedia site lists various Tunnel Brokers around the world.

So that's it on routing IPv6 over an IPv4 network. With the past handful of blog posts

on IPv6, you now have the basis for getting official IPv6 addresses, assigning them to servers and workstations (statically and dynamically), and providing a way for your end clients to communicate IPv6 to your network, even though they may be connecting to an older IPv4 WiFi Hotspot or network connection.

My next blog post will be titled "Best Practices at Configuring Applications for IPv6" where I'm going to address getting things like Exchange, SharePoint, Web servers, and the like active on IPv6. I'll address server naming (ie: do you keep the same name for your IPv6 servers as your IPv4 servers, or do you make special [ipv6.companyabc.com](http://ipv6.companyabc.com) server addresses), and the like. On to application servers...

# Best Practices for Configuring Applications for IPv6

Getting Apps like Exchange, SharePoint, Web Servers on IPv6

BY RAND MORIMOTO

**I**n this posting, I'm focusing on application servers, like Exchange, SharePoint, Web servers, and the like. I'm going to cover configuring servers (like Exchange, SharePoint, Web) with IPv6, and best practices on naming IPv6 servers.

## Configuring Servers (like Exchange, SharePoint, Web) with IPv6

For the past three or four years, those of you who have configured Exchange 2007, Exchange 2010, SharePoint 2010 have run into quirks with IPv6, and up until now, for most IT Pros, the whole IPv6 thing was a bit of a mystery. We've known IPv6 was there on our Windows 2008 servers, we didn't configure it, yet it was showing some IP address, and when there's something going on with our network that we don't understand, we

just disable it. And that's been the problem with Exchange 2007 and 2010. Microsoft has keyed services and dependencies on the IPv6 adapter port, so those who disabled IPv6 would get errors like "MSEXchangeTransport failed to reach status 'Running' on this server" or other odd Exchange errors. And when you searched TechNet for the fix, you would find that you would have to enable IPv6 and reboot the server and the problem went away, so that's what most people did.

Now, the IPv6 address that you were getting (since you didn't statically address your server with IPv6, nor had a working DHCPv6 server on your network) is what is called a Stateless Autoconfigured IPv6 address, randomly assigned to the adapter, but dependent on how your routers are configured in your environment, you may actually find that when you try to ping something over IPv6 (ie: ping xxxx-6), you get a communications error. I cover this in my blog post on DHCPv6, about

Stateless Autoconfiguration, implementing `otherconfig=true` on routers, or manually setting router configurations. You can use this Stateless address, or (better yet), actually get an official block of IPv6 addresses and statically or dynamically configure them using DHCP Reservations so that you and your servers are managing your IPv6 addresses, address allocation, and configuration states.

From my previous blog posts, you know how to set actual IPv6 addresses and you know how to either statically (or dynamically) address servers and systems. So you now have Exchange 2007, Exchange 2010, SharePoint 2010, Web Servers, etc., coming up with real routable, usable, pingable, http-able addresses.

By the way, with DHCPv6, the minute you put in a DHCPv6 server and activate it, ALL IPv6 devices on the network that weren't configured in the past (sitting on Dynamic DHCPv6 settings) will grab an IPv6 address

from that first DHCPv6 server you setup. That's good and bad. If you are reading and implementing IPv6 blog post-by-blog-post, and you have enabled DHCPv6 in your production environment two or three posts ago, you may find that ALL of your hosts now have a DHCPv6 issued actual IP address on the systems! The good thing about it, you are now "done", all your systems have been addressed, and you are all set.

But if you really want to statically address your servers with IPv6 addresses, then you can go back to the systems and give them static IP addresses... After you statically address your servers (see my blog post on Static addressing servers with IPv6 for more info), you can run `IPConfig /RegisterDNS` on the systems so that their new IPv6 address shows up in your DNS properly.

That's pretty much it on server configuration for IPv6. Servers will get IPv6 addresses from DHCPv6 unless you statically address the servers before setting up DHCPv6. Once the systems have an IPv6 address, you can ping the servers, `http://` or `https://` the IPv6 addresses on the servers.

## Best Practices on Naming IPv6 Servers

So the next question people ask is whether they should give their servers dual names, so that a server has an IPv4 name and an IPv6 name. The answer really is "it depends." By default, DNS will pull the name of the server and associate that name with both an IPv4 and IPv6 address. And Windows 7 client systems with IPv6 configured and working will first attempt to access a server using IPv6, then through Teredo, 6to4, and IP-HTTPS. If it can't access via an IPv6 method, it'll access the system using IPv4. So from that basis, leaving the server name the same is a good fallback so that your users will know just one name of a server (ie: `owa.companyabc.com`, or `sharepoint.companyabc.com`) and the client system will walk through the process of connecting to the system with IPv6 and then fallback to IPv4.

There are times when you would want to give your server a specific IPv6 name so that you would make a DNS setting that differs whether a connection is IPv4 or IPv6 based. Remember by default, users will connect by IPv6 and then fallback to IPv4, so if the server has both an IPv6 or IPv4 address, just because you change the DNS name of the single server, users will connect to the server either as IPv6 or as IPv4. This is a good thing in normal practice so that you can support either configuration.

## Configuring Active Directory to Support IPv6

IPv6 has for the most part been available since Windows 2000 / Windows 2003, however the versions of IPv6 in the earlier releases (including Windows 2003) didn't include the current IPv6 standards. While you could get IPv6 to work for your Active Directory 2003 forest/domain, I would advise organizations to migrate their Active Directory to AD/2008 or all the way to AD/2008 R2.

Not only will you get the latest support for IPv6, but there are a number of other benefits and improvements in AD/2008 R2 specific to support around DirectAccess, a VPN replacement remote access technology built in to Windows 7 that actually uses IPv6 as the routing and IP security technology which requires at least one AD/2008 R2 global catalog server in the environment to run. Plus you'll get improved group policies that will help you better manage and administrator AD group policies that are very helpful in implementing an IPv6 environment. (Also see my two-hour workshop on What's New in Windows 2008 R2 SP1 and read a summary)

To get to an IPv6 supported Active Directory, you just need to make sure your Global Catalog servers are running IPv6 off the base operating system of the servers. That's it! Even if you didn't statically address your global catalog servers with IPv6 addresses before, you can just go into your network adapters of the GCs and DCs of your network, give them real IPv6 addresses, and that is it.

In other words, you really need to just have the basic static IPv6 addressing / DNS / DHCP stuff working, and AD is just another server with IPv6 configured.

For those looking to migrate from AD/2003 to AD/2008 R2, again, the workshop I did covers the step by step process of upgrading to AD/2008 R2. It's effectively joining a Windows 2008 R2 member server to the domain, run `DCPromo` to promote that "server" to become a domain controller of the domain (which the first server will also extend the schema), and then going into AD Sites and Services and making the Domain Controller into a Global Catalog Server. I actually covered the step by step migration process to AD/2008 R2 in a previous blog post.

But if you want to leverage the security and connectivity capabilities of IPv6, then having a "frontend" server running IPv6 and another frontend server running IPv4 setup to similar "backend" data (ie: Exchange backend, or SharePoint backend) will provide you isolated IPv6 or IPv4 traffic based on the frontend server you point the user to. With IPv6, not only do you get a new IP addressing scheme, but actually IPv6 embeds IP Security (IPSec) encryption built into the every day transmission of data. For organizations looking to provide policy-based end to end encryption of content, IPv6 communications through something like Microsoft DirectAccess provides that type of functionality both inside and outside of the network.

An IPv6 connected user session can be identified when they are inside the network, allowing access to all content on a server, whereas the same user might have limited access to protected content when they are remote, as opposed to an IPv4-connected user that in this scenario would be blocked from accessing any content when remote

because the remote user's session state, end to end encryption, and policy-based access control is not assured. (You can set all of this through policy.)

With this in mind, we have helped organizations implement SharePoint with separate server infrastructures where `SP.companyabc.com` hit servers that were only available inside the network, but `SPv6.companyabc.com` hit IPv6 SharePoint servers that allowed a handful of users to gain remote access to content even when the users were remote. To setup this configuration, simply setup the backend (SQL) with both IPv4 and IPv6, and setup two separate SharePoint servers with one running just IPv4, and another running just IPv6, with separate DNS names for the different servers.

As for naming of the servers running IPv6, there are no defacto standards. Google is providing `ipv6.google.com`, Facebook is providing `www.v6.facebook.com`. To not make a URL too lengthy (like `ipv6.sharepoint.companyabc.com`), just adding 6 or v6 to a normal URL provides users commonality of naming

like OWA.v6.companyabc.com, SP.v6.companyabc.com, CRM.v6.companyabc.com, and the like.

That's it on configuring applications for IPv6. There's not much to it and, in fact, for many organizations that may have put in a DHCPv6 server into their environment, they may have ended up with IPv6 on all of their servers after that task. Servers that are running on Windows 2003 should be upgraded to Windows 2008 R2 so that the latest IPv6 support is available for the systems. If you have Microsoft-based servers like File/Print servers, DHCP servers, or the like that you want to move from Windows 2003 or Windows 2008 to Windows 2008 R2, or you have physical servers on older Windows server

platforms that you want to make virtual servers on Windows 2008 R2, take a look at <http://www.microsoft.com/migration>. Microsoft has a series of migration guides and migration tools that help you move content from older platforms to newer platforms, and maintain security and configuration states. The file migration tool they provide allows you to point to an old Windows 2003 file server, it grabs all of the files off the server, and moves the files, shares, ACL (security permissions), everything over to a brand new Windows 2008 R2 server making it really easy to get to Windows 2008 R2 for applications. Same with DHCP migrations, if you want to migrate DHCP off an old Windows 2003 (or possibly 2008) server to a DHCP

server running Windows 2008 R2, there's a migration tool up on that link that'll move not only DHCP scopes across, but also moves over leases so that you can easily get from an old DHCP to a new DHCP with minimal (or no) interruption of service.

My next blog post will be titled "Planning Your Cutover to IPv6 for your Microsoft Windows Environment" where I'm going to provide the common business process of getting IPv6 implemented in a Microsoft Windows environment, from architecture, through setup, through final configuration. Basically I'll offer a chicken-and-egg approach where you can organize your migration strategy to an IPv6 environment.

# Planning Your Cutover to IPv6 for Your Windows Environment

## Getting to IPv6 in a Managed and Orderly Manner

BY RAND MORIMOTO

**I**n this blog post on IPv6, I'm going to cover what we've found to be a well organized and managed process of getting from an IPv4 environment to an IPv6 environment, effectively putting the chicken and the egg in the right sequence.

This is about planning and organizing your shift from IPv4 to IPv6 along with tips and guidance along the way.

### Step 1: Getting your IPv6 Address Block

As I covered in my blog post "IPv6 Addressing, Subnets, and Private Addresses," you need to get your IPv6 address block so you know what addresses you plan to use for your IPv6 implementation. If you plan to use Unique Local Addresses (ULAs) in IPv6, similar to private (internal) addresses in IPv4, you'll need to understand how those addresses worked in an IPv6 environment. (I covered that in a previous blog post.) Having

an IPv6 block of addresses is the first place to start

### Step 2: Architecture Design for IPv6 Mapping

The next step is determining how you want to set up your IPv6 architecture, effectively how you plan to subnet your IPv6 environment. A word of caution is to not just take your existing IPv4 subnetting configuration and apply the exact same subnets for IPv6. Take a moment to stop and think about your design. It's not any harder to change your IPv6 subnet structure if you want to, and now would be the time to do so.

A few things to take in consideration is that IPv6 does not have the same limitations as IPv4 had regarding subnet masks. If you were using a Class C addressing space in IPv4 using a 255.255.255.0 subnet mask giving you address space for 256 devices, with IPv6, your device boundary is 18-quintillion devices per subnet, so "masking" is no longer a barrier. However, with IPv6, we haven't changed the laws of physics nor the limitations of

Ethernet collisions, so even if you were to put 18-quintillion devices on a single physical subnet, the devices likely won't be able to communicate at all because of the Ethernet collisions occurring on the subnet. So there's a happy medium of not having a mental barrier of keeping subnets "the same" just because, to trying to stuff too many devices on a single subnet that creates collision in communications.

If you have virtual LANs (VLANs) setup, you can redo your VLAN structure when you switch to IPv6, and with many devices, you can have separate VLANs for IPv6 traffic than you have for IPv4 traffic, so with the same switch and router infrastructure, you can change your subnetting merely by adding IPv6 routing and communications to your environment. Again, there's a whole process of sitting down and putting together an IPv6 architecture that makes the best sense for the way your organization is currently structured, and not to merely mirror the design that may have been setup for IPv4 years (or decades) ago.

### Step 3: Implementing your First IPv6 Server

Since everything really depends on authentication and name resolution, the first servers that typically get implemented with IPv6 are Active Directory Global Catalog (GC) and Domain Controller (DC) systems, along with DNS servers. If your AD and DNS are still running on Windows 2003 / Active Directory 2003, this is where a migration from AD/2003 to AD/2008 R2 comes in. You would complete your migration off of AD/2003 into a fully IPv6/IPv4 ready AD/2008 R2 environment. It's not difficult to do although it takes a little planning to make sure all of your applications support AD/2008 R2 and the schema. Then you advance your environment to AD/2008 R2 and get all of your GCs and DCs and DNS servers up on IPv6 so you know your naming and authentication structure is setup and working properly with IPv6.

In my blog post on "Configuring Active Directory to Support IPv6", I provide step-by-step guidance on how to move an organization from AD/2003 or AD/2008 to AD/2008 R2.

### Step 4: Static Addressing

The next step is to statically address all servers and devices that you want to have IPv6 statically addressed. This is typically your application servers like Exchange, SharePoint, Web Servers, accounting servers, ERP servers, etc., both servers on Windows and servers not on Windows. Don't worry about upgrading "everything" to IPv6. This is not a requirement at this point, so if you have some old application that is running Windows 2000, or a version of Linux that doesn't support IPv6, don't sweat it. The whole purpose of statically addressing stuff with IPv6 addresses is so that when you get to the next step and put in a DHCPv6 server, those servers that have been waiting for a DHCP-issued address are statically addressed the way you want to address them.

My blog post on "IPv6 Static Addressing" covers this topic on a hands-on step-by-step guidance manner.

Effectively, take a good inventory of all of your systems, statically address stuff you can, and have a list of servers, systems, devices that haven't been addressed that you'll need to get to at some point.

### Step 5: Configuring DHCP

Now that you've statically addressed the

servers that you can address, you can setup DHCPv6 in your environment to push out IPv6 addresses for dynamic devices. This will get Windows Vista, Windows 7, Apple Mac Leopard and SnowLeopard desktops and laptops with a qualified IPv6 address.

My blog post on "Setting up DHCPv6 to Dynamically Issue IPv6 Addresses in a Network" covers this topic on a hands-on step-by-step guidance manner.

At this point, your client systems will be able to communicate IPv6 with systems on the subnet with IPv6 enabled. If your servers are on the same subnet as the client systems, you now have end to end IPv6 communications going internally.

### Step 6: Implementing IPv6 Remote Access

At this stage of an IPv6 initiative, implementing something like Microsoft DirectAccess is a great tool that leverages IPv6 that has been put in place in the previous five steps, but can work in an existing IPv4 environment through the implementation and use of Microsoft Unified Access Gateway (UAG) 2010. By implementing an IPv6 application and leveraging the policy-based VPN-less remote access capabilities of DirectAccess, an organization can start seeing and receiving the benefits of having IPv6 in the environment.

It's usually by this step that the organization needs something that users can see and use that utilizes IPv6, as the next step of getting to the InterNetworking piece is a longer / drawn out process, so using IPv6 through DirectAccess is a good addition at this point.

My blog post on "Configuring IPv6 Routing through IPv4" has a whole section on step by step implementation of DirectAccess along with configuration guides and an implementation video

### Step 7: Configuring Internetworking for IPv6

With client systems and servers configured to communicate IPv6, the next step is to inventory, identify, and configure internetworking devices such as switches, routers, appliances, gateways, WiFi access points, and other devices to support IPv6. As much as you can, get your internetworking devices all updated prior to even turning on IPv6 on your servers and clients. If you wait until you get the internetworking all setup before you get your servers and clients setup, it could be a long time before you get IPv6 going in your environment.

With IPv6-enabled clients and servers, for devices that are already IPv6 ready, a quick configuration of the device(s) and you now have IPv6 communications. So for this chicken-and-egg, we usually start with Windows servers and clients, and then get to the Internetworking devices in the configuration process.

Usually for upgrading or configuring internetworking devices, start with core routers and Internet connections so that your main points of communications are setup and working properly with IPv6. Eventually you get to wireless access points, remote sites, and other devices such as printers, fax gateways, and the like.

This could be a time consuming process, for some very large organizations, the internetworking piece can be months or several months to address IPv6 addressing and routing. In many cases, devices that don't have IPv6 support will need to be replaced (if they can't be updated). So having an inventory of all devices and systematically walking through the process of updating the device, setting proper IPv6 addresses, configure IPv6 routing, buying new hardware to replace old hardware, and the like will be a task of organization and coordination.

### Step 8: Final Assessment of IPv6 Completion

By this stage of the process, the organization will have IPv6 pretty well implemented throughout the environment. A task of going through and taking an inventory of all devices and validating that all devices are communicating over IPv6 is a scan of the internal environment to confirm configuration and operation of IPv6 (and not IPv4). For organizations that have very old systems, devices, and other equipment that are not, and will not be, IPv6 compatible, workarounds can be devised to continue to allow IPv4 for those specific systems or applications. Just knowing the state of the environment is helpful so that the organization has clear isolation of what is not switched over to IPv6.

That's the process... There are a lot of little things that you'll discover needs attention, but these are the main steps.

I hope this series was helpful! I really wanted to put a real world spin on HOW to actually implement IPv6 in a Windows network environment.

*Read Morimoto's blog at Network World*