Novell NetWare® 6.5

www.novell.com

NOVELL IPV6 ADMINISTRATION GUIDE

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

You may not export or re-export this product in violation of any applicable laws or regulations including, without limitation, U.S. export regulations or the laws of the country in which you reside.

Copyright © 2003 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

U.S. Patent No. 5,157,663; 5,349,642; 5,455,932; 5,553,139; 5,553,143; 5,572,528; 5,594,863; 5,608,903; 5,633,931; 5,652,854; 5,671,414; 5,677,851; 5,692,129; 5,701,459; 5,717,912; 5,758,069; 5,758,344; 5,781,724; 5,781,733; 5,784,560; 5,787,439; 5,818,936; 5,828,882; 5,832,274; 5,832,275; 5,832,483; 5,832,487; 5,859,978; 5,870,561; 5,870,739; 5,873,079; 5,878,415; 5,884,304; 5,893,118; 5,903,650; 5,903,720; 5,905,860; 5,910,803; 5,913,025; 5,913,209; 5,915,253; 5,925,108; 5,933,503; 5,933,826; 5,946,002; 5,946,467; 5,956,718; 5,956,745; 5,964,872; 5,974,474; 5,983,223; 5,983,234; 5,987,471; 5,991,810; 6,002,398; 6,014,667; 6,016,499; 6,023,586; 6,029,247; 6,052,724; 6,061,726; 6,061,740; 6,061,743; 6,065,017; 6,081,774; 6,081,814; 6,094,672; 6,098,090; 6,105,062; 6,105,069; 6,105,132; 6,115,039; 6,119,122; 6,144,959; 6,151,688; 6,157,925; 6,167,393; 6,173,289; 6,216,123; 6,219,652; 6,233,859; 6,247,149; 6,269,391; 6,286,010; 6,308,181; 6,314,520; 6,324,670; 6,338,112; 6,345,266; 6,353,898; 6,424,976; 6,466,944; 6,477,583; 6,477,648; 6,484,186; 6,496,865; 6,510,450; 6,516,325; 6,519,610; 6,532,451; 6,532,491; 6,539,381; RE37,178. Patents Pending.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.

www.novell.com

Novell IPv6 Administration Guide
April 2003

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

ConsoleOne is a registered trademark of Novell, Inc in the United States and other countries.
Internetwork Packet Exchange and IPX are trademarks of Novell, Inc.
NetWare is a registered trademark of Novell, Inc., in the United States and other countries.
NetWare Loadable Module and NLM are trademarks of Novell, Inc.
Novell is a registered trademark of Novell, Inc., in the United States and other countries.
Novell Cluster Services is a trademark of Novell, Inc.
Novell Directory Services and NDS are registered trademark of Novell, Inc in United States and other countries.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Contents

About This Guide

This Internet Protocol version 6 (IPv6) Administration Guide provides information about the basic features of IPv6, transition mechanisms from IPv4, and how to set up the IPv6 network. Novell IPv6 now ships with Novell® TCP/IP stack as an additional component.

This guide is divided into the following sections:

Documentation Updates

For the most recent version of the Novell IPv6 Administration Guide, see Novell online documentation (http://www.novell.com/documentation/beta/nw65/index.html.)

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

In this documentation, a trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

1 Understanding IPv6

IPv6 is a network layer protocol that resolves the inherent IPv4 problems and incorporates many enhancements. IPv6 solves the Internet scaling problem (addresses), provides a flexible transition mechanism, meets the needs of mobile users, and supports automatic configuration (plug-and-play).This chapter provides the following information about IPv6:

Understanding IPv6

With the unprecedented growth of the Internet and the steady increase of users who use the Internet for varied services, there is a need to increase the Internet address spaces. This is to facilitate real-time traffic, flexible congestion control schemes, security, and privacy. The emerging range of network intelligent devices such as mobile phones and home area networks, has further accentuated the need for larger address spaces.

IPv6 aims to provide larger address spaces to overcome the shortcomings of IPv4. To ensure that IPv6 provides all the features that IPv4 does not, the Internet Engineering Task Force (IETF) revisited the definition and functionality that IPv4 offered. IPv6 is designed to produce a streamlined format while integrating support for emerging services such as expanded address configuration, quality of service, security, and support for mobile devices.

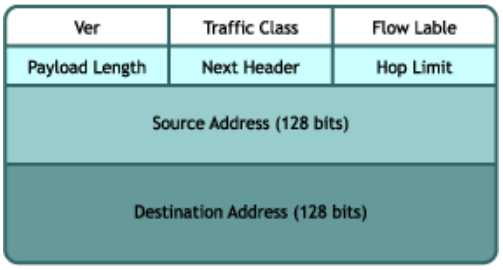
This section explains the following:

IPv6 Header Format

Unlike in IPv4, IPv6 options are placed in separate extension headers and are located between the IPv6 headers and the transport layer headers. IPv6 does not require all the routers on a path to examine these header options. The redundant fields from the IPv4 header have been removed for IPv6. These improvements enhance the IPv6 protocol performance, because they cut down on the additional processing.

The following diagram shows the IPv6 header.

Figure 1 Header Format



Extension Headers

The optional Internet layer information is encoded in separate headers that can be placed between the IPv6 header and the upper-layer header in a packet. These extension headers are identified by a distinct next-header value. The IPv6 packet might or might not carry these extension headers. The following are the currently defined extension headers options:

Option	Functionality
Authentication	Integrity and Authentication
Destination options 1	Options to be examined by intermediate nodes
Destination options 2	Options to be examined by destination node only
Fragmentation	Fragmentation and reassembly
Hop-by-Hop	Special option for processing at every node
Routing	Extended Routing (Loose Source Route)
Security encapsulation	Confidentiality

IPv6 Addressing

IPv6 addresses are 128 bits and identify interfaces or sets of interfaces. The following are the three types of IPv6 addresses:

- **Unicast** identifies a single interface.
- **Anycast** identifies a set of interfaces. A packet sent to this address will be forwarded to the nearest interface with the same address, according to the routing protocols' measure of distance.
- **Multicast** identifies a group of interfaces. A packet sent to this address will be sent to all interfaces in the group.

The IPv6 address space works out to be:

2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456

IPv6 Security

IPv6 offers the following integrated security services:

- The IPv6 Authentication Header provides authentication to IPv6 datagrams.
- The IPv6 Encapsulating Security Header provides integrity and confidentiality to IPv6 datagrams.

IPv6 Routing

RIPv6 and OSPFv6 are protocols that enable routers to exchange information for computing routes through an IPv6 network. The RIPv6 and OSPFv6 protocols must be implemented only on routers because IPv6 hosts use the Neighbor Discovery Protocol to retrieve information about their neighboring nodes. The RIPv6 protocol works on UDP and the OSPFv6 protocol works on IPv6.

Quality-of-Service Capabilities

The IPv6 protocol provides some Quality-of-Service (QoS) mechanisms for those packets that require special handling. The Flow Label and Traffic Class fields in the IPv6 header are used to identify these packets, which include packets that require nondefault quality of service, real-time service, or relative priority. This is especially useful for real-time and multimedia applications.

Two types of header fields enable QoS:

- **Flow Label** identifies a flow, which is a sequence of packets sent from a particular source to a particular destination or multiple destinations for which the source desires special handling.
- **Traffic Class** identifies and distinguishes between different classes or priorities of IPv6 packets.

Address Auto Configuration

Address auto configuration enables a host to automatically learn its interface addresses. This enables the host to operate in a plug-and-play mode.

Path Maximum Transfer Unit

Every network interface has a maximum packet size that it can transfer across the network. This is called the interface's Maximum Transfer Unit (MTU). The complete path that data packets travel to reach the destination might span across many routers with different MTUs. The smallest MTU among all the routers in a path is referred as the path MTU.

If a packet starts out on a network segment with a large MTU, it might arrive at a router with a smaller MTU. The intermediate routers are not allowed to fragment the packet and, therefore the packet would not be able to traverse through this link.

Before sending the data packets, we it is recommended that each host perform the path MTU discovery process and determine the optimum size for the full path from the source to the destination. To ascertain the path MTU, the host can send out a probe packet of the largest size possible. If it cannot traverse through some link in the path, the host will receive a Packet Too Big notification and will be further informed about the optimum size of data packets that can be sent through that link.

The path MTU for each interface can be configured in the ip6.cfg file. The size specified in this file becomes the maximum size of the outgoing data packet from that network link.

Comparing IPv4 and IPv6

The following features differentiate IPv6 from IPv4:

- The IPv6 header (40 bytes) is double the size of the IPv4 header (20 bytes).
- IPv6 (128 bits) has four times as many address bits as IPv4 (32 bits).
- IPv6 has stackable extension headers that replace the IPv4 options. Several extension headers can be stacked on top of the previous extension headers.
- The IPv6 header is not protected by checksum. Instead, UDP checksumming is mandated in IPv6.
- Fragmentation-related fields now belong to the fragment extension header in IPv6.
- The length of the header, protocol type, and the Time to Live are redefined in the IPv6 header.
- Intermediate fragmentation is not allowed in IPv6.

Two additional features are improvements over IPv4:

- "Addressing Differences" on page 12
- "Configuration" on page 12

Addressing Differences

IPv6 supports private and public addresses as part of the architecture and associates them with a lifetime. IPv4 added the concept of scope or private addresses at a later time. Mechanisms like Dynamic Host Control Protocol try to associate lifetime to addresses in IPv4.

IPv6 addresses uses unicast, multicast, and anycast addresses. IPv4 does not have the anycast addressing as part of the base specification.

Configuration

IPv6 brings in plug-and-play support for hosts as part of the base specification. Routers can be configured to advertise subnet prefixes and MTU parameters. Most of the facilities provided by IGMP router discovery and ARP in IPv4 are provided as part of the Neighbor Discovery protocol in IPv6.

Implementing Novell IPv6

IPv6 on NetWare® enables the use of the IPv6 protocol natively over the NetWare server platform by NetWare applications like NDS®, Proxy, and Winsock. NetWare does this by using the IPv6 features that best suit Novell software. Also, IPv6 is used as a part of the existing TCP/IP stack and functions as an add-on component for TCP/IP.

The following RFCs are supported by the Novell IPv6 protocol:

RFC 2460 - Internet Protocol version 6 (IPv6) Specification
RFC 2461 - Neighbor Discovery for IPv6
RFC 2462 - IPv6 Stateless Address Autoconfiguration
RFC 2463 - Internet Control Message Protocol (ICMPv6)
RFC 2464 - Transmission of IPv6 Packets over Ethernet Networks (Only EthernetII Format)
RFC 2553 - Basic Socket Interface Extensions for IPv6

RFC 2373 - IPv6 Addressing Architecture
RFC 2893 - Transition Mechanisms for IPv6 Hosts and Routers

2 Transitioning from IPv4 to IPv6

As moving exclusively to IPv4 is not a practical option for most organizations, it is possible to make the move gradually and to use mixed environments while the transition is taking place. The question for most organizations is when, how, where, and how much to transition.

The above mechanisms cannot be compared for efficiency and performance. However, one or more of these mechanisms will be used eventually for a complete transition.

Some individual networks within an organization can be upgraded as a whole, creating small IPv6 networks surrounded by IPv4 networks, but IPv4/IPv6 gateways are necessary at the borders of these networks to interoperate with IPv4 networks. Different IPv6 networks can also communicate with each other through the IPv4 Internet by setting up IPv6/IPv4 tunnels.

Some organizations will migrate host by host, with dual-protocol IPv4/IPv6 nodes scattered throughout the existing IPv4 network. These nodes will be able to interoperate with each other in native IPv6, or with IPv6 nodes outside the network by tunneling IPv6 inside IPv4 packets.

See the following sections for more information about how IPv6 interoperability with IPv4 is enabled:

- "Dual Stack" on page 15
- "Tunneling" on page 16
- "Testing Your Setup with 6Bone" on page 18

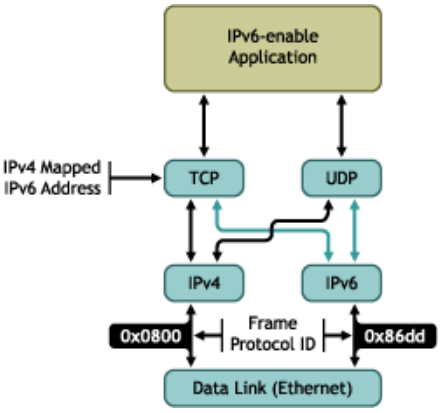
Dual Stack

The IPv6 dual stack mode assumes the following:

- Both IPv4 and IPv6 stacks are enabled
- Applications can talk to both IPv6 and IPv4
- Your choice of the IP version is based on name lookup and application preference

The following diagram shows the dual stack approach.

Figure 2 Dual Stack



Dual stack hosts can handle both IPv4 and IPv6 clients by using IPv4-mapped IPv6 addresses. For example, the following process shows how an IPv4 TCP client communicates with an IPv6 server:

1. The IPv6 server starts and creates an IPv6 listening socket.
2. The IPv4 client calls gethostbyname and finds a record for the server.
3. The client calls connect and the client's host sends IPv4 SYN to the server.
4. The server host receives the IPv4 SYN directed to the IPv6 listening socket. The server sets a flag indicating that this connection is using IPv4-mapped IPv6 addresses and responds with an IPv4 SYN/ACK.
5. All communication between the client and server takes place using IPv4 datagrams.
6. Unless the server explicitly checks whether this IPv6 address is an IPv4 mapped IPv6 address, the server never knows that it is communicating with an IPv4 client.

Tunneling

Tunneling requires only edge ingress and egress router upgrades until native IPv6 networks are commercially deployed or offered end-to-end. Two tunneling mechanisms are explained here:

- “Automatic and Configured Tunneling” on page 16
- “Default Configured Tunnel” on page 18

Automatic and Configured Tunneling

IPv6/IPv4 hosts and routers can tunnel IPv6 datagrams over regions of IPv4 routing topology by encapsulating them within IPv4 packets. Tunneling can be used in the following ways:

- **Router-to-Router —** IPv6/IPv4 routers interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves.
- **Host-to-Router —** IPv6/IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6/IPv4 router that is reachable through an IPv4 infrastructure. This type of tunnel spans the first segment of the packet's end-to-end path.
- **Host-to-Host —** IPv6/IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end-to-end path that the packet takes.

- **Router-to-Host —** IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6/IPv4 host. This tunnel spans only the last segment of the end-to-end path.

Automatic Tunneling

In the Host-to-Host and Router-to-Host scenarios, IPv6 packets are tunneled all the way to the destination. The tunnel end point is the node that the IPv6 packet is addressed to. Because the end point of the tunnel is the destination for the IPv6 packet, the tunnel end point can be determined from the destination IPv6 address of the packet. If the address is an IPv4-compatible address (RFC 2373), the lower-order 32 bits hold the IPv4 address of the destination node and can be used as the tunnel end point address.

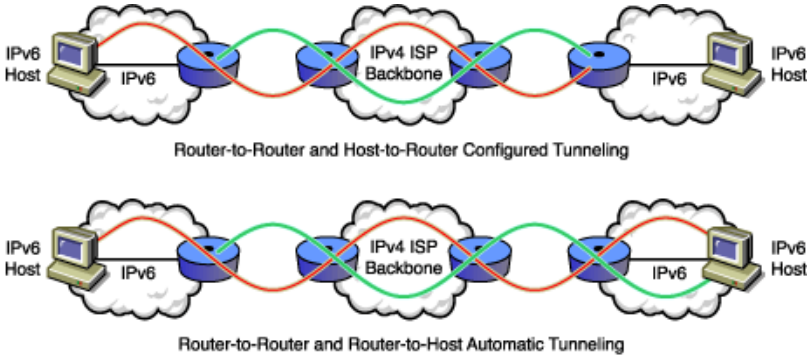
This avoids the need for explicit configuration of the tunnel end point address, which is the reason this method is known as *automatic tunneling*. It requires that the IPv6 address must be an IPv4-compatible IP address.

IPv6/IPv4 nodes need to determine which IPv6 packets can be sent through automatic tunneling. One method is to use the IPv6 routing table to direct automatic tunneling. You can have a special static routing table entry for the prefix 0:0:0:0:0:0/96 (that is, a route to the all-zeros prefix with a 96-bit mask).

Packets that match this prefix are sent to a pseudo-interface driver which performs automatic tunneling. Because all IPv4-compatible IPv6 addresses will match this prefix, all packets to those destinations can be auto-tunneled.

The following diagram shows automatic tunneling.

Figure 3 Automatic Tunneling



Configured Tunneling

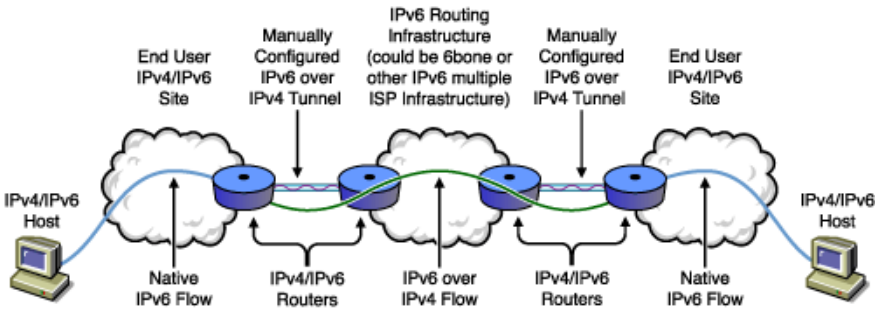
In the Router-to-Router and Host-to-Router scenarios, the IPv6 packet is tunneled to a router. The tunnel end point is a router that must decapsulate the IPv6 packet and forward it to the destination. The end point of the tunnel is different from the destination, so the addresses of the IPv6 packet being tunneled do not provide the IPv4 address of the tunnel end point. The tunnel end point address must be determined from the configuration information on the node performing the tunneling, which is the reason this method is called *configured tunneling*.

The tunnel end point address is determined from the configuration information in the encapsulating node. For each tunnel, the encapsulating node must store the tunnel end point address. When an IPv6 packet is transmitted over a tunnel, the tunnel end point address configured for that tunnel is used as the destination address for the encapsulating IPv4 header.

The routing information on the encapsulating node determines which packets to tunnel. This is done via a routing table that directs packets based on the destination address using the prefix mask and match technique.

The following diagram shows a configured tunnel.

Figure 4 Configured tunneling



Default Configured Tunnel

Nodes that are connected to IPv4 routing infrastructures can use a configured tunnel to reach an IPv6 backbone. If the IPv4 address of the IPv6/IPv4 border router is known, a tunnel can be configured to that router.

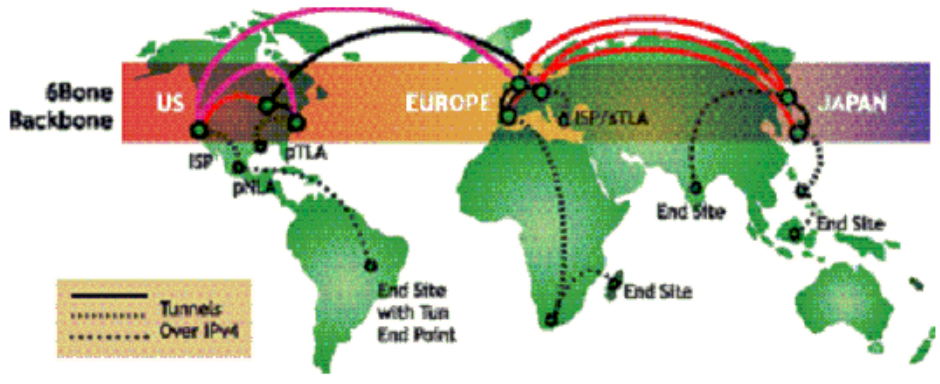
This tunnel can be configured as the default route. All IPv6 destination addresses will match the route and could potentially traverse the tunnel. The tunnel end point address of such a default tunnel could be the IPv4 address of the IPv6/IPv4 border router. Novell uses a default configured tunnel to reach the IPv6/IPv4 border router.

Testing Your Setup with 6Bone

6Bone is a logical test IPv6 network, overlaid on the IPv4 Internet.It is an independent outgrowth of the IPv6 project, resulting from an informal collaboration between the U.S., Japan, and Europe. You can join the network and use it to test your setup.

The following diagram shows the 6Bone network.

Figure 5 The 6Bone Network



Globally addressable IPv6 has a three-level hierarchy that includes the following:

- A public topology (the 48-bit external routing prefix)
- A site topology (typically a 16-bit subnet number)
- An Interface Identifier (usually an automatically generated 64-bit number unique on at least the local LAN segment)

The public topology has two or more levels of hierarchy, specifying the Top Level Aggregator (typically a high-level ISP), Next Level Aggregators (zero or more mid-level ISPs), and a final Next Level Aggregator (which is the end user site). The end user sites get their address prefixes from an ISP that provides their IPv6 service.

To join 6Bone, you get a 48-bit IPv6 external routing prefix from an existing pTLA (pseudo-Top Level Aggregator) 6Bone ISP.

To do this, you use the registry database to identify a suitable pTLA, then contact one of the listed registry contacts through e-mail.

You'll receive an end point address and an Ipv6 format prefix, which you should enter in to the centralized registry database. You can then use the 6Bone network to test your setup for functionality and interoperability.

3 Setting Up Novell IPv6

This section covers the basics of how to install required files and set up the Novell® IPv6 protocol. After IPv6 is set up, the transmission of packets through IPv4 and IPv6 depends on the address type specified in the packets. The following setup and configuration requirements are discussed here:

Required NLM Files

The following core NetWare® Loadable Module™ (NLM™) files used by the Novell IPv6 stack should be copied to the sys:\system directory:

- **bsdsock.nlm —** NLM for socket libraries
- **ipv6.nlm —** IPv6 core NLM
- **netlib.nlm —** NLM housing utility function for scheduler address libraries, etc.
- **resolv.nlm —** NLM used for DNS libraries
- **static6.nlm —** Static routing NLM
- **tcp.nlm —** TCP modified to support IPv6 end points
- **tcpip.nlm —** IPv4 NLM for setting up tunnels

IPv6 Files on a NetWare Server

After the NetWare server is installed, the IPv6-related NLM files are available for IPv6 setup and configuration. Other required NLM files — such as tcp.nlm, tcpip.nlm, bsdsock.nlm and netlib.nlm — will already be loaded when the server comes up. The following sections list the directory structures for the IPv6-related files on the NetWare server:

sys:\research\ipv6

The following files are available at this location:

- gw6
- ip6.cfg
- ip6.ncf
- ipv6.nlm
- resolv.nlm
- rtadvd.cfg
- static6.nlm
- uip6.ncf

sys:\research\ipv6\apache

All the Apache related NLM files are available at this location.

sys:\research\ipv6\ftp

The following files are available at this location:

- nwftpd6.msg
- nwftpd6.nlm

sys:\research\ipv6\nls\4

The following files are message files for English:

- ipv6.msg
- static6.msg

sys:\research\ipv6\nls\9

The following files are message files for Japanese:

- ipv6.msg
- static6.msg

sys:\research\ipv6\nls\12

The following files are message files for Portuguese:

- ipv6.msg
- static6.msg

sys:\research\ipv6\ping6

The following file is available at this location:

- ping6.nlm

Installing Novell IPv6

1. Copy the ipv6.nlm, static6.nlm, and resolv.nlm files from sys:\research\ipv6 to sys:\system.
2. Copy the ip6.cfg, rtadvd.cfg, and gw6 files from sys:\research\ipv6 to sys:\etc.

   These are sample files. Appropriately modify information like board name, addresses required, etc.
3. Copy the ip6.msg and static6.msg files from sys:\research\ipv6\nls\number to sys:\system\nls*number*.

   Here *number* will be 4, 9, or 12. These are the message files for English, Japanese, and Portuguese, respectively.
4. Copy the ip6.ncf and uip6.ncf files from sys:\research\ipv6 to sys:\system.
5. Run ip6.ncf at the command prompt.

   The IPv6 stack is now installed and ready to use.

Utilities That Can Use Novell IPv6

This section discusses the utilities that can use IPv6 and the manner in which they can be used:

- "Apache2" on page 23
- "FTP6" on page 24
- "IPTrace6" on page 24
- "PING6" on page 25

Apache2

Server Side

1. Copy the WinSock NLM files from the sys:\research\ipv6\winsock directory to the c:\nwserver directory .
2. Copy the folders and NLM files from the sys:\research\ipv6\apache2 directory to the sys:\apache2 directory.
3. Load ipv6.nlm.
4. Load resolv.nlm.
5. Load sys:\apache2\apache2.nlm.

This starts the Apache Server.

Client Side (Browser)

This information is specific to the Internet Explorer (IE) on the Windows XP platform. It can also apply to Support Patch for IPv6 for Windows 2000 nodes.

The default browser (IE) that comes with Windows XP does not support literal IPv6 strings as addresses. Therefore, you cannot use the address directly in the browser as http://[3ffe::1]. You have to use DNS names for the address.

1. Disable the proxy in your Browser.

   In IE, click Tools > Internet Options > Connections > LAN Settings and uncheck the *Use a proxy server for your LAN* check box.

   This must be done in case the DNS query is directed to the proxy, which might not have IPv6 support.

2. Verify the DNS entry.

   - If you have a DNS server supporting IPv6 Name-to-Address resolution, you can depend on that for resolution of the query.
   - Otherwise, you can configure the DNS name in the c:/windows/ system32/drivers/etc/ hosts file, similar to the format used for IPv4. For example:

     3ffe::1 ipv6_host

Http://ipv6_host then connects to the Apache Server.

FTP6

1. Copy the nwftpd6.nlm filr from the sys:\research\ipv6\ftp directory to the sys:\system directory.
2. Copy the nwftpd6.msg file from the sys:\research\ipv6\ftp directory to the sys:\system\nls\4 directory.
3. In order to use IPv6-enabled FTP Server (nwftpd6.nlm), make sure to unload the IPv4 version in the system (unload nwftpd.nlm).
4. Run nwftpd6.nlm. It will autoload ftpif.nlm.
5. Enter **`nwftpd6 -a`** at the server console to create an anonymous user for FTP access.

Nwftpd6.nlm binds to all the interfaces in the system.

IPTrace6

IPTrace6 is a debugging tool used to trace the path taken by the packet from the source host to reach the destination host. It lists out the IPv6 addresses of the intermediate routers that have been traversed to reach the destination. It uses the ICMPv6 error messages to achieve the same.

You can locate IPTrace6 in sys:/research/ipv6/tools/ iptrace6.nlm.

Enter the following command to use the IPTrace6 feature:

`IPTrace6 `*`destination`*` [Hops=`*`max_hops`*`] [StartHop=`*`starting_ttl`*`] [Wait = `*`max_wait_time`*`] [Port=`*`destination_port`*`] [Pkt = `*`number_of_packets_for_each_hop`*`]`

Parameters for IPTrace6

- **Hops** specifies the maximum number of hops that will be made before IPTrace6 stops searching. Default = 30.
- **StartHop** specifies the initial value of the time-to-live (ttl) in the outgoing packet. For example, if there are three hops to the destination and the StartHop is specified as 2, the IPTrace6 display skips the router at the first hop and starts from the second one. Default = 1.
- **Wait** specifies the time (in seconds) to wait for the response (ICMPv6 Time Exceeded) to a probe. If no reply is received within this time, an asterisk (*) is displayed. Default = 5.
- **Port** specifies the UDP port number that the IPTrace6 packets are sent to. It should be greater than 6000. Default = 40001.
- **Pkt** specifies the number of packets sent with the same ttl value. Default = 3.

Some examples of using IPTrace6 are given below:

```
IPTrace6 www.novell.com
```

```
IPTrace6 www.novell.com Starthop=3 Pkt=4
```

PING6

1. Copy the ping6.nlm file from the sys:\research\ipv6\ping6 directory to the sys:\system directory.
2. Run ping6.nlm to test the communication between any two nodes.

   HINT: Just typing **ping6** without any options will open the help screen.

Novell IPv6 Configuration

You can configure the IPv6 stack by modifying the default ip6.cfg file found under sys:\etc. You can also use the rtadvd.cfg file in conjunction with the ip6.cfg file to make the node function as an advertising interface. This section describes the formats for the configuration files:

- “Ip6.cfg File” on page 25
- “Rtadvd.cfg File” on page 28

Ip6.cfg File

The configuration of the IPv6 stack is possible through the ip6.cfg file placed under sys:\etc. To configure the IPv6 stack, modify the default file provided with the stack.

Configuration File Format

```
[Interface All]

Router  Yes | No

Autotunnel Yes | No

6to4  Yes | No

[Interface Interface name]

Addr v6_address_128_bit
```

```
Prefixlen Prefix_length_of_Addr
mtu MTU_for_this_interface
rsdelay Router_solicitation_delay_in_seconds
rsinterval Delay_between_the_router_solicitation_in_seconds
rstransmits Number_of_router_solicitations_transmitted
autoconf Yes | No
acceptra Yes | No
acceptredirect Yes | No
hoplimit Hop_limit_to_be_specified_in_the_outgoing_packet
delayFirstProbeTime Neighbor_discovery_parameter
baseReachableTime Neighbor_discovery_parameter
retransTime Delay_between_successive_neighbor_solicitations
sendRedirect Yes | No
DadTransmits [1 - 3]
```

Example Configuration File

```
[Interface All]
Router      Yes
Autotunnel      No
6to4      No
[Interface CE100B_1_EII]
Addr     3ffe:501:ffff:100::1
Prefixlen     64
mtu             1280
rsdelay        1
rsinterval     4
rstransmits     3
autoconf      Yes
acceptra      Yes
acceptredirect    Yes
hoplimit       255
delayFirstProbeTime      5
baseReachableTime      30
retransTime     1
sendRedirect     Yes
DadTransmits     3
```

Field Level Descriptions with Default Values

Table 1 **For the Common Interface Record**

Field Name	Description	Default Value
Router	Router/Host functionality	No (Host)
Autotunnel	Autotunnel enable/disable	No (disable)
6to4	6to4 tunnel enable/disable	No (disable)

Table 2 **For the Interface Specific Record**

Field Name	Description	Default Value
Addr	Correponding IPV6 Address	None
Prefix Len	Prefix length of the address	None
mtu	MTU of the interface	MaxReceiveSize of the corresponding driver
Rsdelay	Router solicitation delay (in seconds)	1
Rsinterval	Router solicitation interval (in seconds)	4
Rstransmits	Number of times router solicitation is transmitted	3
Autoconf	Indicates if autoconfiguration should be performed on the interface (yes/no)	Yes
Acceptra	Indicates if router advertisements should be accepted on this interface (yes/no)	Yes
Acceptredirect	Indicates if redirects should be accepted on this interface (yes/no)	Yes
Hoplimit	Hop limit filled in the outgoing packets	255
DelayFirstProbeTime	Time (in seconds) to wait in Delay state	5
BaseReachableTime	Neighbor discovery parameter (in seconds)	30
RetransTime	Time (in seconds) between successive neighbor solicitations	1
SendRedirect	Indicates if redirects should be sent on this interface (yes/no)	Yes

Field Name	Description	Default Value
DadTransmits	Number of consecutive neighbor solicitation messages sent while performing Duplicate Address Detection on a tentative address.	3

Rtadvd.cfg File

You can use the rtadvd.cfg file in conjunction with the ip6.cfg file to make the node function as an advertising interface. For the values in the rtadvd.cfg file to take effect, you need to set the Router field to Yes in the ip6.cfg file. Refer to "Configuration File Format" on page 25.

Configuration File Format

Router Advertisement

You need to set the following file formats to configure Router Advertisement (interface specific):

```
[Interface Interface_name]

RASendAdvertisements Yes | No

RAMaxInterval [4 - 1800]

RAMinInterval [3 - 0.75 *

RAManagedFlag  Yes | No

RAOtherConfigFlag  Yes | No

RALinkMTU [0 -

RAReachableTime [0 - 3,600,000]

RARetransTimer  Retransmit_time_in_milliseconds

RACurHopLimit  Hop_limit

RADefaultLifeTime [0 | MaxRtrAdvInterval - 9000]
```

NOTE: This record can occur only once for an interface.

Prefix List

You need to set the following file formats to configure the Prefix list to be advertised for each interface:

```
[RAPrefixListIf Interface name]

Prefix  Prefix_to_be_advertised

PrefixLen [64 - 128]

RAValidLifeTime  Valid_life_time_of_the_Prefix_in_seconds

RAPreferredLifeTime  Preferred_life_time_of_the_Prefix_in_seconds

RAAutonomousFlag  Yes | No

RAOnLinkFlag  Yes | No

RAReal_Fixed_TimeFlag   Real | Fixed
```

NOTE: This record can occur multiple times for an interface depending on the number of prefixes that are to be advertised.

Example Configuration File

```
[Interface DLKRTS_EII ]
RASendAdvertisements      Yes
RAMaxInterval        20
RAMinInterval        3
RAManagedFlag         Yes
RAOtherConfigFlag       Yes
RALinkMTU            0
RAReachableTime        200
RARetransTimer        3
RACurHopLimit        255
RADefaultLifeTime       1800

[RAPrefixListIf DLKRTS_EII]
Prefix                3ffe:dad::0
PrefixLen            64
RAValidLifeTime        300
RAPreferredLifeTime    250
RAAutonomousFlag        Yes
RAOnLinkFlag        No
RAReal_Fixed_TimeFlag   Real

[RAPrefixListIf    DLKRTS_EII]
Prefix        3ffe:FEED::0
PrefixLen         64
RAValidLifeTime       50
RAPreferredLifeTime      30
RAAutonomousFlag       Yes
RAOnLinkFlag          No
RAReal_Fixed_TimeFlag        Fixed
```

Field Level Description with Default Values

Table 3 For the Router Advertisement Record

Field name	Description	Default Value
RASendAdvertisements	Router Advertisement enable/ disable.	No (disabled)
RAMaxInterval	The maximum time (in seconds) allowed between unsolicited multicast Router Advertisements transmitted from the interface.	600
RAMinInterval	The minimum time (in seconds) allowed between unsolicited multicast Router Advertisements transmitted from the interface.	198
RAManagedFlag	The value to be placed in the Managedaddress Configuration flag field in the Router Advertisement. Specifies whether hosts should use stateful autoconfiguration to obtain addresses.	No
RAOtherConfigFlag	The value to be placed in the Other Stateful Configuration flag field in the Router Advertisement. Indicates whether hosts should use statefulautoconfiguration to obtain additional information (excluding addresses).	No
RALinkMTU	The value to be placed in MTU options sent by the router. A value of zero indicates that no MTU options are sent.	0
RAReachableTime	The value to be placed in the Reachable Time field in the Router Advertisement messages sent by the router. A value of zero means unspecified (by this router). This denotes the time (in milliseconds) that a node assumes a neighbor is reachable after having received a reachability confirmation.	0
RARetransTimer	The value to be placed in the Retrans Timer field in the Router Advertisement messages sent by the router. A value of zero means unspecified (by this router). This denotes the time (in milliseconds), between retransmitted neighbor solicitation messages.	0

Field name	Description	Default Value
RACurHopLimit	The value to be placed in the Cur Hop Limit field in the Router Advertisement messages sent by the router. The value should be set to the current diameter of the Internet. A value of zero means unspecified (by this router).	255
RADefaultLifeTime	The time value (in seconds) to be placed in the Router Lifetime field of Router Advertisements sent from the interface. A Lifetime of 0 indicates that the router is not a default router. The Router lifetime applies only to the router's usefulness as a default router.	1800

Table 4 For the Prefix Description Record

Field name	Description	Default Value
Prefix	The prefix to be placed in Prefix Information options in Router Advertisement messages sent from the interface.	None
PrefixLen	Prefix length.	64
RAValidLifeTime	The length of time in seconds that the prefix is valid for the purpose of onlink determination.	2592000
RAPreferredLifeTime	The length of time (in seconds) that addresses generated from the prefix via stateless address autoconfiguration remain preferred.	604800
RAAutonomousFlag	Indicates whether this prefix can be used for autonomous address configuration	Yes
RAOnLinkFlag	Indicates whether this prefix can be used for onlink determination	Yes
RAReal_Fixed_TimeFlag	Indicates whether the lifetimes specified decrements in real time, or remains as a fixed time that stays the same in consecutive advertisements	Fixed

Novell IPv6 Commands

This section explains the commands that can be used to set up the Novell IPv6 stack.

- "Interface Related Commands" on page 32

Interface Related Commands

This section covers the following:

Bind Protocol

Syntax	bind *protocol_name board_name* addr *address* len *prefixlen*
Description:	Binds a communication protocol to a network board.
Example:	bind ip6 ce 100b bind ip6 ce 100b addr 3ffe:1 len 64

IP6 Configuration Display

Syntax	ip6config
Description:	Displays the board configuration, which includes all types of IPv6 addresses configured to various boards and the corresponding preferred and valid lifetimes.
Example:	ip6config

Unbind Protocol

Syntax:	unbind rt6add *protocol_name board_name*
Description:	Unbinds a communication protocol from a board .
Example:	unbind ip6 ce 100b

Routing Commands

This section covers the following:

Forwarding Capability Status Display

Syntax	ip6router
Description:	Displays the current status.
Example:	Displays the current status as "IP6Forwarding is ON" / "IP6Forwarding is OFF".

Route Addition

Syntax	Rt6add *IP6_Destination_Prefix Gateway_Address %Interface_Name Prefix_Length*
Description:	Adds an IPv6 Route to the IPv6 Routing Table. The Interface Name will use the default. Does not need to be specified for a global address.
Example:	Rt6add 3ffe:88::1234 fe80::8%CE100_1_EII 64 Rt6add 3ffd:: 3ffe:88::1234 64 To add a default route, give the unspecified address (::) as the IPv6Destination Prefix, and the prefix length as zero. For example: rt6add:: fe80::9%CE100B_1_EII 0

Route Deletion

Syntax	Rt6add *IP6_Destination_Prefix %Interface_Name Prefix_Length*
Description:	Deletes an IPv6 Route from the IPv6 Routing Table. The interface name will use the default. Does not need to be specified for a global address.
Example:	Rt6del fe80::8%CE100B_1_EII 64 Rt6add 3ffe:88::1234 64 To delete a default router, rt6del :: 0

Routing Table Display

Syntax	Rt6Table
Description:	Displays the Routing Table.
Example:	rt6table

Tunnel Commands

This section covers the following:

Configured Tunnel Creation

Syntax	tun6bind *tunnelname local_IPv6_address remote_IPv6_address local_IPv4_address remote_IPv4_address*
Description:	Creates one end of a tunnel.
Example:	tun6bind test-tunnel 3ffe::1 3ffe::2 164.99.150.45 164.99.151.99

Configured Tunnel Deletion

Syntax	tun6unbind *tunnelname*
Description:	Deletes a tunnel.
Example:	tun6unbind test-tunnel

Configured Tunnel Information Display

Syntax	Tun6List -t *tunnel_name* Tun6List -n *number_of_tunnels_to_display_at_a_time*
Description:	Displays the bound tunnels and the tunnel information.
Example:	Tun6List (Displays all the bound tunnels one by one) Tun6List -t test-tunnel (Displays details of test-tunnel alone) Tun6List -n 5 (Displays tunnel info five at a time)

Current Automatic Tunneling Status Display

Syntax	ip6autotunnel
Description:	Displays the status of Automatic Tunneling capability as "Auto Tunneling is ON" or "Auto Tunneling is OFF". Set this feature to on (in the ip6.cfg file) to automatically create corresponding compatible addresses for all the bound ipv4 addresses, thereby enabling the use of automatic tunneling with these addresses. Set this feature to off (in the ip6.cfg file) to automatically unbind all the compatible addresses created. Default = On.

Example:	ip6autotunnel

General Commands

Default Router List Display

Syntax	dfr6list
Description:	Displays the information about the default routers learned dynamically.

Destination Cache Display

Syntax	dc6list
Description:	Lists of destination cache Entries.

Destination Cache Flush

Syntax	dc6flush
Description:	Flushes the destination cache of the system.

Neighbor Cache Display

Syntax	nc6list
Description:	Displays the neighbor cache entries.

Neighbor Cache Entry Deletion

Syntax	nc6del *IP6Address Board_Name*
Description:	Deletes the neighbor cache entry for the specified neighbor of the specified interface.

Neighbor Cache Flush

Syntax	nc6flush
Description:	Flushes the neighbor cache entries of the system.

Prefix List Display

Syntax	pf6list
Description:	Displays the prefix information learned from routers.

4 Using Novell IPv6 in Your Network

The following sections provide details on how to use the Novell® IPv6 stack to set up tunneling within the corporate network between the IPv6 groups, as well as how to connect to 6Bone in order to experience the virtual global IPv6 network.

Setting Up Tunneling

You can use automatic or configured tunneling to set up your own IPv6 tunnel.

NOTE: In the following procedure, Host 1 and Host 2 belong to two different IPv6 networks. Router 1 and Router 2 are the two edge routers, which are dual stacks to interconnect the IPv6 and IPv4 networks. It is assumed that both Router 1 and Router 2 are NetWare® nodes.

Using Configured Tunneling

1. Make sure the IPv6 stack is loaded on both sides.
2. Configure one side of the tunnel to set up Router 1 by entering the following at the server console:

   ```
   Tun6bind test-tunnel 3ffe::1 3ffe::2 164.99.150.1 164.99.150.2
   ```

3. Configure the other side of the tunnel to set up Router 2 by entering the following at the server console:

   ```
   Tun6bind test-tunnel 3ffe::2 3ffe::1 164.99.150.2 164.99.150.1
   ```

   The tunnel names do not have to be same on both sides.

4. Test the connectivity by pinging across the tunnel.

   For example, on Router 1 enter

   ```
   Ping6 3ffe::2
   ```

Using Automatic Tunneling

Unlike configured tunneling, you do not need to explicitly set up an automatic tunnel.

NOTE: The following procedure sets up automatic tunneling between Host 1 and Host 2:

1. Make sure the IPv6 stack is loaded on both nodes.
2. Make sure the IPv6 Auto Tunnel feature is set to on for both nodes.
3. Test the connectivity by pinging across the tunnel. For example, enter

   ```
   to::164.99.151.2 from Host 1 and ::164.99.151.1 from H2.
   ```

Configuring 6to4 Tunnels

You can use the 6to4 tunneling technique to enable the host to encapsulate the IPv6 traffic in the IPv4 header and send over the IPv4 Internetwork (Internet). This is one of the mechanisms used to ease the transition of networks from IPv4 to IPv6.

The 6to4 feature is set to No by default. To enable it, set the 6to4 parameter to Yes in the ip6.cfg file under sys:\etc.

```
[Interface All]

6to4    Yes
```

This enables the machine as a 6to4 host. A 6to4 pseudo-interface is created with an IPv6 address of 2002:AABB:CCDD::AABB:CCDD, where AABB:CCDD is the colon-hexadecimal representation of the IPv4 address a.b.c.d assigned to the node. You can now send and receive 6to4 traffic over this machine. However, you need to ensure that the machine has a public IPv4 Internet address. Private addresses like 192.168.x.x or 10.x.x.x, 172.16/12, auto-conf addresses 169.254.x.x, and loopback addresses 127.x.x.x are ignored.

Configuring a 6to4 Router

You can configure a 6to4 node as a 6to4 router. The 6to4 router encapsulates the IPv6 packets received from the private interface into IPv4 packets before forwarding them on the public interface. For example, if you have only IPv6 nodes in your network or nodes that have IPv4 private addresses (refer previous paragraph), this will prevent the host from using the 6to4 feature directly. Therefore, the host can acquire the 6to4 prefix from the 6to4 router (which has a public IPv4 address and is connected to the IPv4 Internet) and configure IPv6 addresses. The host can then forward its IPv6 traffic to the 6to4 router that takes care of the tunneling.

To configure a 6to4 router:

1. Enable forwarding on the node. Refer to the Router variable in the Interface All record under “Configuration File Format” on page 25.
2. Configure the rtadvd.cfg file to advertise the 6to4 prefix (2002:AABB:CCDD::/48) to the IPv6 nodes on the private interface. Refer to “Rtadvd.cfg File” on page 28.
3. Configure the rtadvd.cfg file to advertise itself as the default router by setting the RADefaultLifeTime to a nonzero value. Refer to “Configuration File Format” on page 28.

Using Novell IPv6 to Connect to 6Bone

Getting an IPv6 Address Space

The Telecom lab sets up new connections within the backbone to provide access to new leaf sites. The lab provides an IPv6 prefix that is a pTLA (pseudo-Top Level Aggregator) and a 6Bone connection. If you want to connect to 6Bone, you must fill in a registration form and agree on the IPv4/IPv6 addresses to be used on the two sides. When this is done, you can connect to 6Bone.

Connecting to 6Bone

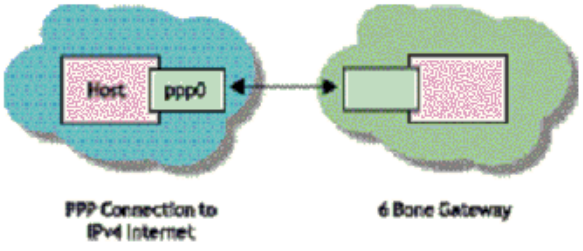
In contrast to the classic IPv6-over-IPv4 tunnel setup, you do not register at a 6Bone gateway or get forwarded to any IPv6 traffic (encapsulated in IPv4). Because your IPv6 address is provided from a source that already has a 6Bone connection, the tunnel establishment and maintenance are done by a Tunnel Broker where you got the IPv6 address.

Tunnel Broker is a mechanism to automatically manage tunnel requests coming from remote users. Standalone remote IPv6 users can register on a dedicated Web site, then obtain a script that will build an automatic tunnel to the IPv6 network.

To send IPv6 packets, the host takes the IPv6 packet and encapsulates it into an IPv4 packet. You still need a 6Bone-connected gateway that will decapsulate your packets and forward them to 6Bone.

The following figure illustrates this.

Figure 6 Connecting through Novell IPv6



Example

Your private network is uplinked through an IPv4-connected PPP link to a 6over4 gateway machine that is connected to 6Bone.

To connect to 6Bone the following sequence of events must occur:

- Assume that 202.169.139.51 is your IPv4 address.
- After becoming a registered user in one of the organizations which provide this service, you provide the IPv4 address at your end.
- Within a few minutes you receive an e-mail containing all the details needed to connect to 6Bone:

Tunnel Information	
Server IPv4 address	**163.162.170.170**
Server IPv6 address	3ffe:1001:0001:b000::2489
Server IPv6 link local address	Fe80::a3a2:aaaa
Client IPv4 address	202.169.139.51
Client IPv6 address	3ff2:1001:0001:b000::2488
Client IPv6 link local address	Fe80::caa9:8b33
Expire date	Mon Aug 20 08:26:32 2003

- Enter the following command to start pinging to the remote IPv6 address with IPv6 loaded:

`Tun6bind` *tunnel_name* 3ff2:1001:0001:b000::2489 3ffe:1001:0001:b000::2489 202.169.139.51 163.162.170.170

If you are able to get the response from pinging, you know you are connected to 6Bone.