




Getting Started with IPv6 on z/OS®

@business on demand.

© 2005 IBM Corporation

Agenda

- Why is IPv6 on z/OS important?
- What IPv6 features are supported by z/OS?
- How do you enable IPv6 support on z/OS?
- How do you configure z/OS CS IPv6 support?
- How do you access z/OS from a remote client?
- How do you verify that IPv6 is working?
- How do you manage IPv6 support on z/OS?
- What are some of the considerations when enabling IPv6 support?
- What are some of the steps to begin the transition to IPv6?



What is IPv6?

- IPv6 is an evolution of the current version of IP, which is known as IPv4
 - ▶ Work on new IETF standard started in early 90's
 - ▶ Not backward compatible, but migration techniques defined
- Today's IPv4 has 32 bit addresses
 - ▶ Practical limit is less than 1 billion useable global addresses
- IPv6 provides almost unlimited number of addresses
 - ▶ IPv6 addresses are 128 bits
 - ▶ No practical limit on global addressability
 - ▶ Enough address space to meet all imaginable needs for the whole world and for generations to come
 - ▶ More addresses *cannot* be retrofitted into IPv4
- Other improvements important, but secondary:
 - ▶ Facilities for automatic configuration
 - ▶ Improved support for site renumbering
 - ▶ End to end IP security
 - ▶ Mobility with route optimization (important for wireless)
 - ▶ Miscellaneous minor improvements

IPv4 Address:
 9.67.122.66

IPv6 Address:
 2001:0DB8:4545:2::09FF:FEF7:62DC

Page 3 | Getting Started with IPv6 on z/OS | © 2005 IBM Corporation

The Internet Protocol, or IP, is the basic building block on which all Internet applications are built. IP provides the mechanisms by which individual data packets are sent from computer to computer, over any mixture of networks links, routers and operating systems. Web, e-mail, instant messaging, remote database access, voice over IP – none would exist without the underlying IP service and its universal addressing system.

Today's internet runs on IPv4. IPv4 uses 32-bit addresses, which in theory allows for over 4 billion unique addresses. In practice, the usable number is less than 1 billion. This limited address space will eventually become exhausted, possibly as soon as 2009.

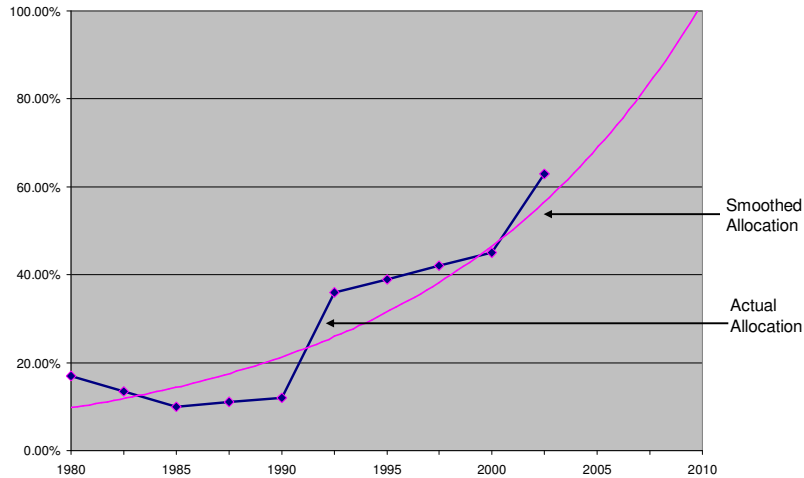
IPv6 is the latest version of the IP standard which is intended to progressively replace IPv4. IPv6 uses 128-bit addresses as compared to the 32-bit addresses used by IPv4, allowing for enough addresses to meet the anticipated needs for the foreseeable future. To give some perspective on how large a number of addresses we are talking about, IPv6 supports 35 trillion interconnected networks, each the size and complexity as those used by large companies such as IBM.

While the immediate benefit provided by IPv6 is the expanded address space, IPv6 also contains additional capabilities. IPv6 allows for automatic configuration of hosts in the network, using both DHCP and a new stateless autoconfiguration protocol. The enhanced autoconfiguration capabilities provided by IPv6 also allows for more seamless site renumbering. IPv6 provides end-to-end security with an adequate number of addresses to make this feasible. IPv6 has improved support for mobile clients. While some benefits provided by IPv6 can be retrofitted to IPv4, the lack of universal addressing in IPv4 means these solutions are cumbersome.

Despite these important changes, IPv6 is a conservative design. IPv6 does not change the fundamental approach to the IP routing infrastructure, DNS naming, QoS, firewall protection, or intrusion detection.


We have IPv4 addresses enough - or do we? Latest IPv4 address space usage overview

The chart shows IPv4 address allocation over time. The "blue" line is the actual allocation, the "purple" line is the smoothed allocation. Current extrapolations place the depletion of IPv4 addresses in the next 5-20 years.



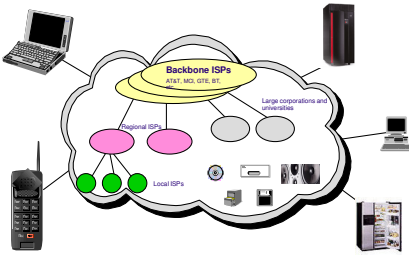
There has been concern since the early 1990s that continued world-wide growth of the internet will lead to the exhaustion of IPv4 addresses early in the 21st century. There is now much evidence that the exhaustion is starting. Restrictive address assignment policies have been implemented, with the result that countries like China or India get a grossly inadequate and fragmented address space, which leads to awkward allocation and management policies. Emerging computing applications, including 3G mobile telephony, cannot get the 100's of millions of addresses they each require.

The latest extrapolations put the exhaustion of the unassigned address space as early as 2009, with the most optimistic view within 20 years.



Trends driving IPv6

- Growing mobility of users
 - ▶ Internet access from anywhere (car, home, office)
 - ▶ Multiple addresses per person
 - ▶ Pervasive Computing
- Continued rapid growth of the Internet
 - ▶ China plans to roll out ~1 billion Internet nodes, starting with a 320 million student educational network
 - ▶ Asia/Pacific, and to a lesser extent Europe, missed out on the early IPv4 address allocations
- Government support
 - ▶ Wide-scale IPv6 promotion underway in China, Japan, Korea and Taiwan
 - ▶ European Commission (EC) encourages IPv6 research, education, and adoption in member countries
 - ▶ US DoD mandates support of IPv6 starting 10/2003
 - Other federal agencies must use IPv6 by June 2008, the White House Office of Management and Budget announced in June, 2005
- Convergence of voice, video and data on IP
 - ▶ Need for reliable and scalable architecture
 - ▶ "Always-on Connections"



Connectivity for **anyone** from **anywhere** (car, plane, home, office) to **anything!**

IPv6 promises true end-to-end connectivity for peer-based collaborative solutions

Page 5 | Getting Started with IPv6 on z/OS | © 2005 IBM Corporation

The requirements for IPv6 are being driven on many fronts. Certainly one is the increased mobility of users. Many of us now require Internet access regardless of where we are working – in the office, at customer sites, at home, or when traveling. We no longer have just one computer in the office, but more typically an additional computer or computers in our homes. Internet access from PDAs and phones is becoming commonplace, and computers are appearing in cars, refrigerators, even in the lights and doors within our homes. And each of these devices requires a unique IP address.

As the Internet makes its way into developing countries, the number of people accessing the Internet will explode. China and India each have populations of over 1 billion people, each needing one or more IP addresses. Simple math makes it apparent there aren't enough IPv4 addresses available, a problem only made worse by these countries having missed out on the early IPv4 address allocations.

As the use of the Internet to carry voice, video, and data continues to grow, the strain on IP address allocations will also grow. In order to be useful, these emerging peer-to-peer devices each require a unique IP address, much like each telephone requires a unique telephone number. Unfortunately, IPv4 lacks the 100s of millions of addresses each of these emerging technologies requires.

Governments around the globe have seen these problems and realize that IPv4 inhibits the continued growth of the Internet. Across Asia, Europe and, more recently, in the US, governments have been promoting and even mandating the use of IPv6.

IPv6 industry platform status

Platform	Availability	Status
AIX 4.3	10/1997	Support now available
z/OS	9/2002	Support now available; download OS/390 demo since 7/98
Cisco	7/2001	Support in IOS 12.2(2) T, with support for Catalyst switches to follow
MS Windows 2000	3/2000	Technical preview available with SP1 via the MS Developer's Network
MS Windows XP	10/2001	Developer's version included on Windows XP CDs; SP1 has a production-quality IPv6 stack
MS Windows 2003	2003	Production level IPv6 stack
Sun Solaris 8	2/2000	Support now available
Linux	Now	Evolving, code now available
FreeBSD, OpenBSD, NetBSD, BSD/OS	Now	All based on the KAME project (joint effort between 7 Japanese companies)
OpenVMS	3/2001	Compaq
Mac OS X	2003	Production level IPv6 stack
Other platforms	~30 versions	Quality variable

Sun Java 1.4.0 has IPv6 support built-in.

Lots of activity in this area. A good place to monitor is <http://www.ipv6tf.org>

Essentially every significant computer operating system and almost every network vendor now includes IPv6 support. Within IBM, AIX and z/OS include production level implementations. AS/400 has a developers version available which is suitable for application development but is not intended for production deployment. Microsoft, Sun, HP, Linux – all include IPv6 support.

Now that the base operating systems support IPv6, middleware adaptation to support IPv6 is underway.

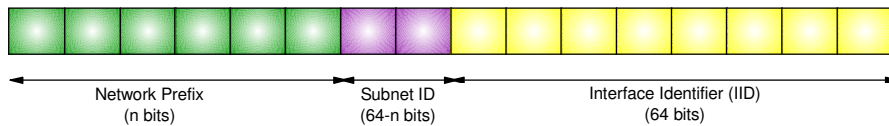
Important IPv6 technical features

- IPv6 header and extensions header
 - ▶ Streamlined IPv6 header
 - ▶ Optional extensions for fragmentation, security, etc.
- Routers no longer fragment forwarded datagrams
- Extended IP Address
 - ▶ 32 bits -> 128 bits (but only 64 bits for routing)
- Neighbor Discovery and Stateless Autoconfiguration
 - ▶ Router Discovery and Neighbor Unreachability Detection (NUD)
 - ▶ Address configuration with no manual or server-based configuration
- IPv4/IPv6 Coexistence and Transition Mechanisms
 - ▶ Coexistence for IPv4 and IPv6
 - ▶ Tunneling and transition mechanisms

IPv6 provides many important technical improvements beyond those found in IPv4. The IPv6 header is now fixed-sized, with each option appearing in its own extension header which is daisy-chained behind the IPv6 header. Expensive, slow-path operations, such as fragmentation, have been removed from the network and instead occur only at the endpoints. Most host configuration is now automated, allowing for improved plug-and-play capabilities. IPv6 also provides many transition and coexistence mechanisms to ease the migration from IPv4 to IPv6.

Expanded routing and addressing

- Expanded size of IP address space
 - ▶ Address space increased to 128 bits
 - Provides 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses
 - Enough for 1.8×10^{19} addresses per person on the planet
 - ▶ A 64-bit subnet prefix identifies the link
 - ▶ Followed by a 64-bit Interface Identifier (IID)
- IID derived from IEEE identifier (i.e., MAC address)
 - ▶ Only leftmost 64 bits available for routing and "network addressing"
 - ▶ The rightmost 64-bits identify the host on the target link



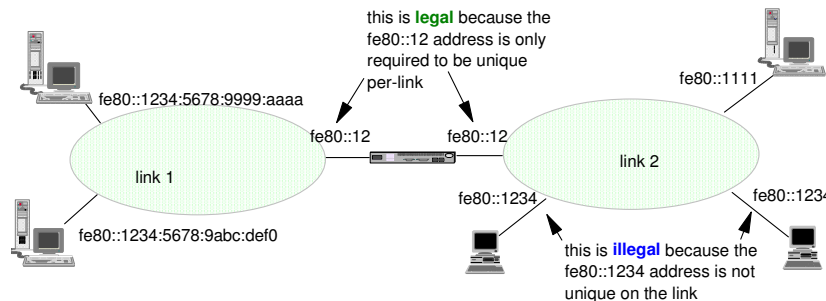
A unicast IPv6 address consists of two parts: the subnetwork prefix and the interface identifier, each of which is 64-bits in size. The subnetwork prefix is used to identify a specific link in the network, while the interface ID is used to identify a specific network interface adapter on that link.

The subnetwork prefix is further divided into two pieces: a network prefix and a subnet ID. The network prefix is used to identify a specific network which is connected to the Internet, while the subnet ID is used to identify a specific link within that network. Normally, the network prefix is 48 bits in size and the subnet ID is 16 bits in size, allowing for up to 64K links in a single network. If this isn't sufficient, an enterprise may request adjacent blocks of 48-bit prefixes from their ISP, effectively increasing the size of the subnet ID to 17, 18, 19 bits, or whatever size is needed to subdivide the network.

IPv6 addresses are owned by the ISP which provides Internet connectivity for the enterprise and not the enterprise itself. An ISP typically provides a 48-bit prefix to each enterprise which connects through the ISP. If an enterprise wishes to change ISPs, then the new ISP will provide a different 48-bit prefix and the enterprise will need to renumber its networks as part of the change-over. IPv6 includes support to aid in this change-over, which I'll discuss later in the presentation.

IPv6 scoped unicast addressing

- Concept of scoped unicast addresses part of architecture
- Link-local addresses for use on a single link
 - ▶ Primarily used for bootstrapping and infrastructure protocols such as Neighbor Discovery
 - ▶ Address = well-known link-local prefix plus node-generated IID
- Site-local addresses for use within a site
 - ▶ Like net 10
 - ▶ Full (negative) implications only recently understood
 - Application complexity
 - Nodes in multiple sites simultaneously
 - ▶ Has been deprecated by the IETF
 - An alternative approach "Unique Local IPv6 Unicast Addresses" is being pursued
- Global address prefixes are provided by ISPs



Every IPv6 address other than the unspecified address has a specific scope. A scope is a topological span within which the address may be used as a unique identifier for an interface or set of interfaces. The scope of an address is encoded as part of the address.

For unicast addresses, this document discusses two defined scopes:

- Link-local scope, for uniquely identifying interfaces within (i.e., attached to) a single link only.
- Global scope, for uniquely identifying interfaces anywhere in the Internet.

A scope zone, or simply a zone, is a connected region of topology of a given scope. For example, a specific link in a network, and the interfaces attached to that link, comprise a single zone of link-local scope. Note that a zone is a particular instance of a topological region (e.g., link-1 or link-2), whereas a scope is the size of a topological region (i.e., a link or a site).

Looking at the example above may help make this a little more clear. The router in the middle is connected to two links. The interface on each link has the same IPv6 address, `fe80::12`. This is valid, because a link-local address only needs to be unique on the link to which the interface it is assigned is attached. To uniquely identify the interface, you must use the combination of the *link* and the *IPv6 address* (e.g., `fe80::12` on link 1).

If you look at link 2, you will also see there are two interfaces that have the same link-local address `fe80::1234`. This is illegal, as the two interfaces to which `fe80::1234` are in the same link-local scope zone, and an IPv6 address must be unique within its scope zone.

IPv6 address textual representation

- Addresses are represented as 8 bits of 4 hex digits (16 bits), separated by colons
`2001:0DB8:0:0:240:2BFF:FE3D:71AD`
- Two colons in a row can be used to denote one or more sets of zeroes, usually used between the prefix and the interface ID
`2001:0DB8::240:2BFF:FE3D:71AD`
- The prefix length can be indicated after a slash at the end
`2001:0DB8::240:2BFF:FE3D:71AD/64`
- A prefix alone is represented as if the interface ID bits are all zero
`2001:0DB8::/64`
- IPv4-Mapped IPv6 Address
`::FFFF:a.b.c.d`
- Obviously, this syntax may be a bit difficult for humans.....
Use of DNS/hostnames no longer an option

IPv6 addresses are represented by 8 sets of 4 hexadecimal digits, each separated by a colon character. The leading zeros in any given set of 4 digits may be omitted if desired.

Two colons may also be used to represent one or more sets of zeros in the IPv6 address. The use of the double-colon notation may appear at most once in any IPv6 address.

IPv4 addresses may be represented as an IPv6 address using a special notation known as an IPv4-mapped IPv6 address. This is typically written as "::" followed by FFFF: and then the dotted-decimal IPv4 address.

An IPv6 prefix is noted by including a "slash" character following the IPv6 address and then the number of bits which are included in the prefix. This notation may be used to define both an IPv6 address and a prefix in a single statement, or may be used to define just the IPv6 prefix.

Neighbor Discovery

- Router Discovery
 - ▶ Router Solicitations and Router Advertisements used to find and keep track of neighboring routers
 - ▶ Includes additional information for IP stack configuration
- Address resolution
 - ▶ Neighbor Solicitations and Neighbor Advertisements perform address resolution (i.e., ARP functions)
- Neighbor Unreachability Detection (NUD)
 - ▶ Keep track of reachability of neighbors
 - ▶ If path to router fails, switch to another router before TCP timeouts

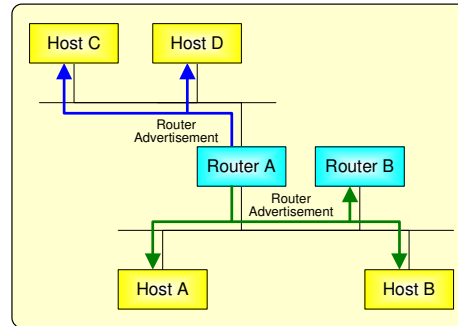
Neighbor Discovery is the name applied to a set of algorithms which are used by an IPv6 node to learn information about directly attached LANs. This includes configuration information, such as default routes, link MTU, and prefix information. The prefix information is used to autoconfigure addresses for the interface, using a process known as Stateless Address Autoconfiguration, as well as to determine which addresses may be reached via direct communication on the local LAN and which are off-link and, therefore, can only be reached via a router

Neighbor Discovery provides the ability to resolve an IP address to the link-layer address for neighbors on the LAN, the IPv6 equivalent of ARP. Neighbor Discovery is also used to verify that a node on a LAN is reachable, providing an architected version of the Dead Gateway Protocol included in some TCP/IP implementations.

Neighbor Discovery uses ICMP as the protocol for communication, allowing the Neighbor Discovery functions to be implemented independently of the underlying physical link

Stateless Address Autoconfiguration

- Address Configuration without separate DHCP server
 - ▶ Router is the server, advertising key address configuration information
- Address formed by combining routing prefix with Interface ID
- Link-local address configured when an interface is enabled
 - ▶ Allows immediate communication with devices on the local link
 - ▶ Primarily used for bootstrapping and discovery
 - ▶ Well-known prefix combined with locally-generated 64-bit IID
- Other addresses configured via Routing Advertisements
 - ▶ RA advertises 64-bit prefixes (e.g., on-link, form an address)
 - ▶ Public (e.g., server) addresses formed from Interface ID



IPv6 provides multiple ways for an IPv6 address to be learned by an IPv6 host. Using Stateless Address Autoconfiguration, a node can dynamically assign one or more addresses to the local interface. The network prefix (the upper 64 bits) is learned from routers on the LAN, and the interface ID (the lower 64 bits) is automatically generated from the MAC address of the LAN adapter. In addition to IP address assignment, various link characteristics, such as MTU, default routers, etc., can be learned from the routers attached to the LAN, eliminating the need to define these values at every host

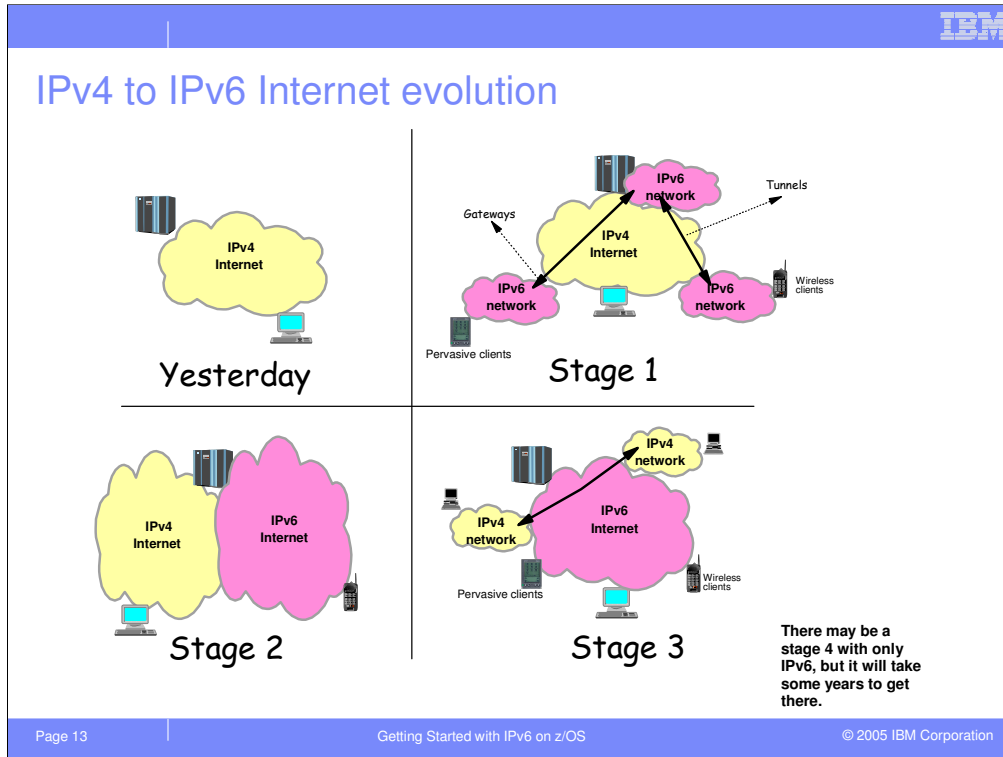
An alternative is to use DHCP to configure the host. A DHCP server can assign one or more IP addresses to an interface, similar to how DHCP works in an IPv4 environment. And, of course, you can manually assign IP addresses to an interface - the equivalent of static addresses.

In the example on this chart, Router A is originating two unique Router Advertisements, one onto the upper link and one onto the lower link. The Router Advertisement sent on upper link will inform all nodes on that link that:

- Router A is a router on the upper link
- Whether Router A is or is not to be used as a default router on the link
- Optionally, what prefixes exist on Link A and how they are to be used
- What internet parameters (MTU and hop-limit) are configured for the link.

Similarly, the Router Advertisement on lower link will inform all nodes on that link:

- Router A exists on Link B as a router
- Router A either is or is not to be used as a default router on the link
- Optionally, what prefixes exist on the link and how they are to be used
- What internet parameters (MTU and hop-limit) are configured for the link



The deployment of IPv6 into an existing IP network will normally be staged over time. Initially, small work groups will appear and begin using IPv6 to communicate among one another.

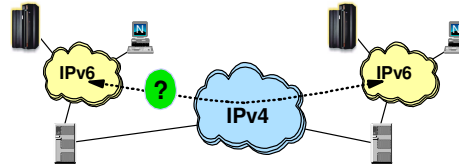
In time, additional IPv6 networks will begin to appear as small islands of IPv6 connectivity in a sea of IPv6. Eventually, individuals in these isolated islands will want to communicate with nodes in one of the other islands, or with devices in the IPv4 network. It is during this period of transition that most of migration issues will be encountered.

Over time, there will be parallel IPv4 and IPv6 networks running over the same physical network equipment – the same routers, hosts and links. Eventually, as the use of IPv4 recedes there will be a reversal of the initial IPv6 deployment, with islands of IPv4 in a sea of IPv6.

General transition considerations

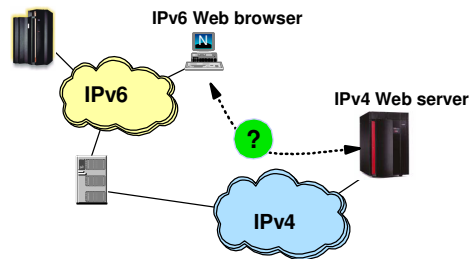
1 *How do we share the physical network so that both IPv4 and IPv6 can be transported over one and the same physical network?*

- Dual-stack
- Tunneling of IPv6 over IPv4



2 *How do applications that have not yet been enhanced to support IPv6 communicate with applications that have been enhanced to support IPv6?*

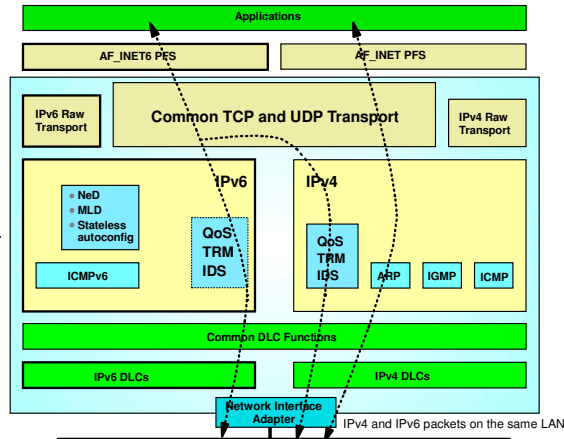
- Dual-stack
- Application Layer Gateways (ALG)
- Network Address Translation – Protocol Translation (NAT-PT)
- Bump-in-the-Stack (BIS) or Bump-in-the-API (BIA)



The IPv6 migration issues can be broken down into two main categories: how do IPv6 nodes communicate with one another when they do not have direct IPv6 connectivity, and how does an IPv4 application communicate with an IPv6 application? Both problems have several possible solutions.

Generalized dual-mode TCP/IP structure

- A dual-mode (or dual-stack) TCP/IP implementation supports both IPv4 and IPv6 interfaces - and both old AF_INET and new AF_INET6 applications.
- The dual-mode TCP/IP implementation is a key technology for IPv4 and IPv6 coexistence in an internet.
- For AF_INET6 applications, the common TCP or UDP transport layer determines per communication partner if the partner is an IPv4 or an IPv6 partner - and chooses IPv4 or IPv6 networking layer component based on that.
- Raw applications make the determination themselves when they choose IPv4 or IPv6 raw transport.



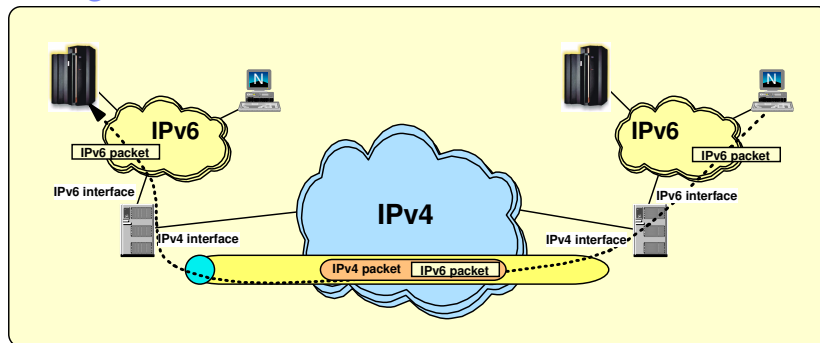
The basic building-block for IPv6 transition is the dual-mode, or dual-stack, TCP/IP node. A dual-mode node is a node which is able to send and receive packets using both an IPv4 network and an IPv6 network.

Existing applications which use AF_INET sockets may continue to run unmodified on a dual-mode node, but may only communicate with peers via the IPv4 network transport. In order for an application to communicate over the IPv6 network, though, the application must be modified to use AF_INET6 sockets. For TCP and UDP applications, a single AF_INET6 socket may be used to send packets via either the IPv4 or IPv6 network transport. The TCP or UDP transport selects the correct network transport protocol to use based on the destination IP address – IPv4 if an IPv4-mapped IPv6 address is used, and IPv6 otherwise.

Note that applications which use RAW sockets select the network transport to be used based on the address family of the socket which is created: IPv4 for an AF_INET socket and IPv6 for an AF_INET6 socket. Applications which use RAW sockets are inherently protocol aware, responsible for building the entire IPv4 packet, and much of the IPv6 packet. Fortunately, few applications outside of those shipped with an operating system, such as ping and traceroute, need to use RAW sockets.

IPv6-enabled applications running on a dual-mode node is the preferred migration path for existing applications and middleware, and the best way to implement any new applications. A single IPv6-enabled application is capable of communicating with both IPv4 and IPv6 partners, with the correct network transport protocol being chosen based on the network topology as well as the partner applications' capabilities.

Tunneling overview



- Tunneling: encapsulating an IPv6 packet in an IPv4 packet and send the IPv4 packet to the other tunnel endpoint IPv4 address.
- Requires applications on both endpoints to use AF_INET6 sockets
- Tunnels endpoints can be in hosts or routers
 - ▶ The tunnel endpoint may be an intermediate node, the final endpoint, or a mixture of the two
- The tunnel endpoint placement depends on connectivity needs
 - ▶ Placing endpoints in routers allows entire sites to be connected over an IPv4 network
 - ▶ Placing endpoints in hosts allows access to remote IPv6 networks without requiring updates to the routing infrastructure

In some cases, it may not be possible to use native communication between two hosts. For instance, if two nodes in isolated IPv6 islands need to communicate over an IPv4 sea, and either one or both of the nodes is not directly connected to the IPv4 network. Or the application protocol may be such that it requires the IPv6 network transport.

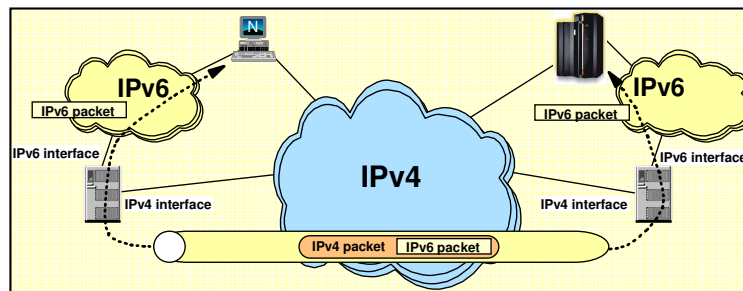
To allow two isolated IPv6 clouds to communicate over an IPv4 network, IPv6-over-IPv4 tunneling may be used. Tunneling refers to encapsulating the entire IPv6 packet inside an IPv4 packet and sending the IPv4 packet from the tunnel ingress to the tunnel egress. The placement of the tunnel endpoints can vary, and may be at the hosts, routers, or a mixture of the two. In general, it is better to place the tunnel endpoints as close to the IPv4 network backbone as possible.

There are many tunneling protocols which may be used: Configured Tunnels, 6to4, 6over4, ISATAP, and so on. Each implements the basics of tunneling the same way. When the IPv6 packet reaches the tunnel ingress, the tunnel ingress encapsulating the IPv6 packet inside an IPv4 packet and sends it over the IPv4 network transport to the tunnel egress. The tunnel egress, upon receiving the packet, decapsulates the packet and processes the IPv6 packet as though it was received over a native IPv6 interface. The IPv6 packet is then routed toward the final destination, either using the native IPv6 network transport or, possibly, another IPv6-over-IPv4 tunnel.

The use of an IPv6-over-IPv4 tunnel is transparent to applications and, if the tunnel endpoint is on a router in the network, to the IPv6 host on which the application is running.

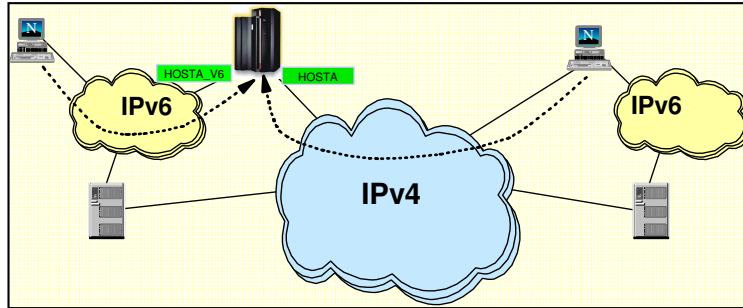
In order to use tunneling, both application endpoints must use the same network transport, IPv6. This, in turn, requires the applications to use AF_INET6 sockets.

IPv6 paths are preferred over IPv4



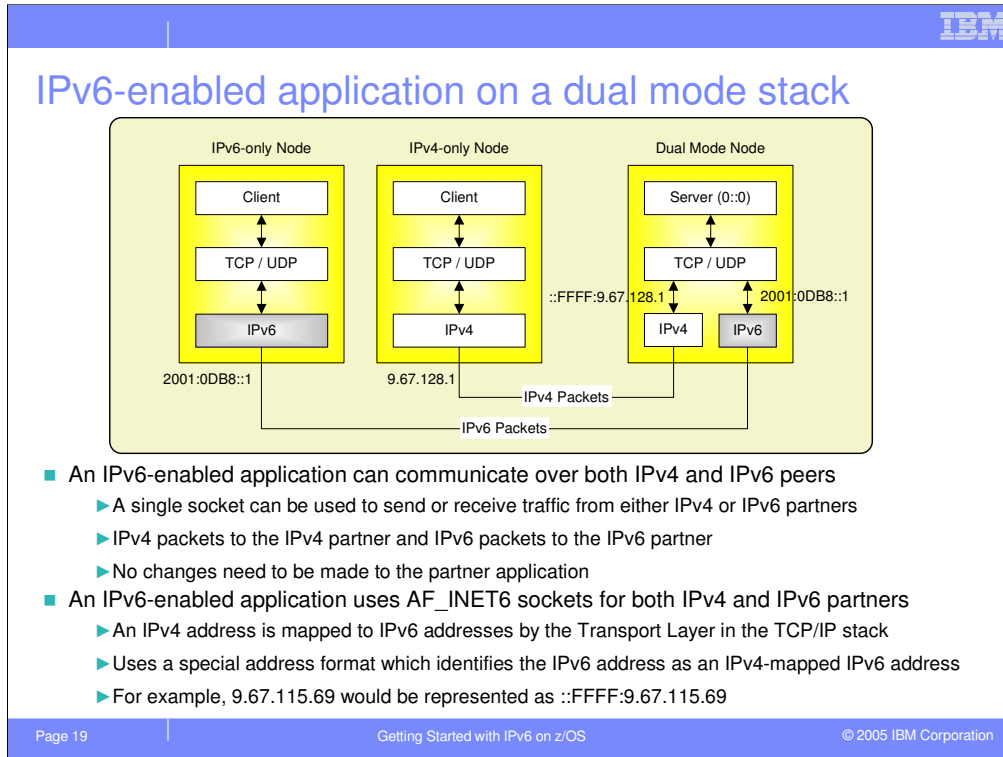
- IPv6 connectivity is preferred over IPv4
 - ▶ In many cases, only if one of the nodes does not support IPv6 will IPv4 be used
 - ▶ Can lead to undesirable paths in the network
 - Data may be tunneled over the IPv4 network even when a native IPv4 path exist
- May lead to longer connection establishment to an AF_INET application on a dual-stack node
 - ▶ IPv6 addresses will be tried before attempting to connect via IPv4
 - ▶ A "well behaved" client will cycle through all addresses returned and try the IPv4 address
 - But this takes time and network resources
 - And not all clients are "well behaved" or bug-free

Use of distinct IPv4 and IPv6 host names



- To avoid undesirable tunneling (and other potential problems), configure two host names in DNS
 - ▶ Continue to use the existing host name for IPv4 connectivity
 - ▶ Create a new host name to be used for IPv6 connectivity
 - ▶ Optionally, a third host name which may be used for both IPv4 and IPv6 can be configured
- Client chooses type of connection based on host name
 - ▶ Using the existing host name results in IPv4 connectivity
 - ▶ Using the new host name results in IPv6 connectivity

Note: Use of distinct host names is only necessary during the initial transition phases when native IPv6 connectivity does not exist



An IPv6-enabled server running on a dual-mode node which binds to the IPv6 wildcard address, `in6addr_any`, is able to accept connections from both IPv4 and IPv6 clients. IPv4 packets are sent and received when communicating with an IPv4 partner, and IPv6 packets are sent and received when communicating with an IPv6 partner. A single `AF_INET6` socket may be used for both IPv4 and IPv6 partners. In both cases, the application sees an IPv6 address for the partner: a native IPv6 address for IPv6 partners, and an IPv4-mapped IPv6 address for IPv4 partners.

Upgrading the server to support `AF_INET6` sockets is completely transparent to the IPv4 partner and requires no changes to the IPv4 partner. The partner continues to use `AF_INET` sockets and send and receive IPv4 packets.

Note that the changes to an IPv6-enabled client is similar to those for the IPv6-enabled server. The IPv6-enabled client may communicate with IPv4 and IPv6 servers, and no change is required at the IPv4 server when adding IPv6 support to the client.

IBM
z/OS

IPv4-only application on a dual-mode stack

- An IPv4 application running on a dual-mode stack can communicate with an IPv4 partner.
 - ▶ The source and destination addresses will be native IPv4 addresses
 - ▶ The packet which is sent will be an IPv4 packet
- If partner is IPv6 running on an IPv6 only stack, then communication fails
 - ▶ If partner was on dual-mode stack, then it would fit in previous page discussion
 - ▶ The partner only has a native IPv6 address, not an IPv4-mapped IPv6 address
 - ▶ The native IPv6 address for the partner cannot be converted into a form the AF_INET application will understand

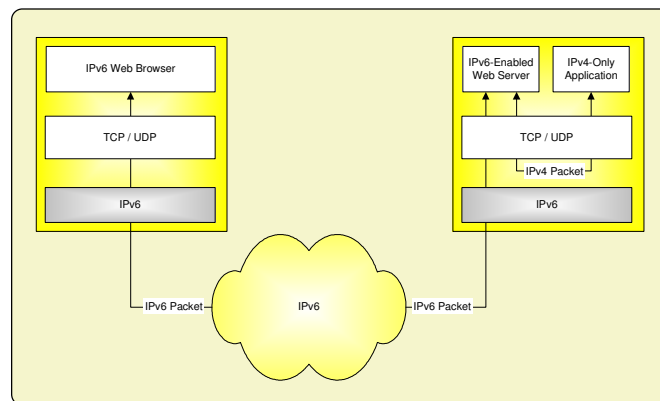
Page 20 | Getting Started with IPv6 on z/OS | © 2005 IBM Corporation

IPv4-only applications on a dual-mode node continue to run as-is. However, such applications are restricted to communicating only over the IPv4 network transport. IPv4 clients running on IPv4-only node or a dual-mode stack, or IPv6-enabled clients running on a dual-mode stack, may communicate with the IPv4-only server.

However, clients on an IPv6-only node cannot communicate directly with the IPv4-only server. The IPv6-only client is only capable of sending and receiving IPv6 packets, and the IPv4-only server is only capable of sending and receiving IPv4 packets. Since there is no common network transport protocol over which to transmit data, the two nodes cannot communicate directly.

Note that the same restrictions apply to an IPv4-only client which tries to communicate with an IPv6-only server.

Accessing IPv4-only applications through an IPv6 proxy



- An IPv6-only client can access IPv4-only servers via an IPv6 proxy
 - ▶ The IPv6 proxy communicates with the IPv6-only client using IPv6, and accesses the IPv4-only server using IPv4
 - ▶ The IPv4-only server may be on the same node as the IPv6 proxy, or may reside on a different node
 - ▶ The use of a backend IPv4-only server is, in most cases, completely transparent to the IPv6 client

One way for an IPv6-only client to access IPv4-only servers in the network is to use an IPv6 proxy. The IPv6 proxy establishes an IPv6 connection to the IPv6-only client, and also establishes an IPv4 connection to the IPv4-only server. When the client wishes to send data to the server, the client sends the data in an IPv6 packet. The proxy receives the data and forwards the data to the server as an IPv4 packet over the IPv4 network. Likewise, when the server wishes to send data to the client, the server sends the data in an IPv4 packet to the proxy, and the proxy sends the data to the client in an IPv6 packet over the IPv6 network.

Note that the IPv4-only application which the client wishes to access may reside on the same server as the proxy, or may reside on a different node which may be accessed using an IPv4 network transport.

z/OS V1R5 and V1R6 have been certified with the IPv6 Ready logo

Item	Content
Logo ID	01-000156
Vendor Name	IBM Corporation
Country Name	US
Product Name (Original)	z/OS
Product version (Original)	V1R5
Product Description (Original)	Highly secure scalable high-performance enterprise operating system
Product Name (Update)	
Product version (Update)	
Product Description (Update)	
Product Category	Host
Applied date	20031217
Application ID	US-20031217-000136
Current Status	Approved
Certificated Date	20040326

IPv6 Rollout on z/OS

IPv6 deployment phases

- The first phase (z/OS V1R4)

- ▶ Stack support for IPv6 base functions - (APIs, Protocol layers)
- ▶ Resolver
- ▶ High speed attach (OSA Express QDIO)
- ▶ Service tools (Trace, Dump, etc.)
- ▶ Configuration and netstat, ping, traceroute, SMF
- ▶ Static Routing
- ▶ FTP, otelnetd, unix rexec, unix rshd/rexecd

- The second phase (z/OS V1R5)

- ▶ Network Management
 - Applications and DPI
 - Version-neutral Tcp/Ip Standard MIBs
 - Additional SMF records
- ▶ Applications/Clients/APIs
 - Tn3270 server, CICS sockets, sendmail, ntp, dcas, rxserve, rsh client
- ▶ Enterprise Extender
- ▶ Point to Point - type DLCS
- ▶ Dynamic Routing Protocol w/ OMPROUTE (only RIPng)

- The third phase (z/OS V1R6)

- ▶ Sysplex Exploitation (Dynamic VIPA, Sysplex Distributor functions)
- ▶ Dynamic Routing Protocol w/ OMPROUTE (OSPFv3)
- ▶ Additional Network Management MIBs

- The fourth phase (z/OS V1R7)

- ▶ SNMP UDP standard MIB (RFC2013) and IBM MVS TCP/IP Enterprise-specific MIB for UDP
- ▶ Advanced Socket API support - RFC3542
- ▶ IPv6 Two Default Routers - required for IPv6 compliance
- ▶ HiperSockets DLC (Requires z9 HW - SW support in z/OS V1R7)

- After z/OS V1R7

- ▶ Integrated IPsec
- ▶ Complete Advanced Socket APIs
- ▶ Extended Stats MIB, OSPFv3 MIB
- ▶ Intrusion Detection Services
- ▶ IPv6 mobility support

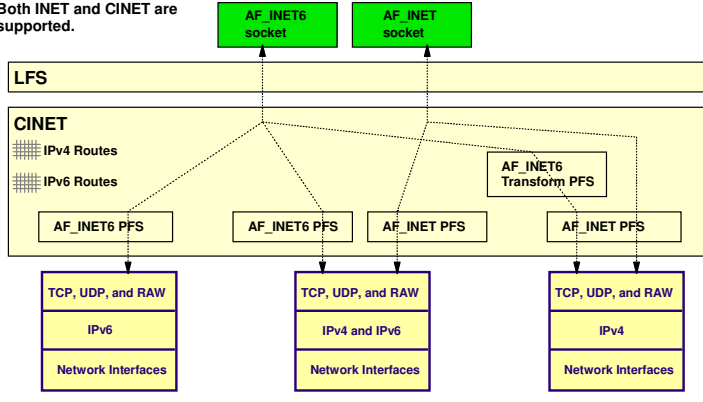
Objective is to have IPv6 production ready on the platform when you need it!

Enabling IPv6 support on z/OS

IPv6 is enabled at an LPAR level via an option in BPXPRMxx to enable AF_INET6 support. Both INET and CINET are supported.

When IPv6 is enabled, a z/OS V1R4 TCP/IP stack will always have an IPv6 Loopback interface. You can define real IPv6 interfaces in addition to the loopback interface.

- Existing AF_INET sockets programs will continue to work as they always did - no difference in behavior or support.
- AF_INET6 enabled sockets programs will be able to communicate with IPv4 partners (just as before they were changed to support IPv6), but in addition to that they will also be able to communicate with IPv6 partners.



IPv6-only TCP/IP Stack
This will not be the case on z/OS for the foreseeable future! If AF_INET6 is enabled, z/OS CS also requires AF_INET!

Dual Mode TCP/IP Stack
A z/OS V1R4 TCP/IP stack will always come up as dual-mode if AF_INET6 is enabled in BPXPRMxx

IPv4-only TCP/IP Stack
(such as AnyNet or an OEM TCP/IP stack)

When IPv6 is enabled, most netstat reports will look different because of the potential for long IPv6 addresses.

Make sure you have modified any netstat screen-scraping REXX programs you might have developed in the past!

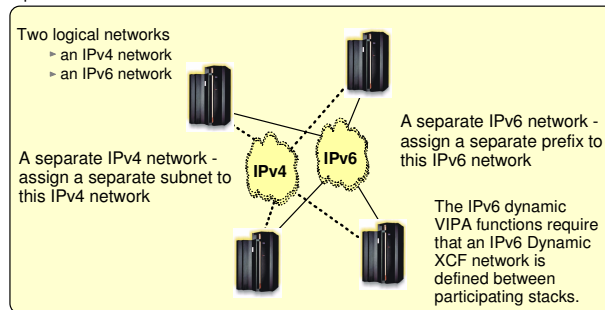
Configuring IPv6 support

- Basic IPv6 configuration is done using the IPCONFIG6 statement
 - ▶ Similar to IPCONFIG, which continues to be used for IPv4
 - ▶ Separate statements for IPv4 and IPv6 allow different values to be specified for IPv4 and IPv6
- Most of the defaults on the IPCONFIG6 statement are good choices
 - ▶ However, we recommend that you code the SOURCEVIPA parameter
 - SOURCEVIPA allows a VIPA to be used as the source IP address for which are established by this node
 - It also allows DNS address-to-name translation to work
 - It is not enabled by default, but is more important in an IPv6 environment
- You may want to enable IP forwarding using the DATAGRAMFWD parameter
 - ▶ The default is to **not** forward IP packets, the same as for IPv4

```
IPCONFIG6 SOURCEVIPA  
          DATAGRAMFWD
```

Connecting to an IPv6 network

- Uses separate logical networks - one IPv4 and one IPv6
 - ▶ The same physical adapter and network infrastructure can be used for both, though
- IPv6 DLC support in z/OS V1R4
 - ▶ Fast Ethernet and Gigabit Ethernet using OSA Express in QDIO Mode
- Additional IPv6 DLC support in z/OS V1R5
 - ▶ IUTSAMEHOST to other stacks in same LPAR
 - ▶ XCF to other stacks in same Sysplex
 - Both static and dynamic XCF
 - ▶ ESCON (MPCPTP) to another z/OS image (not to any known Channel-attached Routers)
- IPv6 HiperSockets support
 - ▶ Requires z9 processor and z/OS V1R7



Defining IPv6 interfaces

- IPv6 interfaces are defined using an INTERFACE statement in the TCP/IP profile
 - ▶ Combines the definitions of DEVICE, LINK and HOME into one statement
 - ▶ In order for one physical device to support both IPv4 and IPv6 traffic, DEVICE, LINK and HOME statements have to be specified in the profile to define the IPv4 side and an INTERFACE statement must be specified to define the IPv6 side
- A single IPv6 interface may have one or more IPv6 addresses at any given time
 - ▶ There will always be a link-local address, which is automatically assigned during interface activation
 - ▶ There may be 0-n site-local and/or global IPv6 addresses as well
- For physical interfaces, IP addresses (except for the link-local address) may be manually configured or may be autoconfigured

```
INTERFACE OSAQDIO15 DEFINE IPAQENET6 PORTNAME OSAQDIO1  
  
INTERFACE OSAQDIO25 DEFINE IPAQENET6 PORTNAME OSAQDIO2  
IPADDR FEC0::9:67:115:5  
2001:0DB8::9:67:115:5
```

IPv6 VIPA and SOURCEVIPA

- Static VIPAs are defined on a VIRTUAL6 interface
 - ▶ Each VIRTUAL6 interface must be manually configured with one or more IPv6 addresses
 - ▶ The TCP/IP stack will choose the "best" address as the source IP address using the Default Address Selection algorithms defined by the IETF
- Use the SOURCEVIPAINterface parameter to associate a physical interface to a specific VIRTUAL6 interface
 - ▶ No ordering considerations like DEVICE/LINK/HOME for IPv4
 - ▶ More than one physical interface can point to the same VIRTUAL6 interface
- IPCONFIG6 SOURCEVIPA definition makes the SOURCEVIPA function available for all IPv6 interfaces configured with SOURCEVIPAINterface.

```
IPCONFIG6 SOURCEVIPA

INTERFACE VIPAV61 DEFINE VIRTUAL6
IPADDR FEC0::9:67:115:5 2001:0DB8::9:67:115:5

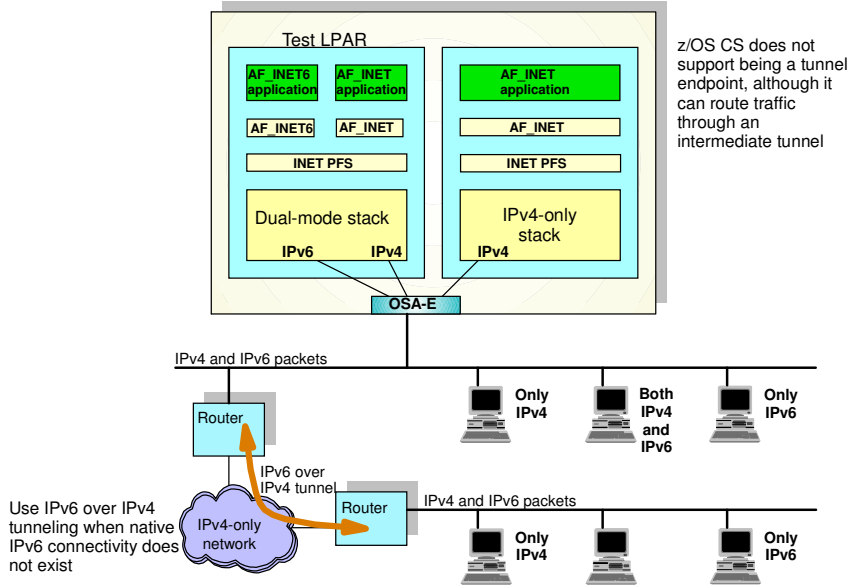
INTERFACE VIPAV62 DEFINE VIRTUAL6
IPADDR FEC0::9:67:115:6 2001:0DB8::9:67:115:6

INTERFACE OSAQDIO16 DEFINE IPAQENET6 PORTNAME OSAQDIO1
SOURCEVIPAIN T VIPAV61

INTERFACE OSAQDIO26 DEFINE IPAQENET6 PORTNAME OSAQDIO2
SOURCEVIPAIN T VIPAV62

INTERFACE OSAQDIO36 DEFINE IPAQENET6 PORTNAME OSAQDIO3
```

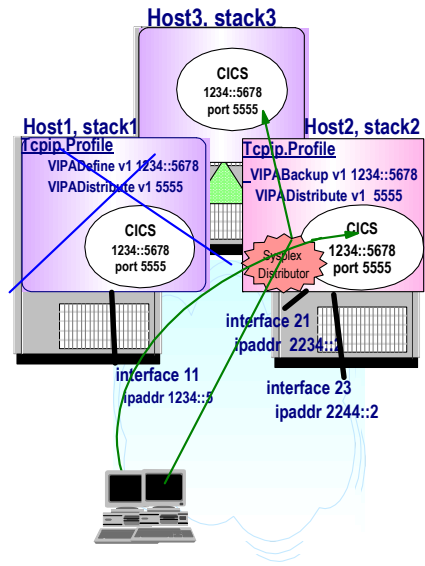
Accessing z/OS from a remote site



z/OS CS does not support being a tunnel endpoint, although it can route traffic through an intermediate tunnel

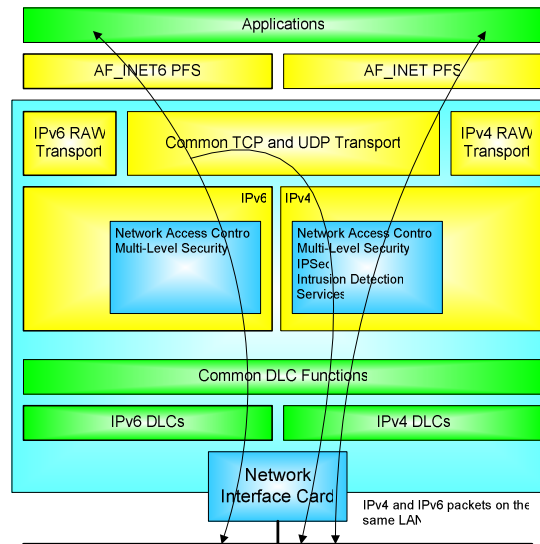
Sysplex functions that support IPv6

- Almost all Sysplex functions support IPv6 as of z/OS V1R6
 - ▶ Dynamic VIPA (DVIPA)
 - ▶ Dynamic VIPA Takeover
 - ▶ Sysplex Distributor
 - ▶ Sysplex Sockets
 - ▶ TCPSTACKSOURCEVIPA
 - ▶ Sysplexports
 - ▶ Fast Connection Reset after System Failure
 - ▶ Enhance Workload Distribution (Application Server Affinity)
 - ▶ Dynamically Assign Sysplex Ports
 - ▶ Activation of DVIPAs through VIPABACKUP
 - ▶ DYNAMICXCF SOURCEVIPAINIT
 - ▶ Sysplex Distributor Round-Robin Distribution
 - ▶ Sysplex Distributor Policy
- A few Sysplex functions are not enabled for IPv6
 - ▶ Sysplex Wide Security Associations (SWSA)
 - IPsec is not yet supported for IPv6
 - ▶ Multi Node Load Balancing (MNLB)
 - Cisco does not support IPv6 for MNLB



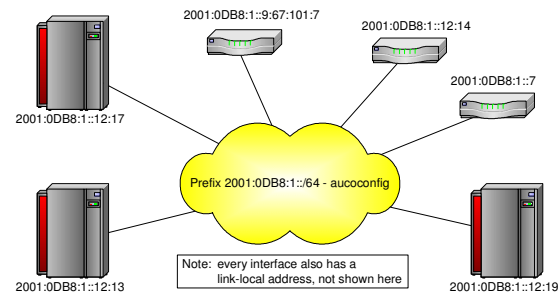
Securing your IPv6 network

- z/OS V1R5 provides the first set of security features for the IPv6 transport
 - ▶ Includes support for both Network Access Control and Multi-Level Security
- IPsec and IDS continue to be supported for the IPv4 transport
 - ▶ Existing IPv4 applications will continue to use both without any impact
 - ▶ IPsec and IDS will be used for IPv6-enabled applications when sending over the IPv4 transport, but not when sending over the IPv6 transport



Dynamic routing

- Support IPv6 RIP (RIPng) in V1R5 and IPv6 OSPF (OSPFv3) in V1R6
 - ▶ Implementation done in OMPROUTE
 - One and the same daemon for both IPv4 (RIP and OSPF) and IPv6 (RIPng and OSPFv3)
- Based on IPv4 specifications with IPv6-specific updates
 - ▶ IPv6 RIP includes minimal changes
 - Replacement for RIPv1 and RIPv2 used in IPv4 networks
 - ▶ IPv6 OSPF is protocol independent
 - Separate IP addressing and network topology where possible
 - It could be used for network protocols other than IPv6, although it isn't today
- Needed for Sysplex-related functions such as dynamic IPv6 VIPA movement



DNS in three easy steps

1. Add and modify statements in the nameserver configuration file
 - ▶ New reverse zone statements
 - ▶ IPv6-specific options (optional)
 - ▶ IPv6 information for options which can take IPv4 or IPv6 addresses (optional)
2. Add IPv6 records to forward zones with hosts that are now IPv6-capable
 - ▶ IPv6 address records: AAAA
3. Create new IPv6 reverse zone files
 - ▶ IPv6 reverse domains: ip6.arpa and ip6.int
 - ▶ Use the same PTR records from IPv4, with a similar label format

Recommendations when adding IPv6 addresses to DNS

- Add Static VIPAs in DNS
 - ▶ You don't need to add addresses assigned to physical interfaces if using VIPA and SOURCEVIPAs
 - ▶ z/OS autoconfigured addresses are not suitable for placement in DNS
 - May (and likely will) change each time a z/OS stack is recycled
 - If you need to place addresses assigned to physical interfaces in DNS, then you should manually configure the addresses
- Configure two (and optionally three) host names in DNS
 - ▶ Continue to use the existing host name for IPv4 connectivity
 - ▶ Create a new host name to be used for IPv6 and IPv4 connectivity
 - ▶ Optionally, a third host name which may be used only for IPv6 can be configured
- Be careful when adding site-local addresses to DNS
 - ▶ Site-local addresses are not globally unique and must not be returned to hosts outside the local site
 - ▶ You need to use split-DNS (sometimes called two-faced DNS) if you use site-local addresses
 - Similar to how private addresses are handled in IPv4
 - ▶ Site-local addresses have been deprecated by the IETF
 - New emerging standard, "Unique Local IPv6 Unicast Addresses", will be globally unique but still targeted for internal network use
- **Never** add link-local addresses to DNS
 - ▶ They can't be used beyond the link on which they are defined, and aren't intended for general-purpose applications

Resolver communication with DNS Name Server

- The Resolver sends queries to DNS server using IPv4
 - ▶ The IPv4 protocol is used to communicate, and does not affect what type of records are returned
 - You can still resolve host names to IPv6 addresses and vice-versa
 - ▶ You can use a local DNS name server (caching only or authoritative) if there is no IPv4 network connectivity, as the DNS name server is able to send queries via IPv4 or IPv6
- Resolver communication with DNS name servers
 - ▶ Name query sends AAAA query to DNS and receives AAAA records in response
 - ▶ Reverse query sends PTR query to the 'ip6.arpa' domain and receives results from the 'ip6.arpa' domain
- The results of Resolver queries varies based on interface availability
 - ▶ Resolver may omit IPv4 or IPv6 results if there aren't any physical interfaces which support the network protocol
 - The behavior is determined by the invoking application
 - ▶ Resolver sorts the addresses returned based on local interface availability
 - Default Address Selection algorithms govern both source address selection and destination address selection
 - Destination Address Selection is performed by Resolver as part of the name-to-address mapping
 - Source Address Selection is performed by the TCP/IP stack after the destination address is chosen
- May want to consider using a local host file for early testing
 - ▶ using the *LOOKUP LOCAL/DNS* resolver directive

Updating the local host file - Useful for early testing

- Many platforms (z/OS, Solaris, Linux, ...) use /etc/ipnodes as the local host file for IPv6 name queries
 - ▶ Local database that associates host names with IP addresses
 - ▶ May be used to store both IPv4 and IPv6 addresses (using the COMMONSEARCH System Resolver option)
 - ▶ Extended version of /etc/hosts
 - Uses the same format as /etc/hosts, but may be used to store both IPv4 and IPv6 addresses
 - ▶ Other platforms may use a different file for this purpose
- /etc/hosts may continue to be used to store IPv4 addresses
 - ▶ But may not be used to store IPv6 addresses (same is true for files created with MAKESITE utility - HOSTS.SITEINFO and HOSTS.ADDRINFO)

```
9.67.43.100    NAMESERVER
9.67.43.126    RALEIGH
9.67.43.222    HOSTNAME1      HOSTNAME1_IPV4
129.34.128.245 YORKTOWN      WATSON
1::2          HOSTNAME1      HOSTNAME1_IPV6
1:2:3:4:5:6:7:8 HOSTNAME2_IPV6
```

Instead of or in addition to making changes to DNS, you can also use a local host file to map between a host name and IP addresses and vice-versa. On most platforms, the /etc/ipnodes may be used to associate a host name with IPv4 and/or IPv6 addresses. /etc/ipnodes is an extended version of /etc/hosts and uses the same format as /etc/hosts, but may be used to store both IPv4 and IPv6 addresses. /etc/hosts may continue to be used to store IPv4 addresses, but may not be used to store IPv6 addresses.

Note that not all platforms use /etc/ipnodes for the local host file. You will want to read through the configuration references for each operating system on which you are testing to determine the exact configuration changes you need to make.

Applications enabled for IPv6

- IPv6-enabled applications in z/OS V1R4
 - ▶ inetd
 - ▶ ftp and ftpd
 - ▶ telnetd
 - ▶ USS rshd and rexecd servers
 - ▶ USS rexec client
 - ▶ ping
 - ▶ tracert
 - ▶ netstat
- Additional applications IPv6-enabled in z/OS V1R5
 - ▶ tftpd (trivial file transfer server)
 - ▶ syslogd
 - ▶ dcas (digital certificate access server)
 - ▶ sntpd (simple network time protocol server)
 - ▶ sendmail 8.12.x (new port of sendmail picks up IPv6 enablement too)
 - ▶ MVS rshd/rexecd server
 - ▶ TSO rsh/rexec clients
 - Updated version that can be used in all z/OS environments (batch, TSO, REXX, etc.)
 - ▶ New UNIX rsh client that is IPv6-enabled from start
 - ▶ CICS Listener (including CICS socket APIs)

FTP

■ FTP server

- ▶ To enable IPv6 support in the FTP server, activate IPv6 stack support
 - No new configuration commands are provided or needed to enable IPv6 support
- ▶ User Exit routines
 - Update server exit routines for IPv6 addressing
- ▶ Trace and Extended Trace
 - Update DUMP IPADDR() and DEBUG IPADDR() as needed
- ▶ NETRC data set
 - Update with IPv6 addresses as needed
- ▶ SMF recording
 - Update SMF statements in client and server FTP.DATA commands

■ FTP client

- ▶ For the client, you may specify the host as an IPv4 address, a hostname, an IPv4 mapped IPv6 address, or as an IPv6 address
 - Examples:
 - `ftp fec0:197:11:105::1`
 - `ftp 9.67.21.33` and `ftp ::ffff:9.67.21.33` are equivalent
 - `ftp linuxipv6.tcp.raleigh.ibm.com`

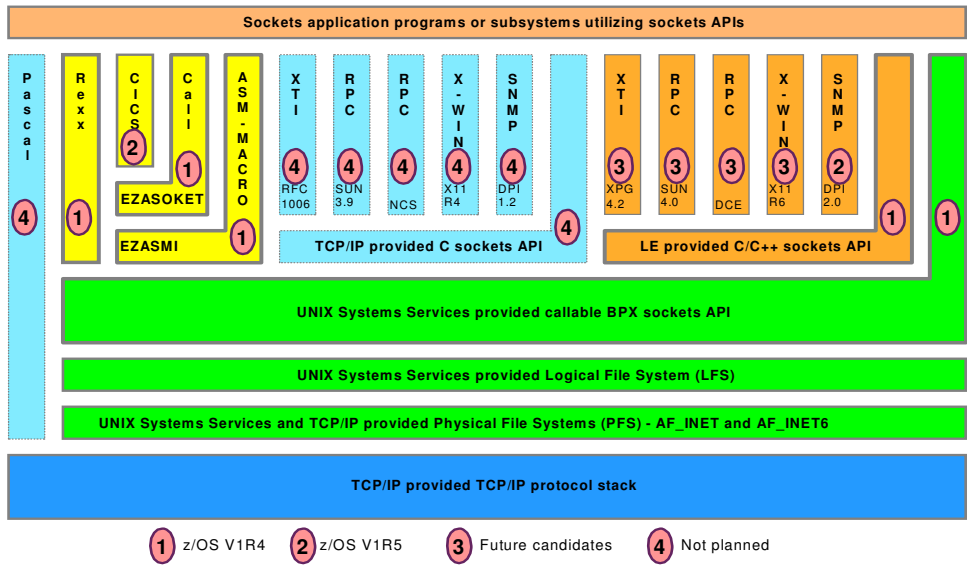
TN3270

- To enable IPv6 support in the TN3270 server, activate IPv6 stack support
 - ▶ No new configuration commands are provided or needed to enable IPv6 support
- IPv6-enable the TN3270 server
 - ▶ Support clients with IPv6 addresses
 - ▶ Support IPv6 addresses in USS messages, displays, command responses, etc.
 - ▶ Support IPv6 addresses as client identifiers for all mapping statements in TN3270 server configuration that allows an IP address client identifier
 - ▶ Includes SSL/TLS support
 - ▶ Changes made to VTAM to support TN3270 visibility when clients are IPv6 clients
 - IPv6 addresses are passed to VTAM
 - VTAM displays that include IP addresses are enhanced to accommodate IPv6 addresses

Enterprise Extender

- Enterprise Extender adds support for IPv6 in z/OS V1R5
 - ▶ Allows Enterprise Extender to exploit an IPv6-enabled network
 - ▶ Architectural changes needed since HPR passes IP addresses in protocol data and is supported on multiple platforms
 - ▶ Changes to VTAM exits to pass IPv6 addresses, hostnames, and port numbers:
 - SME (Session Management exit)
 - Login exit
- IPv6 support requires use of the HOSTNAME keyword (start option, GROUP, path definition)
 - ▶ Existing IPADDR keywords (start option, path definition in SMN) are IPv4-only
- EE Connection networks are IPv4-only or IPv6-only
 - ▶ Nodes supporting both IPv4 and IPv6 must define an IPv4 VRN (local and/or global) and an IPv6 VRN

Sockets-related AF_INET6 enablement



Sockets API considerations when moving to AF_INET6 (Notes)

- IPv6 addresses are 128-bit in size as compared to 32 bits for IPv4
 - ▶ Data structures which store IP addresses must be modified to handle the larger size
- DNS Resolver library changes
 - ▶ New DNS calls replace `gethostbyname()` and `gethostbyaddr()`
- Textual representation of the IP address has changed
 - ▶ IPv4 addresses use dotted-decimal format
 - ▶ IPv6 addresses use colon-hex notation
- IPv6 has several scopes for IP addresses
 - ▶ An address is only unique within its given scope
 - ▶ On multihomed hosts, an IP address alone may be insufficient to select the interface over which to route
 - True for link-local addresses
 - Most applications will not care about this, but it is possible that some may
- IP addresses should not be assumed to be permanent
 - ▶ Long-term use of an address is discouraged due to renumbering
 - ▶ Applications should rely on DNS resolvers to cache the appropriate IP addresses
- `sockaddr_in6`
 - ▶ analogous to `sockaddr_in`, but larger
 - ▶ Holds 128-bit IPv6 address, port numbers, plus Flow Label and Interface Identifier
 - ▶ Two versions of `sockaddr_in6` are available, converging the 4.3 and 4.4 BSD variants.
- `in6_addr`
 - ▶ analogous to 32-bit `in_addr`
 - ▶ holds a 128 bit address
- Socket calls to investigate for possible changes
 - ▶ `socket()`, `bind()`, `connect()`, `sendmsg()`, `sendto()`, `accept()`, `recvfrom()`, `recvmsg()`, `getpeername()`, `getsockname()`
- New calls
 - ▶ `inet_pton()`, `inet_ntop()`

Managing your IPv6 network

- Network management SNMP support
 - ▶ Support SNMP agent (OSNMPD)
 - ▶ DPI 2.0 enabled for AF_INET6 (used between SNMP subagents and SNMP manager)
 - ▶ Support TCPIP (stack) subagent
 - ▶ osnmp command
 - ▶ The trap forwarder daemon enabled for AF_INET6
 - ▶ IPv6 MIB support (as many as we can squeeze into z/OS V1R5!)
 - New RFCs have been published that are IP version neutral - support will gradually converge from supporting version-specific MIBs to the new version-neutral MIBs
 - RFC 2011 (IP and ICMP)
 - RFC 2012 (TCP)
 - RFC 2096 (IP routes)
 - RFC 2333 (Interfaces) - this one is not version neutral

Netstat impacts

- Almost all of the Netstat reports have to be re-designed or re-formatted
 - ▶ The required maximum IPv6 address length is 45 characters long
 - ▶ A Netstat report can be displayed from three environments (TSO, UNIX shell and MVS console), and from MVS console, the maximum characters that can be displayed in a line is 71 characters
 - Because of these limitations, information such as a connection entry that used to be displayed in one line has to take more lines now
 - If the TCP/IP stack is IPv6-enabled, then most of the reports have a different format than before. If the TCP/IP stack is not IPv6-enabled, then the report format is the same as in previous releases, unless the new FORMAT LONG parameter has been specified on the Netstat command or on the IPCONFIG profile statement
- Guideline for Netstat Report Changes
 - ▶ Where a report entry occupies more lines and may have more sub-sections, a consistent syntax of indenting line two bytes, and for each new subsection, another two bytes (always in increments of two bytes). Some table displays use more than two bytes for the second line of a table entry for readability
 - ▶ Since the IPv6 textual address can be quite long, we try consistently to always have the IPv6 address as the last object on a line
 - ▶ Message IDs for command responses are no longer supported under TSO NETSTAT except for error messages

Netstat HOME/-h

```

MVS TCP/IP NETSTAT CS V1R4      TCPIP Name: TCPCS      15:49:35
Home address list:
LinkName: OSAQDIOLINK
Address: 9.67.115.5
Flags: Primary
LinkName: LOOPBACK
Address: 127.0.0.1
Flags:
IntfName: VIPAV6
Address: fec0::a:9:67:115:5
Type: Site_Local
Flags:
Address: 50c9:c2d4:0:a:9:67:115:5
Type: Global
Flags: Deprecated
IntfName: OSAQDI046
Address: fec0::9:67:115:5
Type: Site_Local
Flags:
Address: fe80::6:2900:20dc:217c
Type: Link_Local
Flags: Autoconfigured
IntfName: LOOPBACK6
Address: ::1
Type: Loopback
Flags:
Unavailable IPv6 Home addresses:
IntfName: OSAQDI026
Address: fec0::9:67:115:66
Type: Site_Local
Reason: Duplicate address detection pending start of interface
IntfName: OSAQDI066
Address: fec0::/64
Type: Site_Local
Reason: Interface ID not yet known
  
```

Netstat DEVLINKS/-d

```

MVS TCP/IP onetstat CS V1R5          TCPIP Name: TCPCS          12:55:20
DevName: OSAQDI04                   DevType: MPCIPA
DevStatus: Ready
LnkName: OSAQDIOLINK                 LnkType: IPAQENET   LnkStatus: Ready
NetNum: 0   QueSize: 0   Speed: 0000000100
IpBroadcastCapability: No
CfgRouter: Non                     ActRouter: Non
ArpOffload: Yes                      ArpOffloadInfo: Yes
ActMtu: 1492
VLANid: 1260                          VLANpriority: Enabled
ReadStorage: GLOBAL (8064K)          InbPerf: Balanced
ChecksumOffload: Yes
BSD Routing Parameters:
MTU Size: 00000                      Metric: 00
DestAddr: 0.0.0.0                   SubnetMask: 255.255.255.192
Multicast Specific:
Multicast Capability: Yes
Group      RefCnt
-----
224.0.0.1      0000000001
Link Statistics:
BytesIn      = 11476
Inbound Packets      = 10
Inbound Packets In Error      = 0
Inbound Packets Discarded      = 0
Inbound Packets With No Protocol = 0
BytesOut     = 6707
Outbound Packets      = 10
Outbound Packets In Error      = 0
Outbound Packets Discarded      = 0

```

Netstat DEVLINKS/-d (continued)

```

IntfName: OSAQDIO46      IntfType: IPAQENET6 IntfStatus: Ready
NetNum: 0  QueSize: 0  Speed: 0000000100
MacAddress: 000629DC21BC
SrcVipIntf: VIPAV6
DupAddrDet: 1
CfgRouter: Pri          ActRouter: Pri
RtrHopLimit: 5
CfgMtu: 4096            ActMtu: 1492
VLANid: 1261           VLANpriority: Enabled
IntfID: 0000:0000:0000:0001
ReadStorage: GLOBAL (8064K)  InbPerf: Balanced
Packet Trace Setting:
Protocol: *             TrRecCnt: 00000000  PckLength: FULL
SrcPort: *             DestPort: *
IpAddr/PrefixLen: 9::44/128
Multicast Specific:
Multicast Capability: Yes
RefCnt  Group
-----  -----
0000000001 ff02::1:ff15:5
0000000001 ff02::1:ff00:2
Interface Statistics:
BytesIn           = 12655
Inbound Packets   = 12
Inbound Packets In Error = 0
Inbound Packets Discarded = 0
Inbound Packets With No Protocol = 0
BytesOut          = 4590
Outbound Packets  = 11
Outbound Packets In Error = 0
Outbound Packets Discarded = 0

```

Testing network connectivity

- Ping and Traceroute support for IPv6
 - ▶ IPv6 IP addresses, or host names that resolve to IPv6 IP addresses, can be used for destinations
 - ▶ IPv6 IP addresses can be used as the source IP address for the command's outbound packets
 - ▶ IPv6 IP addresses or interface names can be used as the outbound interface
 - ▶ A new ADDRTYPE/-A command option can be specified to indicate whether an IPv4 or IPv6 IP address should be returned from host name resolution
- IPv4-mapped IPv6 IP addresses are not supported for any option value
- The TSO PING and TRACERTE commands have been rewritten and now have the same behavior as the z/OS UNIX version of the commands
 - ▶ Message ID numbers will no longer be associated with the TSO command output
 - ▶ Message ID numbers will still be displayed for informational and error messages if the TSO PROFILE MSGID option is in effect
 - ▶ All of the old messages have been replaced either with existing messages also used by the z/OS UNIX versions of the commands, or by new messages
 - ▶ In a multi-stack environment, the TSO commands will continue to associate themselves either with the default stack or the stack specified on the stack name option, TCP

Steps for moving to an IPv6 Environment

1. Network access

- ▶ A LAN can carry both IPv4 and IPv6 packets over the same media
- ▶ An OSA-E port can be used for both IPv4 and IPv6
- ▶ Update TCP/IP Profile to include the INTERFACE statement(s) for any IPv6 interfaces
- ▶ For LPAR-LPAR communication for IPv6, several options exist:
 - using QDIO to a shared LAN (or a Shared OSA)
 - MPCPTP6 interfaces (via XCF if on the same sysplex or ESCON CTC links)
 - IPv6 HiperSocket connections (if on the same CEC) – Requires z9 hw and z/OS V1R7

2. IPv6 address selection

- ▶ Obtain an address block from your ISP, or use one of your IPv4 addresses to create a 6to4 prefix
- ▶ For test purposes, site-local IPv6 addresses is sufficient, but avoid using them in production
 - Should look into using emerging “Unique Local IPv6 Unicast Addresses” instead of site-local
- ▶ IPv6 addresses can be assigned to the IPv6 Interfaces and static VIPAs
- ▶ Addresses can be manually configured on the INTERFACE statement in the TCP/IP Profile or autoconfigured using Neighbor Discovery Stateless Autoconfiguration (VIPAs addresses must be manually configured)

Steps for moving to an IPv6 Environment

3. DNS setup

- ▶ DNS BIND 9 Name Server can be used for both IPv4 and IPv6 resources
- ▶ Continue to use the existing host name for IPv4 connectivity to avoid possible disruption in network connectivity and IPv4-only applications on an IPv6-enabled stack
- ▶ Create a new host name to be used for IPv6 and IPv4 connectivity
- ▶ Optionally, a third host name which may be used only for IPv6 can be configured
- ▶ If using stateless autoconfiguration to define IPv6 addresses, static VIPA addresses should be stored in DNS since the autoconfigured addresses will change over time and no Dynamic DNS support is available on z/OS

4. INET or Common INET

- ▶ Both are supported for IPv6, but INET is much simpler
- ▶ Running IPv4 and dual-mode stacks under CINET is not recommended - run dual-mode stacks in a separate LPAR from IPv4 only stacks
- ▶ AF_INET6 NETWORK statement must be coded in BPXPRMxx before starting IPv6-enabled stacks

Steps for moving to an IPv6 Environment

5. Selection and placement of IPv6 to IPv4 protocol converter or application gateway
 - ▶ z/OS does not implement any functions that will allow IPv6-only nodes to communicate with z/OS-resident AF_INET applications, so an outboard protocol converter or application-layer gateway component may be needed
 - ▶ This component will only be needed if the test configuration includes IPv6-only platforms
 - ▶ Various technologies are being made available by various vendors; SOCKS64 seems the simplest technology right now
6. Connectivity to non-local IPv6 locations
 - ▶ Tunneling may be needed between a router connected to the LAN that z/OS is connected to, and a router at another location where IPv6 test equipment is located

z/OS Communications Server Books

z/OS Communications Server V1R4:	Book Number	Softcopy Book Name
IP Migration	GC31-8773-02	f1a1b120
SNA Migration	GC31-8774-02	f1a1b220
IPv6 Network and Application Design Guide	SC31-8885-00	f1a1f100
IP Configuration Guide	SC31-8775-02	f1a1b320
IP Configuration Reference	SC31-8776-02	f1a1b420
SNA Network Implementation	SC31-8777-02	f1a1b520
SNA Resource Definition	SC31-8778-02	f1a1b620
SNA Operation	SC31-8779-02	f1a1b720
Quick Reference	SX75-0124-02	f1a1b820
IP User's Guide and Commands	SC31-8780-02	f1a1b920
IP System Administrator's Commands	SC31-8781-01	f1a1c210
SNA Diagnosis, Vol 1; Techniques and Procedures	LY43-0088-02	f1a1c320
SNA Diagnosis, Vol 2; FFST Dumps and the VIT	LY43-0089-02	f1a1c420
IP Diagnosis	GC31-8782-02	f1a1c520
SNA Messages	SC31-8790-02	f1a1c620
IP Messages: Volume 1 (EZA)	SC31-8783-02	f1a1c720
IP Messages: Volume 2 (EZB)	SC31-8784-02	f1a1c820
IP Messages: Volume 3 (EZY)	SC31-8785-02	f1a1c920
IP Messages: Volume 4 (EZZ-SNM)	SC31-8786-02	f1a1d120
IP and SNA Codes	SC31-8791-02	f1a1d220
IP API Guide	SC31-8788-02	f1a1d420
IP Programmer's Reference	SC31-8787-02	f1a1d320
IP CICS Sockets Guide	SC31-8807-01	f1a1g110
SNA Data Areas, 1	LY43-0090-02	f1a1d520
SNA Data Areas, 2	LY43-0091-02	f1a1d620

For more information...

URL	Content
http://www.ibm.com/software/ipv6	IBM's IPv6 web page
http://www.ibm.com/servers/eserver/zseries	IBM eServer zSeries Mainframe Servers
http://www.ibm.com/servers/eserver/zseries/networking	Networking: IBM zSeries Servers
http://www.ibm.com/servers/eserver/zseries/networking/technology.html	IBM Enterprise Servers: Networking Technologies
http://www.ibm.com/software/network	Networking & Communications Software
http://www.ibm.com/software/network/commserver	Communications Server
http://www.ibm.com/software/network/commserver/library	CS White Papers, Product Doc, etc.
http://www.redbooks.ibm.com	IBM Redbooks
http://www.ibm.com/software/network/commserver/support	Communications Server Technical Support
http://www.ibm.com/support/techdocs/	Advanced Technical Support (Flashes, Presentations, White Papers, etc.)

Supplemental DNS Configuration

named.conf changes for IPv6

- Additional zone { } statements:
 - ▶ New zone statements are required for IPv6 reverse zones.
- Additions to the options { } statement:
 - ▶ listen-on-v6 { any; };
 - ▶ named can listen on ALL IPv6 interfaces or NONE of them. Hence, the only allowable values for listen-on-v6 { } are 'any;' or 'none;'
- Other (optional) options:
 - ▶ transfer-source-v6
 - ▶ query-source-v6
 - ▶ notify-source-v6
- Other statements not specific to v4 or v6 take either IPv4 or IPv6 addresses as arguments.
 - ▶ e.g., the forwarders { } or allow-transfer { } options, or the masters clause in a slave zone statement.

Example IPv6-enabled name server configuration file

```
acl mynets { fec0::/64; fec0:0:0:A::/64;          #IPv6 site-local
              50c9:c2d4::/64; 50c9:c2d4:0:A/64;    #IPv6 global
              9.67.115.0/26; };                   #IPv4 subnet

options {
directory "/etc/dnsdata";
pid-file "/etc/dnsdata/named.pid";
listen-on-v6 { any; };
query-source-v6 address 50c9:c2d4::A:9:67:115:5 port *;
allow-transfer { mynets; };
};

zone "tcp.raleigh.ibm.com" {
type slave;
masters { fec0::9:67:114:45; 9.67.114.45; };
file "db.tcp.slave";
};
zone "A.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.ip6.int" {
type master;
file "db.ipv6.reverse";
};
```


Forward zone changes for IPv6 DNS

- IPv4 information is stored in A records; IPv6 information is stored in AAAA records.
 - ▶ The format is essentially the same, e.g.,
 - `www IN A 9.67.115.5`
 - `www IN AAAA fec0::9:67:115:5`
 - ▶ The new IPv6 records may coexist with existing IPv4 information (whether one is adding IPv6 records to an existing zone or starting from scratch).
- An alternative record format for IPv6 information is the A6 record.
 - ▶ This format is experimental and is not recommended for use
 - ▶ The **`allow-v6-synthesis { }`** `named.conf` option could be useful if you have to deal with other servers that use A6 records
 - ▶ Tells `named` to query for an A6 first when it receives a AAAA query. If the A6 query fails, then the server tries the original AAAA lookup
 - ▶ It takes an address list as an argument--queries from those hosts will invoke this behavior

Example IPv6-enabled forward zone file

```
$TTL 86400
$ORIGIN raleigh.ibm.com.

tcp          SOA      linuxipv63.tcp      dnsadmin.tcp  ( ... )

linuxipv63.tcp NS      tcp
mvs073.tcp   NS      tcp

;hosts in between snipped

mvs073.tcpA  9.67.115.5
mvs073-v6.tcp A      9.67.115.5
              AAAA  fec0::A:9:67:115:5      ; site-local VIPA
              AAAA  50c9:c2d4::A:9:67:115:5 ; global VIPA

;other hosts would follow below...
```

Reverse zones for IPv6 DNS

- IPv6 reverse zones work similarly to IPv4 reverse zones, but there are two reverse domains (ip6.int and ip6.arpa) instead of one (in-addr.arpa).
 - ▶ In both IPv6 domains, the same 'nibble' label format is used:
 - 5.0.0.0.5.1.1.0.7.6.0.0.9.0.0.0.A.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.ip6.int.
 - ▶ The PTR record is used for the IPv6 reverse mapping, like IPv4:
 - 5.0.0.0.5.1.1.0.7.6.0.0.9.0.0.0.A.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.ip6.int. PTR mvs073
 - 5.0.0.0.5.1.1.0.7.6.0.0.9.0.0.0.A.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.ip6.arpa. PTR mvs073
- Why two reverse domains?
 - ▶ RFC 3152 deprecates the ip6.int domain for IPv6 reverse mapping and says that ALL IPv6 reverse zones should fall under ip6.arpa.
 - ▶ Many Resolvers implement IPv6 reverse lookups using the ip6.int domain, but the process of migrating to ip6.arpa is on the way.

Reverse Zones for IPv6 DNS (*continued*)

- Serving two zones containing essentially the same data is more complex, but does not have to be an administrative headache.
 - ▶ Carefully done, the same zone data file can be used for both zones!

```
# example named.conf section...
zone "A.0.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.ip6.int" {
type master;
file "db.ipv6.reverse";
};

zone "A.0.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.ip6.arpa" {
type master;
file "db.ipv6.reverse";
};
```

- Zone files inherit a default \$ORIGIN value from the name of the zone in named.conf. And since the data in the reverse zones is identical other than the top-level domain, the same file can be used for both zones, and named will append the named.conf \$ORIGIN onto each record.

Example reverse IPv6 zone file

```
$TTL 86400
;The default origin is the name of the zone:
; A.0.0.0.0.0.0.0.0.0.0.0.c.e.f.ip6.int. or ...ip6.arpa.

@ SOA mvs073.tcp.raleigh.ibm.com. hostmaster. ( ... )

@ NS mvs073.tcp.raleigh.ibm.com.
@ NS linuxipv6.tcp.raleigh.ibm.com.

5.0.0.0.5.1.1.0.7.6.0.0.9.0.0.0 PTR mvs073.tcp.raleigh.ibm.com.
7.1.0.0.5.1.1.0.7.6.0.0.9.0.0.0 PTR winipv6.tcp.raleigh.ibm.com.

;other records here...

5.4.0.0.4.1.1.0.7.6.0.0.9.0.0.0 PTR linuxv63.tcp.raleigh.ibm.com.
6.4.0.0.4.1.1.0.7.6.0.0.9.0.0.0 PTR linuxv64.tcp.raleigh.ibm.com.

;other records would continue below...
```

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM (logo)		Cloudscape	Informix	OS/390
e (logo) business		DB2	iSeries	OS/400
AIX	DB2 Universal Database	Lotus	pSeries	zSeries
				WebSphere
				xSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested these products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
 IBM Corporation
 North Castle Drive
 Armonk, NY 10504-1785
 U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2004,2005. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.