EMC® VNX® Series

Release 8.1

Configuring and Managing Networking on VNX®

P/N 300-015-133 Rev 01

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Contents

Preface......7

Chapter 1: Introduction......9

System requirements......10

Restrictions......10

User interface choices......10

Related information......11

Chapter 2: Concepts......13

VNX networking......15

Devices and interfaces......15

Routing......16

Packet Reflect......17

VLANs......19

Using VLANs in a VNX environment......19

VLAN tagging......19

Address resolution......20

TCP performance......20

Differences between IPv4 and IPv6......21

IPv6 addresses......21

Automatic default router configuration......24

Neighbor discovery......25

Neighbor cache......25

Routing table......25

Duplicate Address Detection......25

IPv4 and IPv6 addresses supported on the same device......26

Chapter 3: Configuring..27

Configure network devices..28
Network device availability..28
Modify network device settings..29
Verify network device settings..30
Configure single or multiple IP interfaces..31
Configure an IPv4 interface..32
Configure a single device on multiple networks..34
Configure a single device on the same network..36
Configure an IPv6 interface..37
Configure an IPv6 interface with a nondefault prefix length..38
Configure IPv6 and IPv4 interfaces on the same device..39
Verify connectivity..41
Configure interface options..45
Enable VLAN tagging..45
Move an interface from one VLAN to another..46
Verify a VLAN ID..47
MTU configuration..48
Verify Data Mover connectivity..49
Verify Data Mover connectivity for an IPv6 address..51
Ping a link-local address..51
Ping a multicast address..52
Configure routing..52
List routing table entries for IPv4..53
Default interface..53
Add a default route..55
Add a static routing table entry..58
Display current RIP status..59
Disable RIP on a Data Mover..59
List routing table entries for IPv6..60
Add an IPv6 prefix entry to the routing table..61
Add a default gateway by using an IPv6 global unicast address..61
Add a default gateway by using an IPv6 link-local address..62

Chapter 4: Managing..63

Monitor network devices..64
Display device driver events..64
Manage IP interfaces..64

Modify an IPv4 interface......65
Modify an IPv6 interface......66
Modify VLAN tagging for an IPv4 device......67
Modify VLAN tagging for an IPv6 device......69
Modify the MTU size......70
Disable an IP interface......72
Enable an IP interface......73
View all IP interfaces on a Data Mover......74
View all IPv6 interfaces on a Data Mover......74
View all IP interfaces on all Data Movers......75
View all IPv6 interfaces on all Data Movers......77
View all IPv4 interfaces on a Data Mover......77
Manage routing......78
Delete an IPv4 route entry from the routing table......78
Delete an IPv6 route entry from the routing table......79
Flush the routing table......80
Remove all routing table entries......81
Disable RIP......81
Enable RIP......84
Manage address resolution......85
View ARP table entries......86
Add an ARP table entry......86
Delete an ARP table entry......87
Manage the neighbor cache......88
View the neighbor cache 88
Add an entry to the neighbor cache......90
Delete an entry from the neighbor cache......92
Clear all neighbor cache entries......93
Manage Packet Reflect......94
View network statistics......94
Modify Control Station networking......95
Set the Control Station IPv4 address and network mask......96
Set the Control Station IPv6 address and network mask......96
Set the Control Station IPv4 gateway......97
Set the Control Station IPv6 gateway......97
Modify the Control Station hostname......97
Improve TCP performance......100

Chapter 5: Troubleshooting......103

EMC E-Lab Interoperability Navigator..104
Tips for general networking problems..104
Tips for network interface problems..104
Error messages..105
EMC Training and Professional Services...105

Appendix A: Linux Services on the Control Station.........................107
List Linux services..108
Start Linux services...108
Enable Linux services..108
After enabling Linux services..108
Stop Linux services...109
Disable Linux services...109
After disabling services...109

Appendix B: VLANs to Support a Standby Data Mover....................111
Configure a standby Data Mover by using a single switch.........................112
Configure a standby Data Mover in a multiswitch environment...............115

Appendix C: Switch-Specific Guidelines for VLANs.........................119
Configure VLANs on the Alteon 180 switch...120
Configure VLAN tagging at the switch...120
Configure VLAN tagging at VNX IP interface...121

Appendix D: Configuring a Link-Local Address..............................123
Configure a link-local address..124
Verify a link-local address...124

Glossary...127

Index...131

Preface

As part of an effort to improve and enhance the performance and capabilities of its product lines, EMC periodically releases revisions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes.

If a product does not function properly or does not function as described in this document, please contact your EMC representative.

Special notice conventions

EMC uses the following conventions for special notices:

Note: Emphasizes content that is of exceptional importance or interest but does not relate to personal injury or business/data loss.

NOTICE Identifies content that warns of potential business or data loss.

⚠ CAUTION Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

⚠ WARNING Indicates a hazardous situation which, if not avoided, could result in death or serious injury.

⚠ DANGER Indicates a hazardous situation which, if not avoided, will result in death or serious injury.

Where to get help

EMC support, product, and licensing information can be obtained as follows:

Product information—For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to EMC Online Support (registration required) at http://Support.EMC.com.

Troubleshooting—Go to EMC Online Support at http://Support.EMC.com. After logging in, locate the applicable Support by Product page.

Technical support—For technical support and service requests, go to EMC Customer Service on EMC Online Support at http://Support.EMC.com. After logging in, locate the applicable Support by Product page, and choose either **Live Chat** or **Create a service request**. To open a service request through EMC Online Support, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

Note: Do not request a specific support representative unless one has already been assigned to your particular system problem.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications.

Please send your opinion of this document to:

`techpubcomments@EMC.com`

1 Introduction

This document provides information that enables you to perform basic network operations on your EMC VNX. It describes networking concepts such as devices and interfaces, network address resolution, routing, and virtual local area networks (VLANs).

This document is part of the VNX documentation set and is intended for network administrators responsible for configuring and maintaining an EMC file storage and network retrieval infrastructure.

Topics included are:

- System requirements on page 10
- Restrictions on page 10
- User interface choices on page 10
- Related information on page 11

System requirements

Table 1 on page 10 describes the EMC® VNX® software, hardware, network, and storage configurations.

Table 1. System requirements

Software	VNX version 8.0.
Hardware	The network configuration tasks that apply to your system depend on your specific network hardware, such as the adapter cards in the Data Movers and any switches used in your network.
Network	No specific network requirements.
Storage	No specific storage requirements.

Restrictions

VNX does not support the Ethernet IEEE 802.3 and 802.2 headers which are designed to support different LAN types such as token ring or Fiber Distributed Data Interface (FDDI).

As currently implemented, VNX VLAN functionality does not enable exporting file systems by VLAN. Users on different VLANs can see all exported file systems on a Data Mover. In NFS environments, you can use the server_export command to protect or limit user access. In CIFS environments, you can use Virtual Data Movers (VDMs) to administratively separate CIFS servers and their file systems, thus protecting or limiting user access. *Managing Volumes and File Systems with VNX Automatic Volume Management*, *Managing Volumes and File Systems for VNX Manually*, and *Configuring Virtual Data Movers on VNX* provide more information.

User interface choices

VNX offers flexibility in managing networked storage based on your support environment and interface preferences. This document describes how to configure and manage the VNX network by using the command line interface (CLI). You can also perform most of these tasks by using EMC Unisphere®.

For additional information about managing your VNX:

- VNX Documentation on EMC Online Support
- Unisphere online help

Installing Management Applications on VNX includes instructions on launching the Unisphere software, and on installing the MMC snap-ins and the ADUC extensions.

Related information

Specific information related to the features and functionality described in this document is included in:

- *VNX Glossary*
- *Configuring and Managing Network High Availability on VNX*
- *Configuring VNX Naming Services*
- *Configuring Time Services on VNX*
- *Configuring Events and Notifications on VNX for File*
- *Configuring Virtual Data Movers on VNX*
- *EMC VNX Command Line Interface Reference for File*
- *Parameters Guide for VNX*
- *Managing Volumes and File Systems with VNX Automatic Volume Management*
- *Managing Volumes and File Systems for VNX Manually*
- VNX for File man pages
- *Using FTP and TFTP on VNX*

For general information on the Internet Protocol Version 6 (IPv6), refer to:

- RFC 2460, *IPv6*
- RFC 2461, *Neighbor Discovery for IPv6*
- RFC 2463, *Internet Control Message Protocol (ICMPv6) for the IPv6 Specification*
- RFC 2464, *Transmission of IPv6 Packets over Ethernet Networks*
- RFC 4120, *The Kerberos Network Authentication Service (V5)*
- RFC 4291, *IPv6 Addressing Architecture*

EMC VNX documentation on EMC Online Support

The complete set of EMC VNX series customer publications is available on EMC Online Support. To search for technical documentation, go to http://Support.EMC.com. After logging in to the website, click **Support by Product** and type **VNX series** in the Find a Product text box. Then search for the specific feature required.

VNX wizards

Unisphere software provides wizards for performing setup and configuration tasks. The Unisphere online help provides more details on the wizards.

2 Concepts

VNX acts as a link between users located on an IP network and the stored data they want to access. It supports high-speed Ethernet connections and requires an understanding of basic networking concepts such as:

- Devices and interfaces — The configuration of the physical ports on NICs and the IP interfaces associated with them
- Routing — The configuration of directly connected, static, and dynamic routes
- Packet Reflect — The reflection of outbound (reply) packets through the same interface that inbound (request) packets entered
- VLANs — The configuration of logical networks functioning independently of the physical network configuration
- ARP — The association of IP addresses and MAC addresses

VNX provides network high-availability features such as Fail-Safe Network (FSN) devices, Ethernet channels, and link aggregations to help reduce the risk of link failures. *Configuring and Managing Network High Availability on VNX* provides more information.

Topics included are:

- VNX networking on page 15
- Devices and interfaces on page 15
- Routing on page 16
- Packet Reflect on page 17
- VLANs on page 19
- Address resolution on page 20
- TCP performance on page 20
- Differences between IPv4 and IPv6 on page 21
- IPv6 addresses on page 21
- Automatic default router configuration on page 24
- Neighbor discovery on page 25

- Duplicate Address Detection on page 25
- IPv4 and IPv6 addresses supported on the same device on page 26

VNX networking

To configure and manage networking on VNX you need to understand the devices and interfaces, routing, Packet Reflect, and VLANs.

VNX supports both IPv4 and IPv6 simultaneously by means of a dual IPv4/IPv6 protocol stack. Applications that support either IPv4 or IPv6 can run on the same network.

Devices and interfaces

VNX Clustered Network Server and the integrated and gateway NAS systems support several different types of NICs. Each VNX model supports multiple NIC combination in I/O modules:

- Four-port 10/100/1000Base-T (twisted-pair)
- Two-port 10 GbE (fiber-optic)
- Two-port 1Gb (twisted-pair) plus two-port 1 Gb (fiber-optic)

Data Movers on a VNX7500 and VNX5500 systems support one, two, or three of the following network I/O modules in any combination:

- Two-port 10 Gbe (fiber-optic) or active Twinax[1]
- Four-port 10/100/1000Base-T (twisted-pair)
- Two-port 1Gb (twisted-pair) plus two-port 1 Gb (fiber-optic)

For each physical port on a NIC, the Data Mover software defines a network device. The device name consists of a device prefix, representing the type of NIC, and a number that is sequentially appended, starting with 0 (zero), for each port on the NIC. For example, in the four-port 10/100/1000Base-T I/O module, the first copper Ethernet port is named cge0 with the remaining ports named cge1, cge2, and cge3, respectively.

Table 2 on page 15 lists the various network cards used in VNX with their respective cabling options and device prefixes.

Table 2. VNX NICs

NIC	Cabling	Device prefix
Four-port 10/100/1000Base-T	Twisted-pair	cge
Two-port 10 Gb Ethernet port	Fiber-optic	fxg
Two-port 1Gb (twisted-pair) plus two-port 1 Gb (fiber-optic)	Twisted-pair and fiber-optic	Two cge's plus two fxg's

[1] Twinax is a type of cable similar to coax, but with two inner conductors instead of one. The two-port 10-Gbe and FCoE I/O modules can also use active twinaxial (Twinax) cables.

Each network device can have one or more IP interfaces associated with it. The IP interfaces define the IP addresses that the device presents to the network.

Routing

When the Data Mover initiates network traffic, it uses its own routing table, which keeps track of how to reach destinations on the network. The Data Mover routing table can contain three types of routes:

- Directly connected — The network is directly connected to the Data Mover by an IP interface.
- Static — A route whose information is entered manually into the routing table by a network administrator and which takes priority over dynamic routing protocols.
- Dynamic — The routing table is managed automatically by the routed daemon on the Data Mover. The routed daemon listens for RIP V1 and V2 messages on the network and changes the routing table based on the messages.

  By default, all IPv4 interfaces on the Data Movers process RIP messages. Support for automatic routing daemons/protocols is not provided for IPv6 in this release.

  Note: The routed daemon provides minimal RIP V2 support, parsing only the next hop and mask fields. RIP V2 updates must be sent by broadcast.

In addition to supporting the different route types, the route table also supports the mask (or IPv6 prefix length) field, whether it is a default mask based on the class of the IP address or a variable length subnet mask (VLSM). VLSMs add flexibility, allowing different subnet masks of different lengths. VLSMs allow for better use of the available address space by allocating the appropriate number of host addresses per subnet and reducing the number of unused addresses in each subnet.

If you do not specify a mask, the default mask is assigned based upon the class of the IP address. Only contiguous masks are supported, and masks of 0.0.0.0 or 255.255.255.255 are invalid for network routes.

The routing table functions as follows:

- The route table contains only one route per destination subnet or destination host.
- Directly connected routes (interface routes) are considered permanent routes that cannot be updated manually or dynamically.
- Static routes can be updated manually by the user or by changes in the interface state, but are not modified by dynamic routing protocols.
- For IPv4 interfaces, if RIP advertises multiple routes to the same destination subnet or destination host, the route with the lowest cost (smallest hop count) is used. If the hop count is the same, the first route entered into the routing table for the destination remains in the table.

- IPv6 allows multiple default routes and it automatically chooses/manages a 'selected default' from the multiple defaults. It also learns about these default routes automatically from directly attached IPv6-capable routers by using the Router-discovery protocol.
- Dynamic routing protocols are currently not supported for IPv6.
- Routes might be added by the ICMP/ICMP6 redirects.
- The route table supports longest prefix matching. During a route lookup, if an exact match is unavailable, the routing table uses the closest matching address rather than an exact match.

Packet Reflect

Packet Reflect ensures that outbound (reply) packets always exit through the same interface that inbound (request) packets entered. Because the majority of network traffic on a Data Mover (including all file system I/O) is client-initiated, the Data Mover uses Packet Reflect to reply to client requests. With Packet Reflect, there is no need to determine the route to send the reply packets.

Packet Reflect works as follows:

- When Packet Reflect is enabled, the Data Mover does not examine the route, Neighbor Cache, and ARP tables when sending outbound packets from established connections. The Data Mover uses the same interface and the same next-hop MAC address as the request.

  Note: IPv6 does not use ARP tables, it uses Neighbor Cache instead.

- When Packet Reflect is disabled, the Data Mover examines the route and ARP tables when sending outbound packets from established connections.
- By default, Packet Reflect is enabled.

Packet Reflect mode applies to all interfaces and is useful in many situations, including:

- Supporting VLANs on VNX

  This ensures that the reply packets are always tagged with the same VLAN ID as incoming request packets. Packets coming in on one VLAN always go out on that VLAN. If the incoming request arrives without a tag, the outbound reply is sent without a tag.

- Providing better network security

  Packet Reflect can provide users with an additional security level. Because reply packets always go out the same interface as the request packets, request packets cannot be used to indirectly flood other LANs. In cases where two network devices exist, one connected to the Internet and the other connected to the intranet, replies to Internet requests do not appear on the intranet. Also, the internal networks used by VNX are not affected by any packet from external networks.

- Supporting multiple subnets each on a different NIC

In this case, each subnet uses a router, and the router port for each subnet filters incoming packets, so only packets from that subnet are forwarded. Replies, therefore, must be sent through the same interface as the incoming requests. Packet Reflect satisfies this requirement.

- Replacing multiple default routers

  In some environments, multiple default routers are used so that requests and replies are load balanced between available routers. VNX does not support multiple default routers for the IPv4 interfaces.

  Packet Reflect provides a similar functionality without the setup of additional default routes in the server. Packet Reflect performs this by reflecting replies back through the same router that forwarded the requests.

- Speeding up router failover

  In networking environments where multiple routers are located on a single LAN segment, Packet Reflect helps in quick failover to a secondary router.

  For example, when a Data Mover needs to reply to a request packet, Packet Reflect allows the Data Mover to avoid using a routing table to determine the interface and the next-hop MAC address. Instead, to send the reply, it uses the same interface and the same next-hop MAC address as the request.

  With Packet Reflect, replies failover faster to the backup router. The backup router does not need to wait until the routing table entries are updated by the RIP. If all outbound packets are replies to requests, RIP might not be needed. However, server-initiated outbound packets (NIS and DNS inquiries) still require that the server receive RIP messages to update routing tables.

- Helping clients with a single IP address and multiple MAC addresses

  Problems arise when the IP address of a Data Mover's client maps into multiple MAC addresses. Although unusual, this can happen when a client is in the same LAN segment as the Data Mover on one port and simultaneously, the client appears behind a proxy ARP server from a different port. This creates a problem for the server if Packet Reflect is not enabled. For each IP address, the Data Mover keeps only one associated MAC address in the ARP table. With Packet Reflect enabled, this problem is resolved because the server does not need to look up the MAC address from the ARP database for the reply. Instead, to send the reply, the server uses the MAC address of the request.

- Supporting high-availability devices

  Normally, Packet Reflect determines the interface for the reply packets. However, if the interface is a high-availability device, such as Ethernet channel or link aggregation, VNX load-balancing algorithm determines the specific link or port within the high-availability device for the outgoing packets. Incoming and outgoing packets do not need to be on the same link. Moreover, switches might have a number of load-balancing algorithms to choose from (IP, MAC, or session); so the switch determines the incoming link. Therefore, the load-balancing algorithms are deterministic, preventing out-of-order packet delivery in a single direction. *Configuring and Managing Network High Availability on VNX* provides information about high-availability devices.

VLANs

VLANs are logical networks that function independently of the physical network configuration. For example, VLANs enable you to put all of a department's computers on the same logical subnet, which can increase security and reduce network broadcast traffic.

VLANs are especially useful when configuring standby Data Movers. Typically, there is one standby Data Mover for multiple production Data Movers. To take over for any production Data Mover, the standby needs to assume the network identity of that production Data Mover. Because different Data Movers often service different subnets, the standby will need to connect to all the subnets it might serve. Using VLANs, the production and standby Data Movers can be physically connected to the same few switches, and use VLAN tagging to connect to the appropriate individual subnets they serve.

VNX support IEEE 802.1Q VLAN tagging on network devices defined by the device prefix ace, cge, fge, and fxg. VLAN tagging is unsupported on network devices defined by the prefix ana.

Note: Before establishing VLANs on Data Movers, ensure basic connectivity within the environment. Review the Data Mover port configuration and the switch port configuration, verifying that speed and duplex values are consistent and equal end to end. Any unmatched values between Data Movers and switches cause unique errors to appear in the overall performance of the Data Mover and possibly the network as a whole.

Using VLANs in a VNX environment

When a single NIC is assigned multiple logical interfaces, a different VLAN can be assigned to each interface. When each interface has a different VLAN, a packet is accepted only if its destination IP address is the same as the IP address of the interface, and the packet's VLAN tag is the same as the interface's VLAN ID. If the VLAN ID of an interface is set to zero, packets are sent without tags.

VLAN tagging

A VLAN-tagged frame carries an explicit identification of the VLAN to which it belongs. It carries this nonnull VLAN ID within the frame header. The tagging mechanism implies a frame modification. For IEEE 802.1Q-compliant switches, the frame is modified according to the port type used.

VLANs establish multiple broadcast domains through switch port association, MAC addresses, or network layer parameters (such as IP subnets). This association allows Data

Movers to understand and process the VLAN tags on messages coming from a network switch.

There are two ways to work with VLANs:

- Configure a switch port with a VLAN identifier and connect a VNX port to that switch port. This is the way clients are typically configured for VLANs. The file server is unaware that it is part of the VLAN, and no special configuration of the file server is needed. The VLAN ID is set to zero.
- Make the server responsible for interpreting the VLAN tags and processing the packets appropriately. This enables the server to connect to multiple VLANs and their corresponding subnets through a single physical connection. In this method, the switch ports for servers, including VNX, are configured to include VLAN tags on packets sent to the server.

Address resolution

You can configure the ARP tables on each of the Data Movers for all the IPv4 interfaces. The ARP table defines the relationship between IPv4 addresses and physical Ethernet addresses (MAC addresses) for network components.

IPv6 interfaces does not support ARP tables. Neighbor Discovery defines the relationship between IPv4 addresses and physical Ethernet addresses.

TCP performance

VNX provides two parameters that enhance TCP performance associated with lost TCP segments. These parameters enable or disable the NewReno, Selective Acknowledgement (SACK), and Forward Acknowledgement (FACK) algorithms:

- tcp do_newreno
- tcp do_sack

The do_newreno and do_sack parameters address TCP throughput issues when multiple segments are dropped. The difference between the two mechanisms controlled by these parameters is that SACK exchanges explicit missing-packet information between receiver and transmitter and allows for more efficient retransmission, but it requires that both the transmitter and receiver support the SACK protocol. NewReno provides for a slightly less efficient recovery, but demands no specific protocol support from the receiver.

FACK algorithm is closely associated with SACK and is used to manage traffic flow during TCP congestion. Because FACK relies on SACK, it is enabled and disabled automatically along with the do_sack parameter.

The NewReno algorithm can operate simultaneously with SACK or function as a stand-alone algorithm to optimize the retransmission of dropped TCP packets. Connections where SACK is disabled, either because the do_sack parameter is disabled or the peer does not allow SACK, fall back to using the NewReno algorithm, if it is enabled by means of the do_newreno parameter. The do_newreno parameter then provides enhanced dropped-packet recovery for the TCP transmitter. These two parameters are enabled by default.

EMC recommends that you accept the default for these two parameters, so that TCP can choose the most efficient algorithm. Improve TCP performance on page 100 describes the procedure to improve TCP performance.

Differences between IPv4 and IPv6

Table 3 on page 21 shows some common differences between IPv4 and IPv6.

Table 3. Differences between IPv4 and IPv6

Functionality	IPv4	IPv6
Addressing	32-bit addresses, dotted decimal	128-bit addresses, hex-based
Address resolution	Uses the Address Resolution Protocol (ARP), which broadcasts to all devices on the link.	Uses Neighbor Discovery (ND) address resolution, which uses multicast groups.
Subnet masks	Yes	No. Uses prefix length notation instead.
Broadcast addresses	Yes	No. Uses multicast addresses instead.
Minimum MTU	576 bytes	1280 bytes
Fragmentation	Sender/router	Sender
Forward DNS	A record	AAAA record
Reverse DNS	in-addr.arpa	ip6.arpa

IPv6 addresses

IPv6 solves the problem of address depletion in IPv4. IPv6 uses a 128-bit address in contrast to the 32-bit address used in IPv4. An IPv6 address is expressed as a hexadecimal value consisting of eight, 16-bit, colon-separated fields as in the address:

3ffe:0000:3c4d:0015:0000:0000:0300:00aa

An IPv6 global unicast address uses a global hierarchy made up of three components:

- Global routing prefix — Leftmost (high-order) 48 bits. The global routing prefix defines the network portion of the address and is used for routing.
- Subnet ID — Next 16 bits. The subnet ID defines your site’s topology.
- Interface ID — Rightmost 64 bits. This is the interface identifier.

The components of the preceding IPv6 address are:

Global routing prefix	Subnet ID	Interface ID
3ffe:0000:3c4d:	0015:	0000:0000:0300:00aa

Abbreviating an IPv6 address

You can drop leading zeros in an IPv6 address. For example, you can abbreviate the address 3ffe:0000:3c4d:0015:0000:0000:0300:00aa by dropping the leading zeros as 3ffe:0:3c4d:15:0:0:300:aa. You can further abbreviate an address by replacing one or more sets of zeros with a double colon as 3ffe:0:3c4d:15::300:aa. You can use the double colon only once per address. Address representations are not case-sensitive.

When you configure an IPv6 address, the Data Mover parses the address into the 128-bit format and saves it. When that IPv6 address is displayed by the Data Mover, it can be displayed in any one of the abbreviated formats defined by the IPv6 protocol. In other words, the saved address value is converted into different presentation strings for display. These presentation strings are not saved. For example, if you use the following command to configure an IP interface with an IPv6 address and then display the results, you will see two forms of the same address. In the displayed IPv6 address, the leading zeros have been dropped:

```
$ server_ifconfig server_2 -create -Device cge0 -name cge0_int1 -protocol
IP6 3ffe:0000:3c4d:0015:0435:0200:0300:ED20
```

```
$ server_ifconfig server_2 cge0_int1
```

```
cge0_int1 protocol=IP6 device=cge0
inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2c:f6
```

The displayed address could also have appeared as 3ffe::3c4d:15:435:200:300:ed20, where the double colon replaces the zero in the second 16-bit field.

Likewise, if you use an abbreviated address, a different abbreviated address might be displayed:

```
$ server_ifconfig server_2 -create -Device cge0 -name cge0_int2 -protocol
IP6 3ffe::3c4d:15:435:200:300:ed40
```

```
$ server_ifconfig server_2 cge0_int2
```

```
cge0_int2 protocol=IP6 device=cge0
inet=3ffe:0:3c4d:15:435:200:300:ed40 prefix=64
UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2c:f6
```

If you use an IPv6 address with a dotted decimal representation, the Data Mover will display the address in its hex-equivalent format:

c

```
$ server_ifconfig server_2 -create -Device cge0 -name cge0_int3 -protocol
IP6 3ffe::128.22.11.10
```

```
$ server_ifconfig server_2 cge0_int3

cge0_int3 protocol=IP6 device=cge0
inet=3ffe::8016:b0a prefix=64
UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2c:f6
```

IPv6 prefixes

IPv6 uses prefixes, as in Classless Inter-Domain Routing (CIDR), to replace the use of subnet masks. IPv6 does not support network masks. You can write a network address as 3ffe:0:3c4d:15::300:aa/64. The default network prefix is 64.

Address types

IPv6 supports a number of different types of addresses. The following section summarizes those relevant when configuring and managing VNX:

- Unicast addresses

The following unicast addresses are the addresses most relevant to VNX. They identify a single interface.

- Global aggregatable addresses

Global aggregatable addresses are globally unique addresses on the Internet.

- Link-local addresses

Link-local addresses are used primarily for administrative purposes, such as address resolution and duplicate address detection. For example, the Neighbor Discovery process uses link-local addresses to locate neighboring nodes on the link. Link-local traffic is never forwarded by an IPv6 router beyond the attached link; link-local scope is limited to the link or broadcast domain.

Link-local addresses are configured automatically when you configure a global address. Similarly, they are removed when you delete a global address. If you manually configure a link-local address first, then, when you create a global address, another link-local address is not created. Appendix D provides more information about manually configuring a link-local address.

The first interface on an IPv6 link (whether created automatically or manually) will be a link-local address (identified by the device name, four digit VLAN ID, and two letter els for link-local), and it will be designated as the default link-local interface. Any subsequent IPv6 interface (whether link-local or global) created on the defined logical link will be dependent on the default link-local interface for network administration. Because of this dependency, you cannot delete the default link-local interface.

Link-local interfaces are reserved for core IPv6 protocols (such as Duplicate Address Detection, Neighbor Discovery, Router Discovery, and ICMP6). The server_ping6 will support link-local interface ping operations for diagnosing/testing network connectivity. However, other applications should not be configured to use link-local addresses for client/server communication. Additionally, Unisphere will suppress the display of any/all link-local addresses.

Link-local addresses always begin with the prefix fe80::/64, with the form fe80::xxxx:yyyy:zzzz:wwww

These address can be used only on a single, local link. They are not recognized beyond the local link; that is, they are not routable. Any IPv6 address beginning with fe80::/10 is always a link-local address. A link-local address permits communications between neighboring devices on a local link such as an Ethernet broadcast domain.

Link-local addresses are required only to be unique between neighbors communicating on the same link. Therefore, a node can have the same link-local address for all of its ports. The use of a link-local or multicast destination requires the specification of a '%<intf>' scope identifier. For example: fe80::1:2:3:4%fxg-1-0_v6intf

Other than during creation of link-local addresses, anytime a link-local address is used without the interface name, an error message appears.

Packets with destination addresses beyond their scope are not forwarded.

- Special addresses

Some commonly used special addresses with their compressed notation are:

- Loopback address — ::1/128
- Link-local address — fe80::/10
- Multicast address — ff00::/8
- Unspecified address — ::/128

Multicast addresses provide communications from one interface to many interfaces.

Note: IPv6 does not support broadcast addresses. A broadcast is a specific type of multicast where packets are sent to the all-nodes multicast with hop count of 1.

- Anycast addresses

Anycast addresses provide communications to one interface out of many interfaces, and the interface is usually the closest one associated with the address as determined by routing distance.

Automatic default router configuration

In IPv6, routers advertise their presence and the prefixes they support. Router Discovery enables the Data Mover to listen for these advertisements on the attached link and update its default router list based on that information. If the default router becomes unreachable, the Data Mover selects another router from the list.

Neighbor discovery

IPv6 hosts and routers use Neighbor Discovery (ND) to determine the link-layer addresses for neighboring nodes on attached links. Hosts also use ND to locate neighboring routers that can forward packets.

Neighbor cache

The neighbor cache is an address resolution database created and maintained by the ND process by means of Neighbor Advertisements and Neighbor solicitations. Like IPv4's Address Resolution Protocol (ARP), the ND resolves a neighboring node's IPv6 address to its link-layer address (MAC address) and link-layer address to IPv6 address. ND automatically updates the Neighbor cache. You can also manually manage the Neighbor cache.

The neighbor cache maintains a list of neighbors by their global unicast address, link-layer address, whether the neighbor is a router or host, and the state of the neighbor entry, such as if the neighbor is reachable.

Routing table

The routing table determines the next hop when a Data Mover is sending a packet. ND maintains the routing table database and automatically fills in the table by listening to the router advertisements. You can also manually managed it. The routing table can contain two types of route entries:

- Prefix entry which specifies the interface through which to reach a specific prefix.
- Default entry which specifies the gateway to use as default when none of the prefixes match the destination.

The routing table also maintains the name of the interface used for the route and information about when the route entry expires.

Duplicate Address Detection

By default each time you boot an IPv6 system or configure a new interface, Neighbor Discovery uses the DAD process to determine if the address on the link is a duplicate address. For each new address, the host multicasts two neighbor solicitations messages to all nodes on the link to verify that the new address is unique. If during the 600 milliseconds since the first neighbor solicitation, the host receives a neighbor advertisement for this address, the address is marked as a duplicate, and you must change one of the two addresses to eliminate the duplication. If after 600 milliseconds, the host receives no response to the neighbor solicitations, a neighbor advertisement is sent to the all-nodes multicast address declaring that the host owns the new address.

IPv4 and IPv6 addresses supported on the same device

VNX provides dual-stack support for both IPv4 and IPv6 addresses on the same device. That is, you can configure both address types on the same device.

3

Configuring

The tasks to configure networking are:

- Configure network devices on page 28
- Configure single or multiple IP interfaces on page 31
- Configure interface options on page 45
- Verify Data Mover connectivity on page 49
- Verify Data Mover connectivity for an IPv6 address on page 51
- Configure routing on page 52

Configure network devices

To configure network devices:

- Network device availability on page 28
- Modify network device settings on page 29

CAUTION Mismatched settings for speed and duplex settings can cause errors and affect network performance. Ensure that speed and duplex settings are equal and consistent on both ends of the physical connection. For example, duplex settings of half on one end and full on the other end of the physical connection can cause network errors and performance issues. Similarly, speed settings of 100 on one end and 1000 on the other end might cause problems.

Network device availability

Use the server_sysconfig command to determine which network devices are available. The server_sysconfig command displays information for all devices.

In the example, the network devices are in slots 3 and 4 of server_2. The snmpwalk -v1 -c public server_# | grep ifOperStatus command provides the physical port status of the network ports on VNX.

Action

To determine which devices and options are available on a Data Mover, use this command syntax:
`$ server_sysconfig <movername> -pci`
where:
<movername> = name of the Data Mover
Example:
To determine which devices and options are available for server_2, type:
`$ server_sysconfig server_2 -pci`

Output

```
server_2 : PCI DEVICES:

On Board:
  Agilent Fibre Channel Controller
    0:  fcp-0  IRQ: 22 addr: 50060160306008be

    0:  fcp-1  IRQ: 21 addr: 50060161306008be

    0:  fcp-2  IRQ: 18 addr: 50060162306008be

    0:  fcp-3  IRQ: 20 addr: 50060163306008be

  Broadcom Gigabit Ethernet Controller
    0:  cge0  IRQ: 23
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge1  IRQ: 25
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge2  IRQ: 24
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge3  IRQ: 26
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge4  IRQ: 27
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge5  IRQ: 28
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable
```

Note

The devices cge0 through cge5 are available and each device has the settings you can modify: speed, duplex, and transmit and receive flow control.

Modify network device settings

Considerations to modify network device settings:

- Options in the command must be enclosed in double quotes.
- Ensure that the IP network switch settings match the device settings.
- The speed and duplex cannot be modified for a fiber connection (ace, fge, and fxg).
- Unless you have specific network configuration requirements, leave the default settings in place.

The *EMC VNX Command Line Interface Reference for File* entry for server_sysconfig contains a detailed description of the available options and default settings for each NIC type.

Action
To modify a network device setting, use this command syntax: `$ server_sysconfig <movername> -pci <device> -option "speed=10|100|1000|auto, duplex=full|half|auto"` where: *`<movername>`* = name of the Data Mover *`<device>`* = network adapter card installed in the Data Mover -option speed=*`{10|100|1000|auto}`* = speed is automatically detected from the Ethernet line and turns on autonegotiation -option duplex=*`{full|half|auto}`* = auto (default) turns autonegotiation on Example: To modify the speed for device cge0 to 100 and the duplex to full on server_2, type: **`$ server_sysconfig server_2 -pci cge0 -option "speed=100,duplex=full"`**
Output
`server_2 : done`

Verify network device settings

Action
To verify the device settings you modified, use this command syntax: `$ server_sysconfig <movername> -pci` where: *`<movername>`* = name of the Data Mover Example: To verify the device settings you modified on server_2, type: **`$ server_sysconfig server_2 -pci`**

Output

```
server_2 : PCI DEVICES:

On Board:
    0:  fcp-0  IRQ: 22 addr: 50060160306008be
  Agilent Fibre Channel Controller
    0:  fcp-1  IRQ: 21 addr: 50060161306008be

    0:  fcp-2  IRQ: 18 addr: 50060162306008be

    0:  fcp-3  IRQ: 20 addr: 50060163306008be

  Broadcom Gigabit Ethernet Controller
    0:  cge0  IRQ: 23
    speed=100 duplex=full txflowctl=disable rxflowctl=disable

    0:  cge1  IRQ: 25
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge2  IRQ: 24
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge3  IRQ: 26
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge4  IRQ: 27
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable

    0:  cge5  IRQ: 28
    speed=auto duplex=auto txflowctl=disable rxflowctl=disable
```

Configure single or multiple IP interfaces

You have the option of configuring a single IP interface or multiple IP interfaces on a single device. If you are configuring a virtual device such as an Ethernet channel, a link aggregation, or FSN, you must create the virtual device before configuring the interface. *Configuring and Managing Network High Availability on VNX* provides information about configuring virtual devices.

Configuring an IP interface consists of one of these tasks:

- Configure an IPv4 interface on page 32
- Configure a single device on multiple networks on page 34
- Configure a single device on the same network on page 36
- Configure an IPv6 interface on page 37
- Configure an IPv6 interface with a nondefault prefix length on page 38
- Configure IPv6 and IPv4 interfaces on the same device on page 39
- Verify connectivity on page 41

Configure an IPv4 interface

Considerations to configure an IP interface:

- VNX supports different types of subnet masks.
- VNX does not support noncontiguous network masks. Noncontiguous network masks are masks without a continuous stream of 1 bit.
- If a physical device is used by a virtual device and you assign a new name to the interface, you will receive the error: server_2 : No such device or address.
- Use the server_sysconfig command to view information about the device.

Action
To create an IP interface for a network device, use this command syntax: `$ server_ifconfig <movername> -create -Device <device_name> -name <if_name> -protocol IP <ipv4_addr> <ipmask> <ipbroadcast>` where: *`<movername>`* = name of the Data Mover where the device is located *`<device_name>`* = name of the network device *`<if_name>`* = name for the interface created by this command *`<ipv4_addr>`* = IP address for the interface *`<ipmask>`* = network mask for the interface *`<ipbroadcast>`* = broadcast address for the interface Example: To configure an interface for network devices cge0, cge1, and cge4 on server_2, type: **`$ server_ifconfig server_2 -create -Device cge0 -name cge0 -protocol IP`** **`172.24.108.10 255.255.255.0 172.24.108.255`** **`$ server_ifconfig server_2 -create -Device cge1 -name cge1 -protocol IP`** **`172.24.101.10 255.255.255.0 172.24.101.255`** **`$ server_ifconfig server_2 -create -Device cge4 -name cge4 -protocol IP`** **`172.24.106.10 255.255.255.128 172.24.106.127`**
Output
`server_2 : done`

Verify an IPv4 interface

To view settings of all the IPv4 interfaces, use the -ip4 option.

Action
To verify the settings for the IPv4 interfaces, use this command syntax: `$ server_ifconfig <movername> -all -ip4` where: `<movername>` = name of the Data Mover Example: To verify the settings for the IPv4 interfaces on server_2, type: **`$ server_ifconfig server_2 -all -ip4`**

Output

```
server_2 :
cge4 protocol=IP device=cge4
     inet=172.24.106.10 netmask=255.255.255.128 broadcast=172.24.106.127
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:98
cge3_1 protocol=IP device=cge3
     inet=172.24.104.10 netmask=255.255.255.0 broadcast=172.24.104.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:9a
cge2_2 protocol=IP device=cge2
     inet=172.24.103.10 netmask=255.255.255.0 broadcast=172.24.103.255
     UP, ethernet, mtu=1500, vlan=103, macaddr=8:0:1b:42:86:95
cge2_1 protocol=IP device=cge2
     inet=172.24.102.10 netmask=255.255.255.0 broadcast=172.24.102.255
     UP, ethernet, mtu=1500, vlan=102, macaddr=8:0:1b:42:86:95
cge1 protocol=IP device=cge1
     inet=172.24.101.10 netmask=255.255.255.0 broadcast=172.24.101.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:97
```

Verify the settings for all interfaces

The server_ifconfig server_# -all does list all the ipv4 and ipv6 interfaces. To view just ipv4 or just ipv6 use the -ip4 or -ip6 options.

Action
To verify the settings for all interfaces on a Data Mover, use this command syntax: `$ server_ifconfig <movername> -all` where: `<movername>` = name of the Data Mover Example: To verify the settings for all interfaces on server_2, type: **`$ server_ifconfig server_2 -all`**

Output

```
server_2 :
cge4 protocol=IP device=cge4
     inet=172.24.106.10 netmask=255.255.255.128 broadcast=172.24.106.127
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:98
cge3_1 protocol=IP device=cge3
     inet=172.24.104.10 netmask=255.255.255.0 broadcast=172.24.104.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:9a
cge2_2 protocol=IP device=cge2
     inet=172.24.103.10 netmask=255.255.255.0 broadcast=172.24.103.255
     UP, ethernet, mtu=1500, vlan=103, macaddr=8:0:1b:42:86:95
cge2_1 protocol=IP device=cge2
     inet=172.24.102.10 netmask=255.255.255.0 broadcast=172.24.102.255
     UP, ethernet, mtu=1500, vlan=102, macaddr=8:0:1b:42:86:95
cge1 protocol=IP device=cge1
     inet=172.24.101.10 netmask=255.255.255.0 broadcast=172.24.101.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:97
cge0 protocol=IP device=cge0
     inet=172.24.108.12 netmask=255.255.255.0 broadcast=172.24.108.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:96
loop protocol=IP device=loop
     inet=127.0.0.1 netmask=255.0.0.0 broadcast=127.255.255.255
     UP, loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0
     netname=localhost
el31 protocol=IP device=fxp0
     inet=128.221.253.2 netmask=255.255.255.0 broadcast=128.221.253.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:43:91:2
     netname=localhost
el30 protocol=IP device=fxp0
     inet=128.221.252.2 netmask=255.255.255.0 broadcast=128.221.252.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:43:91:2
     netname=localhost
cge0_int1 protocol=IP6 device=cge0
       inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
       UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
cge0_0000_ll protocol=IP6 device=cge0
       inet=fe80::260:16ff:fe0c:205 prefix=64
       UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
```

Configure a single device on multiple networks

When creating more than one interface, specify unique names, so that multiple interfaces can access the same device. For each new interface, configure a unique IP address, a subnet mask, and a broadcast address.

Action

To configure multiple interfaces for a network device, use this command syntax:

$ **server_ifconfig** *<movername>* **-create -Device** *<device_name>* **-name** *<if_name>* **s***<ipaddr> <ipmask> <ipbroadcast>*

where:

<movername> = name of the Data Mover where the device is located

Action
<device_name> = name of the network device *<if_name>* = name for the interface created by this command *<ipaddr>* = IP address for the interface *<ipmask>* = network mask for the interface *<ipbroadcast>* = broadcast address for the interface Example: To configure interfaces cge2_1 and cge2_2 for a network device on server_2, type: **$ server_ifconfig server_2 -create -Device cge2 -name cge2_1 -protocol IP 172.24.102.10 255.255.255.0 172.24.102.255** **$ server_ifconfig server_2 -create -Device cge2 -name cge2_2 -protocol IP 172.24.103.10 255.255.255.0 172.24.103.255**
Output
`server_2 : done`

Verify the configuration settings on multiple networks

Action
To verify the settings for a network device, use this command syntax: $ **server_ifconfig** *<movername> <if_name>* where: *<movername>* = name of the Data Mover *<if_name>* = name of the interface Example: To verify the settings for the cge2_1 interface on server_2, type: **$ server_ifconfig server_2 cge2_1**
Output

```
server_2 :
cge2_1 protocol=IP device=cge2
       inet=172.24.102.10 netmask=255.255.255.0 broadcast=172.24.102.255
       UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:95
```

After configuring a single device on multiple networks

After you configure a single device on multiple networks, you can assign a different VLAN tag to each interface. Enable VLAN tagging on page 45 provides more information.

Configure a single device on the same network

You can configure a single device on the same network by assigning multiple addresses in the same network to the device. One of the interfaces is the default. If this interface goes down, another interface on the same network becomes the default.

When creating multiple interfaces, specify unique names, so each interface can access the same device. For each new interface, configure a unique IP address on the same network, a subnet mask, and a broadcast address.

Action

To configure multiple interfaces on the same network, use this command syntax:
`$ server_ifconfig <movername> -create -Device <device_name> -name <if_name> -protocol IP <ipaddr> <ipmask> <ipbroadcast>`
where:
`<movername>` = name of the Data Mover where the device is located
`<device_name>` = name of the network device
`<if_name>` = name for the interface created by this command
`<ipaddr>` = IP address for the interface
`<ipmask>` = network mask for the interface
`<ipbroadcast>` = broadcast address for the interface
Example:
To configure interfaces cge3_1 and cge3_2 on the same network on server_2, type:
`$ server_ifconfig server_2 -create -Device cge3 -name cge3_1 -protocol IP`
`172.24.104.10 255.255.255.0 172.24.104.255`
`$ server_ifconfig server_2 -create -Device cge3 -name cge3_2 -protocol IP`
`172.24.104.12 255.255.255.0 172.24.104.255`

Output
`server_2 : done`

Verify the configuration settings on the same network

Action
To verify the settings for a network device, use this command syntax: `$ server_ifconfig <movername> <if_name>` where: `<movername>` = name of the Data Mover

Action
<if_name> = name of the interface Example: To verify the settings for the cge3_1 interface on server_2, type: **`$ server_ifconfig server_2 cge3_1`**
Output

`server_2 :`
`cge3_1 protocol=IP device=cge2`
`       inet=172.24.104.10 netmask=255.255.255.0 broadcast=172.24.104.255`
`       UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:9a`

Configure an IPv6 interface

Action
To create an IP interface with an IPv6 address for a network device, use this command syntax: `$` **`server_ifconfig`** *`<movername>`* **`-create -Device`** *`<device_name>`* **`-name`** *`<if_name>`* **`-protocol IP6`** *`<ipv6_addr>[/PrefixLength]`* where: *<movername>* = name of the Data Mover where the device is located *<device_name>* = network device name *<if_name>* = name for the interface created by this command *<ipv6_addr>* = IP address for the interface *[/PrefixLength]* = is a decimal value ranging from 1 to 128 that indicates the number of contiguous, higher-order bits of the address that make up the network portion of the address (optional) IPv6 does not use a mask or broadcast address. Example: To configure an interface for network device cge0 with an IPv6 address on server_2, type: **`$ server_ifconfig server_2 -create -Device cge0 -name cge0_int1 -protocol IP6 3ffe:0000:3c4d:0015:0435:0200:0300:ED20`**
Output
`server_2 : done`

Verify an IPv6 interface

To view settings of all the IPv6 interfaces, use the -ip6 option.

Action
To verify the settings for the IPv6 interfaces on a Data Mover, use this command syntax: `$ server_ifconfig <movername> -all -ip6` where: *`<movername>`* = name of the Data Mover on which to verify the interfaces Example: To verify the settings for IPv6 interfaces on server_2, type: **`$ server_ifconfig server_2 -all -ip6`**
Output

```
server_2 :
cge0_int1 protocol=IP6 device=cge0
       inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
       UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
cge0_0000_ll protocol=IP6 device=cge0
       inet=fe80::260:16ff:fe0c:205 prefix=64
       UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
```

Note
The link-local name and address are automatically generated when you configure a global address for cge0. The name is made up of the device name, a four digit VLAN ID between 0 and 4094, and two letter els to represent link local. Note that the interface you configured with the IPv6 address 3ffe:0:3c4d:15:435:200:300:ed20 and the address with the link-local address fe80::260:16ff:fe0c:205 share the same MAC address. The link-local address is derived from the MAC address.

Configure an IPv6 interface with a nondefault prefix length

Action
To create an IP interface with an IPv6 address and a nondefault prefix length for a network device, use this command syntax: `$ server_ifconfig <movername> -create -Device <device_name> -name <if_name> -protocol IP6 <ipv6_addr>` where: *`<movername>`* = name of the Data Mover where the device is located *`<device_name>`* = network device name *`<if_name>`* = name for the interface created by this command *`<ipv6_addr>`* = IP address for the interface Example: To configure an interface for network device cge0 with an IPv6 address with a nondefault prefix length on server_2, type:

Action
$ server_ifconfig server_2 -create -Device cge0 -name cge0_int1 -protocol IP6 3ffe:0000:3c4d:0015:0435:0200:0300:ED20/48
Output
`server_2 : done`

Verify an IPv6 interface with a nondefault prefix length

Action
To verify that the settings for an interface are correct, use this command syntax: $ **server_ifconfig** *<movername> <if_name>* where: *<movername>* = name of the Data Mover on which to verify the interfaces *<if_name>* = name of the interface Example: To verify that the settings for the cge0_int1 interface for server_2 are correct, type: **$ server_ifconfig server_2 cge0_int1**
Output

```
server_2 :
cge0_int1 protocol=IP6 device=cge0
         inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=48
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:5:5
```

Configure IPv6 and IPv4 interfaces on the same device

You can configure an IPv4 address on the device on which you configured the IPv6 address.

Action
To create an IPv4 interface for a network device, use this command syntax: $ **server_ifconfig** *<movername>* **-create -Device** *<device_name>* **-name** *<if_name>* **-protocol IP** *<ipv4_addr> <ipmask> <ipbroadcast>* where: *<movername>* = name of the Data Mover where the device is located *<device_name>* = network device name *<if_name>* = name for the interface created by this command

Action
<ipv4_addr> = IP address for the interface *<ipmask>* = network mask for the interface *<ipbroadcast>* = broadcast address for the interface Example: To configure an IPv4 interface for network device cge0 on server_2, type: **`$ server_ifconfig server_2 -create -Device cge0 -name cge0_int2 -protocol IP 172.24.108.10 255.255.255.0 172.24.108.255`**
Output
`server_2 : done`

Verify IPv4 and IPv6 interfaces on the same device

Action
To verify that the settings for all interfaces on a device are correct, use this command syntax: `$ server_ifconfig <movername> -all` where: *<movername>* = name of the Data Mover on which to verify the interfaces Example: To verify that the settings for server_2 are correct, type: **`$ server_ifconfig server_2 -all`**

Output

```
server_2 :
cge0_int2 protocol=IP device=cge0
        inet=172.24.108.10 netmask=255.255.255.0 broadcast=172.24.108.255

         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
cge0_int1 protocol=IP6 device=cge0
         inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
cge0_0000_ll protocol=IP6 device=cge0
         inet=fe80::260:16ff:fe0c:205 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
el30 protocol=IP device=mge0
        inet=128.221.252.2 netmask=255.255.255.0 broadcast=128.221.252.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:d:30:b1 netname=lo-
calhost
el31 protocol=IP device=mge1
        inet=128.221.253.2 netmask=255.255.255.0 broadcast=128.221.253.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:d:30:b2 netname=lo-
calhost
loop6 protocol=IP6 device=loop
         inet=::1 prefix=128
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
loop protocol=IP device=loop
         inet=127.0.0.1 netmask=255.0.0.0 broadcast=127.255.255.255

         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
```

Verify connectivity

After configuring the IP interfaces, verify connectivity for each interface you configured.

You can ping by using either the hostname or the IP address of the client for which you want to check connectivity with the Data Mover. Invoking ping with a hostname requires the local host file, NIS, or DNS server to be configured and operational on the Data Mover. If it is not, type the client's IP address.

When invoking server_ping6 with a hostname, the /.etc/hosts file on the Data Mover is checked first, then NIS and the DNS servers are checked, if configured and operational. If the name does not exist in any of these locations, an error message appears. You can change the search order by means of the nsswitch.conf file.

Configuring VNX Naming Services provides more information about name resolution, the local host file, NIS, and DNS.

Verify connectivity for an IPv4 interface

To verify connectivity after configuring the IPv4 interface:

- Verify connectivity between a Data Mover and a client on page 42
- Verify connectivity between a Data Mover and a client continuously on page 42
- Verify connectivity between a Data Mover and a client by using an IPv4 interface on page 43

Verify connectivity between a Data Mover and a client

Action
To verify connectivity of a Data Mover to the outside world, use this command syntax:
`$ server_ping <movername> <ipv4_addr>`
where:
`<movername>` = name of the Data Mover
`<ipv4_addr>` = IP address of the client to ping
Example:
To verify connectivity of a Data Mover to an IP address, type:
`$ server_ping server_2 172.24.102.2`
Output
`server_2 : 172.24.102.2 is alive, time= 0 ms`

Verify connectivity between a Data Mover and a client continuously

Action
To verify connectivity fr a Data Mover to the outside world while sending continuous ECHO_REQUEST messages, use this command syntax:
`$ server_ping <movername> -send <ipaddr>`
where:
`<movername>` = name of the Data Mover
`<ipaddr>` = IP address of the client to ping
Example:
To verify connectivity for a Data Mover to the outside world while sending continuous ECHO_REQUEST messages, type:
`$ server_ping server_2 -send 172.24.102.2`

Output

`server_2 :`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 3 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`
`172.24.102.2 is alive, time= 0 ms`

Verify connectivity between a Data Mover and a client by using an IPv4 interface

Action
To verify connectivity from a Data Mover to the client by using a specified interface, use this command syntax: `$` **`server_ping`** *`<movername>`* **`-interface`** *`<if_name>`* *`<ipaddr>`* where: *`<movername>`* = name of the Data Mover *`<if_name>`* = name of the interface *`<ipaddr>`* = IP address of the client to ping To display connectivity from a Data Mover to the client IP address by using a specified interface, type: **`$ server_ping server_2 -interface cge0 172.24.102.2`**
Output
`server_2 : 172.24.102.2 is alive, time= 0 ms`

Verify connectivity for an IPv6 interface

To verify connectivity after configuring the IPv6 interface:

- Verify connectivity between a Data Mover and a client by using an IPv6 interface on page 43
- Verify connectivity between a Data Mover and a link-local destination on page 44

Verify connectivity between a Data Mover and a client by using an IPv6 interface

Action
To verify connectivity of a Data Mover for an IPv6 interface, use this command syntax: `$` **`server_ping6`** *`<movername>`* **`-interface`** *`<if_name>`* *`<ipv6_addr>`*

Action

where:

<movername> = name of the Data Mover

<if_name> = name of the interface

<ipv6_addr> = IP address of the client to ping

Example:

To verify connectivity of server_2 for cge0_int1 interface, type:

```
$ server_ping6 server_2 -interface cge0_int1 3ffe:0000:3c4d:0015:0435:0200:0300:00aa
```

Output

```
server_2 : 3ffe:0000:3c4d:0015:0435:0200:0300:00aa is alive, time= 0 ms
```

Verify connectivity between a Data Mover and a link-local destination

Action

To verify connectivity between a Data Mover and a link-local destination, use this command syntax:

```
$ server_ping6 <movername> <ipv6_addr>
```

where:

<movername> = name of the Data Mover

<ipv6_addr> = IP address of a link-local destination to ping

Example:

To verify the connectivity from server_2 to a link-local destination, type:

```
$ server_ping6 server_2 fe80::260:16ff:fe0c:205%cge0_0000_ll
```

Output

```
server_2 : fe80::260:16ff:fe0c:205%cge0_0000_ll is alive, time= 0 ms
```

Action

To ping a multicast address through a Data Mover, use this command syntax:

```
$ server_ping6 <movername> <ip6_addr>
```

where:

<movername> = name of the Data Mover

<ipaddr> = IP address of a multicast destination to ping

Example:

To ping multicast address ff02::1%cge0_0000_ll, type:

```
$ server_ping6 server_2 ff02::1%cge0_0000_ll
```

Output

```
server_2 : ff02::1%cge0_0000_ll is alive, time= 0 ms
```

Configure interface options

To configure interface options after configuring the IP interfaces:

- Enable VLAN tagging on page 45
- Move an interface from one VLAN to another on page 46
- MTU configuration on page 48

Enable VLAN tagging

To enable VLAN tagging on an interface, you must set the VLAN ID for the interface. The value of the VLAN ID can be 0 or between 1 and 4094. A VLAN ID of 0 means the interface is not configured for VLAN tagging. A VLAN ID between 1 and 4094 enables the interface to process VLAN IDs.

On many switches, VLAN 1 is used for the native VLAN; so a VLAN ID of 1 should not be set on a Data Mover. Every physical switch port has a native VLAN ID assigned to it. By default, this is VLAN 1. When a packet with a VLAN tag is received by a port, the tag is used to direct the packet to the appropriate VLAN. If an incoming packet is untagged, the native VLAN ID is used as the tag. Appendix B provides more information.

Note: Ensure that the IP network switch settings match the device settings.

Action

To enable VLAN tagging, use this command syntax:

```
$ server_ifconfig <movername> <if_name> vlan=<vlanID>
```

where:

<movername> = name of the Data Mover on which the interface exists

<if_name> = name of the interface on which you are changing the VLAN ID

<vlanID> = VLAN ID between 0 (the default, no VLAN tagging) and 4094

Example:

To enable VLAN tagging on network interfaces cge2_1 and cge2_2 on server_2 with VLAN IDs of 102 and 103, type:

```
$ server_ifconfig server_2 cge2_1 vlan=102
$ server_ifconfig server_2 cge2_2 vlan=103
```

Output

```
server_2 : done
```

Verify the settings for each interface

Action
To verify the settings for each interface, use this command syntax:
`$ server_ifconfig <movername> <if_name>`
where:
`<movername>` = name of the Data Mover
`<if_name>` = name of the interface
Example:
To verify the settings for the cge2_1 interface on server_2, type:
`$ server_ifconfig server_2 cge2_1`

Output

```
server_2 :
cge2_1 protocol=IP device=cge2
        inet=172.24.102.10 netmask=255.255.255.0 broadcast=172.24.102.255
        UP, ethernet, mtu=1500, vlan=102, macaddr=8:0:1b:42:86:95
```

Move an interface from one VLAN to another

To move the interface from one VLAN to another, first remove the VLAN tagging by setting the VLAN ID to 0. Then assign the new VLAN ID to each interface.

Action
To remove the VLAN tagging, use this command syntax:
`$ server_ifconfig  <movername> <if_name> vlan=<vlanID>`
where:
`<movername>` = name of the Data Mover on which the interface exists
`<if_name>` = name of the interface on which you are changing the VLAN ID
`<vlanID>` = VLAN ID between 0 (the default no VLAN tagging) and 4094
Examples:
To change the VLAN settings for cge4 and cge3 interfaces on server_2 to 0, type:
`$ server_ifconfig server_2 cge4 vlan=0`
`$ server_ifconfig server_2 cge3 vlan=0`

Output

```
server_2: done
```

After setting the VLAN ID to 0, enable the VLAN tagging to assign new VLAN ID to each interface.

Action
To enable VLAN tagging, use this command syntax: `$ server_ifconfig <movername> <if_name> vlan= <vlanID>` where: *<movername>* = name of the Data Mover on which the interface exists *<if_name>* = name of the interface on which you are changing the VLAN ID *<vlanID>* = VLAN ID between 0 (the default, no VLAN tagging) and 4094 Examples: To change the VLAN settings for cge4 and cge3 interfaces on server_2, type: **`$ server_ifconfig server_2 cge4 vlan=100`** **`$ server_ifconfig server_2 cge3 vlan=200`**
Output
`server_2: done`

Note: If the Data Mover has multiple interfaces configured in the same subnet, then the VLAN tag needs to be removed on ALL of the interfaces (by setting VLAN ID to 0) before the new VLAN ID can be configured on any of them.

Verify a VLAN ID

Note: The link-local address uses the VLAN tag as part of its name.

Action
To verify that the settings for an interface are correct, use this command syntax: `$ server_ifconfig <movername> -all` where: *<movername>* = name of the Data Mover on which to verify the interfaces Example: To verify that the settings for server_2 are correct, type: **`$ server_ifconfig server_2 -all`**

Output

```
server_2 :
cge0_int1 protocol=IP6 device=cge0
         inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
         UP, Ethernet, mtu=1500, vlan=40, macaddr=0:60:16:c:2:5
cge0_0040_ll protocol=IP6 device=cge0
         inet=fe80::260:16ff:fe0c:205 prefix=64
         UP, Ethernet, mtu=1500, vlan=40, macaddr=0:60:16:c:2:5
loop6 protocol=IP6 device=loop
         inet=::1 prefix=128
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
```

Note: Changing the VLAN for global IPv6 interface will change the VLAN for both the global and associated link-local interfaces. A VLAN tag for an existing link-local interface cannot be modified. The existing link-local interface must be deleted and a new one specifying the new VLAN tag should be created.

MTU configuration

The MTU setting determines the largest packet size that can be transmitted without undergoing fragmentation.

The initial settings are defined per protocol and are configured by default depending on the type of network interface cards installed. The default MTU setting for an Ethernet interface card is 1500 bytes. The valid range is 1 to 9000 bytes.

Regardless of whether you have Ethernet or Gigabit Ethernet, the initial default MTU size is 1500 bytes. To take advantage of the capacity of Gigabit Ethernet, you can increase the MTU to 9000 bytes if your switch supports jumbo frames. Jumbo frames should be used only when the entire infrastructure, including client NICs, supports them.

For UDP, ensure that the client and server use the same MTU size. TCP negotiates the MTU size when the connection is initialized. The switch's MTU must be greater than or equal to the host's MTU.

Note: When one IP host has a large amount of data to send to another host, the data is transmitted as a series of IP datagrams. Typically, these datagrams should be the largest size that does not require fragmentation anywhere along the path from the source to the destination.

Configure the MTU

Action

To configure the MTU size for an interface on a Data Mover, use this command syntax:

```
$ server_ifconfig <movername> <if_name> mtu=<MTUbytes>
```

Action
where:
<movername> = name of the Data Mover on which the interface exists
<if_name> = name of the interface on which you are changing the MTU size
<MTUbytes> = new MTU size, in bytes
Example:
To configure the MTU size to 9000 bytes for the interfaces cge3_1 and cge3_2 on server_2, type:
`$ server_ifconfig server_2 cge3_1 mtu=9000`
`$ server_ifconfig server_2 cge3_2 mtu=9000`
Output
`server_2 : done`

Verify the MTU size configured

When verifying the configured MTU size, ensure that the IP network switch settings match the device settings.

Action
To verify the settings for an interface, use this command syntax:
`$` **`server_ifconfig`** *`<movername> <if_name>`*
where:
<movername> = name of the Data Mover
<if_name> = name of the interface
Example:
To verify the settings for the cge3_1 interface on server_2, type:
`$ server_ifconfig server_2 cge3_1`

Output

```
server_2 :
cge3_1 protocol=IP device=cge3
     inet=172.24.104.10 netmask=255.255.255.0 broadcast=172.24.104.255
     UP, ethernet, mtu=9000, vlan=0, macaddr=8:0:1b:42:86:9a
```

Verify Data Mover connectivity

After configuring the IP interfaces, verify Data Mover connectivity for each interface configured.

You can ping by using either the hostname or the IP address of the client for which you want to check connectivity with the Data Mover. Invoking ping with a hostname requires the local host file, Lightweight Directory Access Protocol (LDAP), Network Information Service (NIS), or Domain Name System (DNS) server to be operational on the Data Mover. To invoke ping with the IP address, type the client's IP address.

Configuring VNX Naming Services provides more information about name resolution, the local host file, OpenLDAP, NIS, and DNS.

Note: When invoking server_ping with a hostname, the /.etc/hosts file on the Data Mover is checked first, and then the LDAP-based directory, NIS, and the DNS servers are checked, if operational. If the name does not exist in any of these locations, an error message appears.

Action

To verify Data Mover connectivity, use this command syntax:

```
$ server_ping <movername> -interface <interface> <ipaddr>
```

or

```
$ server_ping <movername> -interface <interface> <hostname>
```

where:

<movername> = name of the Data Mover from which to ping the client system

<interface> = a specific port

<ipaddr> = IP address of the client to ping

<hostname> = hostname of the client to ping

Example:

Ping from the Data Mover to another host on the same network as the interface being verified. To verify Data Mover connectivity from cge0 on server_2 to a client at 172.24.108.20, type:

```
$ server_ping server_2 -interface cge0 172.24.108.20
```

Output

If the ping is successful:

```
server_2 : 172.24.108.20 is alive, time= 0 ms
```

If a response message is not received in 20 seconds:

```
no answer from client
```

Verify Data Mover connectivity for an IPv6 address

Action
To verify Data Mover connectivity for IPv6, use this command syntax:
`$ server_ping6 <movername> -interface <interface> <ipv6_addr>`
where:
`<movername>` = name of the Data Mover from which to ping the client system
`<interface>` = a specific port
`<ipv6_addr>` = IP address of the client to ping
Example:
Ping through an IPv6 interface. To verify Data Mover connectivity from cge0_int1 on server_2 to a client at 3ffe:0000:3c4d:0015:0435:0200:0300:00aa, type:
`$ server_ping6 server_2 -interface cge0_int1 3ffe:0000:3c4d:0015:0435:0200:0300:00aa`
Output
If the ping is successful:
`server_2 : 3ffe:0000:3c4d:0015:0435:0200:0300:00aa is alive, time= 0 ms`
If a response message is not received in 20 seconds:
`no answer from client`

Ping a link-local address

Action
To verify a link-local address, use this command syntax:
`$ server_ping6 <movername> <link-local-addr>`
where:
`<movername>` = name of the Data Mover
`<link-local-addr>` = link-local address to ping
Example:
To ping link-local address fe80::260:16ff:fe0c:205 by using interface cge0_0000_ll, type:
`$ server_ping6 server_2 fe80::260:16ff:fe0c:205 via interface cge0_0000_ll`

Output

If the ping is successful:
`server_2 : fe80::260:16ff:fe0c:205 via interface cge0_0000_ll is alive, time= 0 ms`
If a response message is not received in 20 seconds:
`no answer from client`

Ping a multicast address

To ping a multicast address use the prefix ff02::1 which specifies a link-local scope all-nodes multicast address.

Action
To verify a multicast address, use this command syntax: `$` **`server_ping6`** *`<movername> <multicast-addr>`* where: *`<movername>`* = name of the Data Mover from which to ping the client system *`<multicast-addr>`* = multicast address to ping Example: To ping multicast address ff02::1 by using interface cge0_0000_ll, type: **`$ server_ping6 server_2 ff02::1 via interface cge0_0000_ll`**
Output

If the ping is successful:
`server_2 : ff02::1 via interface cge0_0000_ll is alive, time= 0 ms`
If a response message is not received in 20 seconds:
`no answer from client`

Configure routing

Perform these tasks to configure routing after configuring your IPv4 interfaces:

- List routing table entries for IPv4 on page 53
- Default interface on page 53
- Add a default route on page 55
- Display current RIP status on page 59
- Disable RIP on a Data Mover on page 59

Perform these tasks to configure routing after configuring your IPv6 interfaces:

- List routing table entries for IPv6 on page 60
- Add an IPv6 prefix entry to the routing table on page 61
- Add a default gateway by using an IPv6 global unicast address on page 61
- Add a default gateway by using an IPv6 link-local address on page 62

List routing table entries for IPv4

Action
To list the routing table entries (including destination, gateway, subnet mask, interface, and type of route) for a Data Mover, use this command syntax: `$ server_route <movername> -list` where: *<movername>* = name of the Data Mover Example: To list the routing table entries for server_2, type: **`$ server_route server_2 -list`**
Output
The routing table lists all interface (directly connected) routes created by the system when you configure the IP interfaces. Interface routes are permanent routes that cannot be updated manually through the RIP or by the administrator. However, static routes can be updated manually by the administrator and dynamic routes can be updated through the RIP. The routing table contains one route per destination network:

```
server_2 :
net 128.221.253.0 128.221.253.2 255.255.255.0 el31
net 128.221.252.0 128.221.252.2 255.255.255.0 el30
net 172.24.106.0 172.24.106.10 255.255.255.128 cge4
net 172.24.104.0 172.24.104.10 255.255.255.0 cge3_1
net 172.24.103.0 172.24.103.10 255.255.255.0 cge2_2
net 172.24.102.0 172.24.102.10 255.255.255.0 cge2_1
net 172.24.101.0 172.24.101.10 255.255.255.0 cge1
net 172.24.108.0 172.24.108.10 255.255.255.0 cge0
host 127.0.0.1 127.0.0.1 255.255.255.255 loop
```

Default interface

When there are multiple interfaces on the same network, one interface is the default. If the default interface goes down, another interface on the same network becomes the default. To list the default interface:

- List the default interface on page 54

- Simulate a disruption on an interface on page 54
- List the new default interface on page 55

In this example two interfaces, cge3_1 and cge3_2, have been created on the 104 network. When cge3_1, the default, goes down, it is replaced by cge3_2.

List the default interface

Action

To list the routing table entries for a Data Mover, use this command syntax:

```
$ server_route <movername> -list
```

where:

<movername> = name of the Data Mover

Example:

To list the routing table entries for server_2, type:

```
$ server_route server_2 -list
```

Output

```
server_2 :
net 128.221.253.0 128.221.253.2 255.255.255.0 el31
net 128.221.252.0 128.221.252.2 255.255.255.0 el30
net 172.24.106.0 172.24.106.10 255.255.255.128 cge4
net 172.24.104.0 172.24.104.10 255.255.255.0 cge3_1
net 172.24.103.0 172.24.103.10 255.255.255.0 cge2_2
net 172.24.102.0 172.24.102.10 255.255.255.0 cge2_1
net 172.24.101.0 172.24.101.10 255.255.255.0 cge1
net 172.24.108.0 172.24.108.10 255.255.255.0 cge0
host 127.0.0.1 127.0.0.1 255.255.255.255 loop
```

Simulate a disruption on an interface

Action

To simulate a disruption on an interface, use this command syntax:

```
$ server_ifconfig <movername> <if_name> down
```

where:

<movername> = name of the Data Mover

<if_name> = name of the interface

Example:

To simulate a disruption on cge3_1, set the interface to down by typing:

```
$ server_ifconfig server_2 cge3_1 down
```

Output
`server_2 : done`

List the new default interface

Action

To list the routing table entries for a Data Mover, use this command syntax:

```
$ server_route <movername> -list
```

where:

<movername> = name of the Data Mover

Example:

To list the routing table entries for server_2, type:

```
$ server_route server_2 -list
```

Output

```
server_2 :
net 128.221.253.0 128.221.253.2 255.255.255.0 el31
net 128.221.252.0 128.221.252.2 255.255.255.0 el30
net 172.24.106.0 172.24.106.10 255.255.255.128 cge4
net 172.24.104.0 172.24.104.12 255.255.255.0 cge3_2
net 172.24.103.0 172.24.103.10 255.255.255.0 cge2_2
net 172.24.102.0 172.24.102.10 255.255.255.0 cge2_1
net 172.24.101.0 172.24.101.10 255.255.255.0 cge1
net 172.24.108.0 172.24.108.10 255.255.255.0 cge0
host 127.0.0.1 127.0.0.1 255.255.255.255 loop
```

Add a default route

If the Data Mover needs to access networks without a route defined, you can add a default route. For most communications, Packet Reflect is used. Manage Packet Reflect on page 94 provides more information. However, when the Data Mover initiates communications, the routing table is used. When you configure a default route, the Data Mover passes all packets not matching a specific route entry to the specified gateway.

To add a default route:

- Configure a default route on page 56
- Ping the gateway on page 57
- Verify Data Mover connectivity to another network on page 57

Configure a default route

Action
To configure a default route to the routing table for a Data Mover, where 172.24.101.254 is a router in the network, use this command syntax: `$ server_route <movername> -add default <gateway>` where: *<movername>* = name of the Data Mover *<gateway>* = IP address of the gateway machine Example: To configure a default route to the routing table for server_2, type: **`$ server_route server_2 -add default 172.24.101.254`**
Output
`server_2 : done`

Verify the default route configured

Action
To list the routing table entries for a Data Mover, use this command syntax: `$ server_route <movername> -list` where: *<movername>* = name of the Data Mover Example: To list the routing table entries for server_2, type: **`$ server_route server_2 -list`**
Output

```
server_2 :
default 172.24.101.254 0.0.0.0 cge1
net 128.221.253.0 128.221.253.2 255.255.255.0 el31
net 128.221.252.0 128.221.252.2 255.255.255.0 el30
net 172.24.106.0 172.24.106.10 255.255.255.128 cge4
net 172.24.104.0 172.24.104.10 255.255.255.0 cge3
net 172.24.103.0 172.24.103.10 255.255.255.0 cge2_2
net 172.24.102.0 172.24.102.10 255.255.255.0 cge2_1
net 172.24.101.0 172.24.101.10 255.255.255.0 cge1
net 172.24.108.0 172.24.108.10 255.255.255.0 cge0
host 127.0.0.1 127.0.0.1 255.255.255.255 loop
```

Note
The interface cge1 is used to access the IP address of the default gateway on the 101 network.

Ping the gateway

You can ping the 172.24.101.254 gateway, because it is connected to the local Data Mover network (172.24.101.10 on cge1) and it provides a path to other networks that are not explicitly defined in the routing table.

Action
To verify Data Mover connectivity to 172.24.101.254 on server_2, type: **`$ server_ping server_2 172.24.101.254`**
Output

If the ping is successful:
`server_2 : 172.24.101.254 is alive, time= 0 ms`
If a response message is not received in 20 seconds:
`no answer from client`

Verify Data Mover connectivity to another network

The routing table does not contain a route for the 105 network. Check if the default route you configured is working properly by pinging a client on the 105 network.

Action
To verify Data Mover connectivity to another network, use this command syntax: `$` **`server_ping`** *`<movername>`* **`-interface`** *`<interface>`* *`<ipv4_addr>`* where: *`<movername>`* = name of the Data Mover from which to ping the client system *`<interface>`* = a specific port *`<ipv4_addr>`* = IP address of the client to ping Example: Ping from the Data Mover to ensure that the routing table is used. To verify Data Mover connectivity to 172.24.105.30 on server_2, type: **`$ server_ping server_2 172.24.105.30`**

Output

If the ping is successful:
`server_2 : 172.24.105.30 is alive, time= 0 ms`
If a response message is not received in 20 seconds:
`no answer from client`

Add a static routing table entry

VNX does not support noncontiguous network masks; that is, masks without a continuous stream of 1 bit. A netmask of 0.0.0.0 or 255.255.255.255 is invalid for net routes. By default, a netmask of 255.255.255.255 is assigned to host routes.

Action
To add a static route to the routing table entry on a Data Mover, use this command syntax: `$ server_route <movername> -add [host|net] <dest> <gateway> [<netmask>]` where: *`<movername>`* = name of the Data Mover *`<dest>`* = IP address of the destination *`<gateway>`* = IP address of the gateway machine *`<netmask>`* = network mask for the interface **`host`** = routing entry to a particular host **`net`** = routing entry to a particular network Examples: To add a static host route on server_2, type: **`$ server_route server_2 -add host 172.24.105.20 172.24.101.254 255.255.255.255`** To add a static network route on server_2, type: **`$ server_route server_2 -add net 172.24.107.0 172.24.101.254 255.255.255.0`**
Output
`server_2 : done`

Display current RIP status

RIP is a dynamic routing protocol supported by VNX. RIP determines a network route based on the smallest hop count between the source and the destination. By default, the Data Mover listens for RIP route advertisements on all interfaces.

Action
To display RIP status and the interfaces with RIP disabled for a Data Mover, use this command syntax: `$ server_rip <movername> status` where: *<movername>* = name of the Data Mover Example: To display the RIP status for server_2, type: **`$ server_rip server_2 status`**
Output

`server_2 :`
`routed started, RIP processing is on`

Disable RIP on a Data Mover

You can disable RIP for the Data Mover or disable RIP at the interface level. Disable RIP on page 81 provides information about disabling RIP for an interface. To prevent RIP routes from being entered into the routing table, disable RIP for the Data Mover.

Action
To disable RIP for a Data Mover, use this command syntax: `$ server_setup <movername> -Protocol <protocol> -option stop` where: *<movername>* = name of the Data Mover with the interface *<protocol>* = rip for this example Example: To disable RIP for server_2, type: **`$ server_setup server_2 -Protocol rip -option stop`**

Output
`server_2 : done`

Verify the RIP status

Action
To verify the RIP status for a Data Mover, use this command syntax:
`$ server_rip <movername> status`
where:
<movername> = name of the Data Mover
Example:
To verify the RIP status for server_2, type:
`$ server_rip server_2 status`

Output

```
server_2 :
routed started, RIP processing is off
```

List routing table entries for IPv6

Action
To list the routing table entries (including destination, gateway, interface, and age of the route entry) for a Data Mover, use this command syntax:
`$ server_ip <movername> -route -list`
where:
<movername> = name of the Data Mover
Example:
To list the routing table entries for server_2, type:
`$ server_ip server_2 -route -list`

Output

```
server_2 :
Destination                                 Gateway
      Interface                             Expires (secs)
2620:0:170:84f::/64
      2620:0:170:84f::ac18:fc36             0
2620:0:170:879::/64
      2620:0:170:879::2dfc:12a              0
selected default                            fe80::215:2cff:febd:1c00
      fe80::260:16ff:fe26:113b              1789
```

Add an IPv6 prefix entry to the routing table

Action

To permanently add a prefix entry to the routing table, use this command syntax:

```
$ server_ip <movername> -route -create -destination <destination> -interface
<if_name>
```

where:

<movername> = name of the Data Mover

<destination> = IPv6 prefix (including the prefix length) for the route entry, a global unicast address (required)

<if_name> = name of the interface. The name can consist of 1 to 63 characters including a-z, A-Z, 0-9, or underscore (_). No leading or trailing spaces are allowed.

The -interface is optional when adding a default gateway.

Example:

To permanently add a prefix entry to the routing table on server_2, type:

```
$ server_ip server_2 -route -create -destination 2002:8c8:0:2314::/64 -interface
cge4v6
```

Output

```
OK
```

Add a default gateway by using an IPv6 global unicast address

A default entry specifies the gateway to use as default when none of the prefixes match the destination.

Action

To permanently add a default gateway by using an IPv6 global unicast address to the routing table, use this command syntax:

Action
`$ server_ip <movername> -route -create -default -gateway <v6gw>`
where:
`<movername>` = name of the Data Mover
`<v6gw>` = IPv6 gateway for the default route, a global unicast address (required). The global unicast address should have a prefix route entry to reach the gateway.
Example:
To permanently add a default gateway by using an IPv6 global unicast address to the routing table on server_2, type:
`$ server_ip server_2 -route -create -default -gateway 2002:8c8:0:2314::1`
Output
`OK`

Add a default gateway by using an IPv6 link-local address

Action
To permanently add a default gateway by using an IPv6 link-local address to the routing table, use this command syntax:
`$ server_ip <movername> -route -create -default -gateway <v6gw> -interface <ifname>`
where:
`<movername>` = name of the Data Mover
`<v6gw>` = IPv6 gateway for the default route, a link-local address (required).
`<if_name>` = name of the interface (Optional). The name can consist of 1 to 63 characters including a-z, A-Z, 0-9, or underscore (_). No leading or trailing spaces are allowed.
Example:
To permanently add a default gateway by using an IPv6 link-local address to the routing table on server_2, type:
`$ server_ip server_2 -route -create -default -gateway fe80::1 -interface cge1v6`
Output
`OK`

4

Managing

The tasks to manage networking are:

- Monitor network devices on page 64
- Manage IP interfaces on page 64
- Manage routing on page 78
- Manage address resolution on page 85
- Manage the neighbor cache on page 88
- Manage Packet Reflect on page 94
- View network statistics on page 94
- Modify Control Station networking on page 95

Monitor network devices

You can monitor the state of your network links by means of VNX notification feature.

You can configure notifications in response to a variety of device driver events to notify you through e-mail, a Simple Network Management Protocol (SNMP) trap, or messages written to a log file. *Configuring Events and Notifications on VNX for File* provides more information about VNX events.

Display device driver events

Action
To view a list of events and event ID numbers associated with a facility, use this command syntax:
`$ nas_event -list -c DART -f <facility>`
where:
`<facility>` = name of the facility
Example:
To list the events associated with the DRIVERS facility, type:
`$ nas_event -list -c DART -f DRIVERS`
Output
The output from the nas_event command displays a list of driver events.

Manage IP interfaces

When changing or deleting IP interfaces, applications that use the interface, such as CIFS or NFS for exporting file systems, need to be reconfigured to account for the new conditions (for example, changed IP addresses).

To modify or manage existing IP interfaces:

- Modify an IPv4 interface on page 65
- Modify VLAN tagging for an IPv4 device on page 67
- Modify the MTU size on page 70
- Disable an IP interface on page 72
- Enable an IP interface on page 73
- View all IP interfaces on a Data Mover on page 74

Modify an IPv4 interface

To modify the IP address, netmask, or broadcast address of an IPv4 interface, you must delete the interface and create a new IP interface for the device with the appropriate settings.

1. Delete an IP interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> -delete <if_name>
   ```

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

   Example:

   Delete interface cge0 on server_2, by typing:

   ```
   $ server_ifconfig server_2 -delete cge0
   ```

   Output:

   ```
   server_2 : done
   ```

2. Create a new interface for a network device by using this command syntax:

   ```
   $ server_ifconfig <movername> -create -Device <device_name> -name <if_name>
   -protocol IP <ipv4_addr> <ipmask> <ipbroadcast>
   ```

   where:

   <movername> = name of the Data Mover where the device is located

   <device_name> = name of the network device

   <if_name> = name for the interface created by this command

   <ipv4_addr> = IPv4 address for the interface

   <ipmask> = network mask for the interface

   <ipbroadcast> = broadcast address for the interface

   Example:

   Create a new interface for network device cge0 on server_2 by typing:

   ```
   $ server_ifconfig server_2 -create -Device cge0 -name cge0

   -protocol IP 172.24.108.12 255.255.255.0 172.24.108.255
   ```

   Output:

   ```
   server_2 : done
   ```

Modify an IPv6 interface

To modify the IPv6 address of an IP interface, you must delete the interface and create a new IP interface for the device with the appropriate settings.

This procedure deletes the interface 3ffe:0000:3c4d:0015:0435:0200:0300:00aa and then creates a new interface 3ffe:0000:3c4d:0015:0435:0200:0400:DDE0 with a new IP address.

1. To delete an IP interface on a Data Mover, use this command syntax:

   ```
   $ server_ifconfig <movername> -delete <if_name>
   ```

   where:

   <movername> = name of the Data Mover on which the interface to delete exists

   <if_name> = name of the interface to delete

   Example:

   To delete interface cge0_int1 on server_2, type:

   ```
   $ server_ifconfig server_2 -delete cge0_int1
   ```

   Output:

   ```
   server_2 : done
   ```

2. To create a new interface for a network device, use this command syntax:

   ```
   $ server_ifconfig <movername> -create -Device <device_name> -name <if_name>
   -protocol IP6 <ipv6_addr>
   ```

   where:

   <movername> = name of the Data Mover where the device is located

   <device_name> = network device name

   <if_name> = name for the interface created by this command

   <ipv6_addr> = IPv6 address for the interface

   Example:

   To create a new interface for network device cge0 on server_2, type:

   ```
   $ server_ifconfig server_2 -create -Device cge0 -name cge0_int1 -protocol IP6
   3ffe:0000:3c4d:0015:0435:0200:0400:DDE0
   ```

   Output:

   ```
   server_2 : done
   ```

3. To verify that the new interface on the Data Mover is correct, use this command syntax:

   ```
   $ server_ifconfig <movername> <if_name>
   ```

   where:

<movername> = name of the Data Mover on which to verify the interfaces

<if_name> = name for the interface created by this command

Example:

To view the cge0_int1 interface for server_2, type:

```
$ server_ifconfig server_2 cge0_int1
```

Output:

```
server_2 :
cge0_int1 protocol=IP6 device=cge0
         inet=3ffe:0:3c4d:15:435:200:400:dde0 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:5:5
```

Modify VLAN tagging for an IPv4 device

To modify a VLAN tag for an interface, change the value of the VLAN ID. To remove a VLAN tag for an interface, change the value of the VLAN ID to 0 (zero).

Ensure that the IP network switch settings match the device settings. VNX and the network switch must support IEEE 802.1Q VLAN tagging.

1. Verify the current settings for an interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> <if_name>
   ```

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

   Example:

   Verify the current settings for the cge1 interface on server_2 by typing:

   ```
   $ server_ifconfig server_2 cge1
   ```

   Output:

   ```
   server_2 :
   cge1 protocol=IP device=cge1
        inet=172.24.101.10 netmask=255.255.255.0
        broadcast=172.24.101.255
        UP, ethernet, mtu=1500, vlan=0, macaddr=0:60:16:4:35:30
   ```

2. Enable VLAN tagging for an interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> <if_name> vlan=<vlanID>
   ```

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

<vlanID> = VLAN ID between 0 (the default, no VLAN tagging) and 4094

Example:

Enable VLAN tagging on network interface cge1 on server_2 with a VLAN ID of 101 by typing:

```
$ server_ifconfig server_2 cge1 vlan=101
```

Output:

```
server_2 : done
```

3. Verify the new settings for an interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> <if_name>
   ```

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

   Example:

   Verify the new settings for the interface cge1 on server_2 by typing:

   ```
   $ server_ifconfig server_2 cge1
   ```

   Output:

   ```
   server_2 :
   cge1 protocol=IP device=cge1
        inet=172.24.101.10 netmask=255.255.255.0
        broadcast=172.24.101.255
        UP, ethernet, mtu=1500, vlan=101, macaddr=0:60:16:4:35:30
   ```

4. Remove VLAN tagging from an interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> <if_name> vlan=0
   ```

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

   Example:

   Remove VLAN tagging from network interface cge1 on server_2 by typing:

   ```
   $ server_ifconfig server_2 cge1 vlan=0
   ```

   Output:

   ```
   server_2 : done
   ```

5. Verify the settings for an interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> <if_name>
   ```

   where:

   <movername> = name of the Data Mover

<if_name> = name of the interface

Example:

Verify the settings for the cge1 interface on server_2 by typing:

```
$ server_ifconfig server_2 cge1
```

Output:

```
server_2 :
cge1 protocol=IP device=cge1
     inet=172.24.101.10 netmask=255.255.255.0
     broadcast=172.24.101.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=0:60:16:4:35:30
```

Modify VLAN tagging for an IPv6 device

To modify a VLAN tag for an interface, change the value of the VLAN ID. To remove a VLAN tag for an interface, change the value of the VLAN ID to 0 (zero).

Ensure that the IP network switch settings match the device settings. Both VNX and the network switch must support IEEE 802.1Q VLAN tagging.

1. To verify that the current settings for the cge0_int1 interface for server_2 are correct, type:

   ```
   $ server_ifconfig server_2 cge0_int1
   ```

   Output:

   ```
   server_2 :
   cge0_int1 protocol=IP6 device=cge0
            inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
            UP, Ethernet, mtu=1500, vlan=40, macaddr=0:60:16:c:2:5
   ```

2. To enable VLAN tagging, use this command syntax:

   ```
   $ server_ifconfig <movername> <if_name> vlan=<vlanID>
   ```

   where:

   <movername> = name of the Data Mover on which the interface exists

   <if_name> = name of the interface for which you are setting the VLAN ID

   <vlanID> = VLAN ID between 0 (the default, no VLAN tagging) and 4094

   Example:

   To enable VLAN tagging on network interface cge0_int1 on server_2 with a VLAN ID of 101, type:

   ```
   $ server_ifconfig server_2 cge0_int1 vlan=101
   ```

   Output:

   ```
   server_2 : done
   ```

3. To verify that the new settings for the cge0_int1 interface for server_2 are correct, type:

   $ server_ifconfig server_2 cge0_int1

   Output:

   ```
   server_2 :
   cge0_int1 protocol=IP6 device=cge0
            inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
            UP, Ethernet, mtu=1500, vlan=101, macaddr=0:60:16:c:2:5
   ```

4. To remove VLAN tagging from an interface, use this command syntax:

   $ **server_ifconfig** *<movername> <if_name> <vlanID>*

   where:

   <movername> = name of the Data Mover on which the interface exists

   <if_name> = name of the interface on which you are removing the VLAN ID

   <vlanID> = VLAN ID between 0 (the default, no VLAN tagging) and 4094

   Example:

   To remove VLAN tagging from network interface cge0_int1 on server_2, type:

   $ server_ifconfig server_2 cge0_int1 vlan=0

   Output:

   ```
   server_2 : done
   ```

5. Verify the settings for the cge0_int1 interface for server_2, type:

   $ server_ifconfig server_2 cge0_int1

   Output:

   ```
   server_2 :
   cge0_int1 protocol=IP6 device=cge0
            inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
            DOWN, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
   ```

Note: The vlan tag cannot be modified for an existing link-local interface. The existing link-local needs to be deleted and a new one specifying the new vlan tag needs to be created.

Modify the MTU size

1. Verify the settings for an interface on a Data Mover by using this command syntax:

   $ **server_ifconfig** *<movername> <if_name>*

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

Example:

Verify the settings for the cge3_1 interface on server_2 by typing:

```
$ server_ifconfig server_2 cge3_1
```

Output:

```
server_2 :
cge3_1 protocol=IP device=cge3
 inet=172.24.104.10 netmask=255.255.255.0 broadcast=172.24.104.255
 UP, ethernet, mtu=9000, vlan=0, macaddr=8:0:1b:42:86:9a
```

2. Modify the MTU size of an interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> <if_name> mtu=<MTUbytes>
   ```

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

   <MTUbytes> = new MTU size, in bytes

   Note: Ensure that you modify the MTU settings to the same value for all interfaces on the device.

   Example:

   Modify the MTU size to 1500 bytes for the interfaces cge3_1 and cge3_2 on server_2 by typing:

   ```
   $ server_ifconfig server_2 cge3_1 mtu=1500
   ```

   ```
   $ server_ifconfig server_2 cge3_2 mtu=1500
   ```

   Output:

   ```
   server_2 : done
   ```

3. Verify the settings for an interface on a Data Mover by using this command syntax:

   ```
   $ server_ifconfig <movername> <if_name>
   ```

   where:

   <movername> = name of the Data Mover

   <if_name> = name of the interface

   Examples:

   Verify the settings for the cge3_1 interface on server_2 by typing:

   ```
   $ server_ifconfig server_2 cge3_1
   ```

   Output:

   ```
   server_2 :
   cge3_1 protocol=IP device=cge3
    inet=172.24.104.10 netmask=255.255.255.0 broadcast=172.24.104.255
    UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:9a
   ```

Verify the settings for the cge3_2 interface on server_2 by typing:

```
$ server_ifconfig server_2 cge3_2
```

Output:

```
server_2 :
cge3_2 protocol=IP device=cge3
 inet=172.24.104.10 netmask=255.255.255.0 broadcast=172.24.104.255
 UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:9a
```

Disable an IP interface

Action

To disable an interface on a Data Mover, use this command syntax:

```
$ server_ifconfig <movername> <if_name> down
```

where:

<movername> = name of the Data Mover

<if_name> = name of the interface

Example:

To disable the interface cge0 on server_2, type:

```
$ server_ifconfig server_2 cge0 down
```

Output

```
server_2 : done
```

Note

If access is not permitted to the internal interfaces, when using the server_ifconfig *<movername> <if_name>* down command, you receive the error message:

```
server_<x>: interface: invalid interface specified, you might be trying
to disable one of the internal interfaces.
```

Verify the state of the disabled IP interface

Action

To verify the state of an interface on a Data Mover, use this command syntax:

```
$ server_ifconfig <movername> <if_name>
```

where:

<movername> = name of the Data Mover

<if_name> = name of the interface

Action
Example: To verify the state of the cge0 interface on server_2, type: **`$ server_ifconfig server_2 cge0`**
Output

`cge0 protocol=IP device=cge0`
`     inet=172.24.108.12 netmask=255.255.255.0 broadcast=172.24.108.255`
`     DOWN, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:96`

Enable an IP interface

When you create a new IP interface on a network device, it is enabled automatically.

Action
To enable an interface on a Data Mover, use this command syntax: `$` **`server_ifconfig`** *`<movername> <if_name>`* **`up`** where: *`<movername>`* = name of the Data Mover *`<if_name>`* = name of the interface Example: To enable the interface cge0 on server_2, type: **`$ server_ifconfig server_2 cge0 up`**
Output
`server_2 : done`

Verify the state of the enabled IP interface

Action
To verify the state of the interface on a Data Mover, use this command syntax: `$` **`server_ifconfig`** *`<movername> <if_name>`* where: *`<movername>`* = name of the Data Mover *`<if_name>`* = name of the interface Example:

Action
To verify the state of the cge0 interface on server_2, type: **`$ server_ifconfig server_2 cge0`**
Output

```
cge0 protocol=IP device=cge0
     inet=172.24.108.12 netmask=255.255.255.0 broadcast=172.24.108.255
     UP, ethernet, mtu=1500, vlan=0, macaddr=8:0:1b:42:86:96
```

View all IP interfaces on a Data Mover

Action
To view the interfaces on a Data Mover, use this command syntax: `$` **`server_ifconfig`** *`<movername>`* **`-all`** where: *`<movername>`* = name of the Data Mover Example: To view the interfaces on server_2, type: **`$ server_ifconfig server_2 -all`**
Output

```
server_2 :
ar6667 protocol=IP device=cge1
         inet=11.11.11.11 netmask=255.255.255.0 broadcast=11.11.11.255
        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:5:d:49
cge1 protocol=IP device=cge1
         inet=45.244.1.182 netmask=255.255.255.0 broadcast=45.244.0.255
        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:5:d:49
cge1_v6 protocol=IP6 device=cge1
         inet=2002:8c8:0:2310:0:2:ac18:f412 prefix=64
        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:5:d:49
cge1_0000_ll protocol=IP6 device=cge1
         inet=fe80::260:16ff:fe05:d49 prefix=64
        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:5:d:49
```

View all IPv6 interfaces on a Data Mover

Action
To view all interfaces with IPv6 addresses on a Data Mover, use this command syntax:

Action
$ **server_ifconfig** *<movername>* **-all -ip6** where: *<movername>* = name of the Data Mover on which you want to view the interfaces Note: You can use the -ip6 flag to display the IPv6 configuration. Example: To view all interfaces with IPv6 addresses for server_2, type: **$ server_ifconfig server_2 -all -ip6**

Output

```
server_2 :
cge0_int1 protocol=IP6 device=cge0
         inet=3ffe:0:3c4d:15:435:200:400:dde0 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:5:5
cge0_0000_ll protocol=IP6 device=cge0
         inet=fe80::260:16ff:fe0c:205 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:5:5
```

View all IP interfaces on all Data Movers

Action
To view all interfaces with IPv4 and IPv6 addresses on all Data Movers, use this command syntax: $ **server_ifconfig ALL -all** Example: To view all interfaces for all Data Movers, type: **$ server_ifconfig ALL -all**

Output

```
server_2 :
el30 protocol=IP device=mge0
        inet=128.221.252.2 netmask=255.255.255.0 broadcast=128.221.252.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:d:30:b1 netname=lo-
calhost
el31 protocol=IP device=mge1
        inet=128.221.253.2 netmask=255.255.255.0 broadcast=128.221.253.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:d:30:b2 netname=lo-
calhost
loop6 protocol=IP6 device=loop
         inet=::1 prefix=128
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
loop protocol=IP device=loop
         inet=127.0.0.1 netmask=255.0.0.0 broadcast=127.255.255.255
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
cge0-new protocol=IP device=cge0
        inet=172.24.104.10 netmask=255.255.255.255 broadcast=172.24.24.255

         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
cge0_int1 protocol=IP6 device=cge0
         inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
cge0_0000_ll protocol=IP6 device=cge0
         inet=fe80::260:16ff:fe0c:205 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5

server_3
el30 protocol=IP device=mge0
        inet=128.221.252.3 netmask=255.255.255.0 broadcast=128.221.252.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:b:b7:18 netname=lo-
calhost
el31 protocol=IP device=mge1
        inet=128.221.253.3 netmask=255.255.255.0 broadcast=128.221.253.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:b:b7:17 netname=lo-
calhost
loop6 protocol=IP6 device=loop
         inet=::1 prefix=128
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
loop protocol=IP device=loop
         inet=127.0.0.1 netmask=255.0.0.0 broadcast=127.255.255.255
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
```

View all IPv6 interfaces on all Data Movers

Action
To view all interfaces with IPv6 addresses on all Data Movers, use this command syntax: $ **server_ifconfig ALL -all -ip6** Note: You can use the -ip6 flag to display the IPv6 configuration. Example: To view all IPv6 interfaces for all Data Movers, type: **$ server_ifconfig ALL -all -ip6**

Output

```
server_2 :
loop6 protocol=IP6 device=loop
         inet=::1 prefix=128
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
cge0_int1 protocol=IP6 device=cge0
         inet=3ffe:0:3c4d:15:435:200:300:ed20 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5
cge0_0000_ll protocol=IP6 device=cge0
         inet=fe80::260:16ff:fe0c:205 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:c:2:5

server_3
loop6 protocol=IP6 device=loop
         inet=::1 prefix=128
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
```

View all IPv4 interfaces on a Data Mover

Action
To view all IPv4 interfaces on a Data Mover, use this command syntax: $ **server_ifconfig** *<movername>* **-all -ip4** where: *<mover_name>* = name of the Data Mover Example: To view all IPv4 interfaces for server_3, type: **$ server_ifconfig server_3 -all -ip4**

Output

```
server_3 :
el30 protocol=IP device=mge0
        inet=128.221.252.3 netmask=255.255.255.0 broadcast=128.221.252.255

       UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:5:65:4a netname=lo-
calhost
el31 protocol=IP device=mge1
        inet=128.221.253.3 netmask=255.255.255.0 broadcast=128.221.253.255

       UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:5:65:4b netname=lo-
calhost
loop protocol=IP device=loop
         inet=127.0.0.1 netmask=255.0.0.0 broadcast=127.255.255.255
       UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=local-
host
```

Manage routing

To manage routing table information:

- Add a static routing table entry on page 58
- Delete an IPv4 route entry from the routing table on page 78
- Flush the routing table on page 80
- Remove all routing table entries on page 81
- Disable RIP on page 81
- Enable RIP on page 84

Delete an IPv4 route entry from the routing table

VNX does not support noncontiguous network masks; that is, masks without a continuous stream of 1 bit. A netmask of 0.0.0.0 or 255.255.255.255 is invalid for net routes. By default, a netmask of 255.255.255.255 is assigned to host routes.

Action

To permanently delete a routing table entry on a Data Mover, use this command syntax:

```
$ server_route <movername> -delete [host|net] <dest> <gateway> [<netmask>]
```

where:

<movername> = name of the Data Mover

<dest> = IP address of the destination

<gateway> = IP address of the gateway machine

<netmask> = network mask for the interface

Action
`host` = routing entry to a particular host
`net` = routing entry to a particular network
Example:
To delete a routing table entry from server_2, type:
`$ server_route server_2 -delete host 172.24.105.20 172.24.101.254 255.255.255.255`
Output
`server_2 : done`

Delete an IPv6 route entry from the routing table

Delete one or more IPv6 route entry permanently from the route table. The route entry can be a specified destination (prefix) or a default. The default route requires a gateway which can be either a global unicast or a link-local IPv6 address.

Delete a prefix entry from the routing table

Action
To delete a prefix entry from the routing table, use this command syntax:
`$` **`server_ip`** *`<movername>`* **`-route -delete`** *`<v6prefix>`*
where:
`<movername>` = name of the Data Mover
`<v6prefix>` = IPv6 prefix for the route entry, a global unicast address
Example:
To delete a prefix entry from the routing table on server_2, type:
`$ server_ip server_2 -route -delete 2002:8c8:0:2314::/64`
Output
`OK`

Delete a default gateway with an IPv6 global unicast address from the routing table

Action
To delete a default gateway with an IPv6 global unicast address from the routing table, use this command syntax:
`$` **`server_ip`** *`<movername>`* **`-route -delete -default -gateway`** *`<v6gw>`*

Action
where:
<movername> = name of the Data Mover
<v6gw> = IPv6 gateway for the default route, a global unicast or link-local address (required)
Example:
To delete a default gateway with an IPv6 global unicast address from the routing table on server_2, type:
`$ server_ip server_2 -route -delete -default -gateway 2002:8c8:0:2314::1`
Output
`OK`

Delete a default gateway with an IPv6 link-local address from the routing table

Action
To delete a default gateway with an IPv6 link-local address from the routing table, use this command syntax:
`$ server_ip <movername> -route -delete -default -gateway <v6gw> -interface <ifname>`
where:
<movername> = name of the Data Mover.
<v6gw> = IPv6 gateway for the default route, a global unicast or link-local address.
<if_name> = name of the interface (Optional). The name can consist of 1 to 63 characters including a-z, A-Z, 0-9, or underscore (_). No leading or trailing spaces are allowed.
Example:
To delete a default gateway with an IPv6 link-local address from the routing table on server_2, type:
`$ server_ip server_2 -route -delete -default -gateway fe80::1 -interface cge1v6`
Output
`OK`

Flush the routing table

When you flush the manually configured routing entries from the routing table, the system-generated entries are not removed.

Action
To temporarily flush manually configured routing entries from the routing table on a Data Mover, use this command syntax:
`$ server_route <movername> -flush`

Action
where:
<movername> = Data Mover on which to flush entries from the routing table
Example:
To temporarily flush the routing table from server_2, type:
`$ server_route server_2 -flush`

Output
`server_2 : done`

After flushing the routing table

If you are using static routing, the routing table is cleared of all entries only until the next time the Data Mover restarts. If RIP is enabled, the routed daemon replaces the entries as new RIP messages are processed.

Remove all routing table entries

When you remove all static routing table entries from the routing table, the deletion persists across restarts, permanently flushing all the static route entries from the specified Data Mover's routing table.

Action
To permanently remove all static routing table entries from the routing table on a Data Mover, use this command syntax:
`$` **`server_route`** *`<movername>`* **`-DeleteAll`**
where:
<movername> = name of the Data Mover
Example:
To remove all static routing table entries on server_2, type:
`$ server_route server_2 -DeleteAll`

Output
`server_2 : done`

Disable RIP

RIP is enabled on all interfaces by default. You can disable RIP for specific interfaces.

Interfaces that allow RIP add dynamic RIP routes to the Data Mover's routing table along with the interface routes and the default route you configured.

Note: Because RIP is enabled on all interfaces by default, the server_rip server_2 status command displays only those interfaces with RIP disabled.

The tasks to disable RIP on a specific interface or the interfaces of all Data Movers are:

- Disable RIP on an interface on page 82
- Disable RIP on the interfaces on page 83

Disable RIP on an interface

Action

To disable RIP on a Data Mover interface, use this command syntax:
`$ server_rip <movername> noripin {<interface_name> [,...]}`
where:
`<movername>` = name of the Data Mover
`<interface_name>` = name of the interface
Example:
To disable RIP on server_2, type:
`$ server_rip server_2 noripin cge0`

Output
`server_2 : done`

Verify that RIP is disabled on an interface

Action

To verify RIP status for a Data Mover interface, use this command syntax:
`$ server_rip <movername> status`
where:
`<movername>` = name of the Data Mover
Example:
To verify the RIP status on server_2, type:
`$ server_rip server_2 status`

Output

The output shows that RIP is disabled for interface cge0 on server_2:

```
server_2 :
routed started, RIP processing is on
    cge0 (10.172.128.239), ifp = 0xf6bd2274, disabled
```

Disable RIP on the interfaces

Action

To disable RIP for the interfaces on all Data Movers, use this command syntax:

`$ server_rip ALL noripin <interface_name> [,...]`

where:

`<interface_name>` = name of the interface

Example:

To disable RIP for cge0 and cge1 on all Data Movers, type:

`$ server_rip ALL noripin cge0, cge1`

Output

```
server_2 : done
server_3 : done
```

Verify that RIP is disabled on all interfaces

Action

To verify the RIP status for the interfaces on all Data Movers, use this command syntax:

`$ server_rip ALL status`

Example:

To verify the RIP status on all Data Movers, type:

`$ server_rip ALL status`

Output

The output shows that RIP is disabled for interfaces cge0 and cge1 on all Data Movers:

```
server_2 :
routed started, RIP processing is on
    cge0 (10.172.128.239), ifp = 0xf6bd2274, disabled
    cge1 (10.172.128.240), ifp = 0xf6bd2274, disabled
server_3 :
routed started, RIP processing is on
    cge0 (10.172.128.239), ifp = 0xf6bd2274, disabled
```

Enable RIP

RIP is enabled by default on all interfaces. Use this procedure to enable RIP after it has been disabled.

The tasks to enable RIP on a specific interface or the interfaces of all Data Movers are:

- Enable RIP on an interface on page 84
- Enable RIP on the interfaces on page 85

Enable RIP on an interface

Action

To enable RIP on a Data Mover interface, use this command syntax:
`$ server_rip <movername> ripin {<interface_name> [,...]}`
where:
`<movername>` = name of the Data Mover
`<interface_name>` = name of the interface
Example:
To enable RIP for cge0 on server_2, type:
`$ server_rip server_2 ripin cge0`

Output
`server_2 : done`

Verify that RIP is enabled on an interface

Action

To verify RIP status for a Data Mover interface, use this command syntax:
`$ server_rip <movername> status`
where:
`<movername>` = name of the Data Mover
Example:
To verify the RIP status on server_2, type:
`$ server_rip server_2 status`

Output

`server_2 :`
`routed started, RIP processing is on`

Enable RIP on the interfaces

Action
To enable RIP on interfaces for all Data Movers, use this command syntax: `$ `**`server_rip ALL ripin`**` {<interface_name> [,...]}` where: *`<interface_name>`* = name of the interface Example: To enable RIP for cge0 and cge1 on all Data Movers, type: **`$ server_rip ALL ripin cge0, cge1`**
Output

`server_2 : done`
`server_3 : done`

Verify that RIP is enabled on the interfaces

Action
To verify the RIP status for the interfaces on all Data Movers, use this command syntax: `$ `**`server_rip ALL status`** Example: To verify the RIP status on all Data Movers, type: **`$ server_rip ALL status`**
Output

`server_2 :`
`routed started, RIP processing is on`
`server_3 :`
`routed started, RIP processing is on`

Manage address resolution

The physical address is the MAC address in the form of nn:nn:nn:nn:nn:nn, where nn is one or two hexadecimal digits.

A dynamic ARP entry remains in the ARP table for approximately 10 minutes without activity. If there is activity, a dynamic ARP entry requests for a refresh after 5 minutes.

You can manage the ARP table, which is the IP-to-MAC address translation table. Managing ARP consists of some or all of these tasks:

- View ARP table entries on page 86
- Add an ARP table entry on page 86
- Delete an ARP table entry on page 87

View ARP table entries

You can view a maximum of 64 ARP entries by using the server_arp command.

Action

To view the ARP table on a Data Mover, use this command syntax:

```
$ server_arp <movername> -all
```

where:

<movername> = name of the Data Mover

Example:

To view the ARP table on server_2, type:

```
$ server_arp server_2 -all
```

Output

```
server_2 :
172.24.102.254 at 0:d0:3:f9:28:fc
172.24.103.120 at 8:0:20:a8:4f:2e
172.24.104.240 at 8:0:20:e9:1f:3f
172.24.101.222 at 0:c:29:4c:f:eb
128.221.252.100 at 0:2:b3:ea:e:11
128.221.253.100 at 0:2:b3:ea:e:11
```

Add an ARP table entry

Action

To add a permanent ARP table entry to the default gateway IP address on a Data Mover, use this command syntax:

```
$ server_arp <movername> -set <ip_addr> <physaddr>
```

where:

<movername> = name of the Data Mover

Action
<ip_addr> = IP address
<physaddr> = physical MAC address to associate with this IP address
Example:
To add an ARP table entry to the default gateway IP address on server_2, type:
`$ server_arp server_2 -set 172.24.101.254 0:d0:3:f9:37:fc`
Output
`server_2 : added: 172.24.101.254 at 0:d0:3:f9:37:fc`

Verify the ARP table entry added

Action
To verify the ARP table entry you created on the Data Mover, use this command syntax:
`$` **`server_arp`** *`<movername> <ip_addr>`*
where:
<movername> = name of the Data Mover
<ip_addr> = IP address of the gateway
Example:
To view the ARP table on server_2, type:
`$ server_arp server_2 172.24.101.254`
Output
`server_2 : 172.24.101.254 at 0:d0:3:f9:37:fc`

Delete an ARP table entry

Action
To delete an ARP table entry on a Data Mover, use this command syntax:
`$` **`server_arp`** *`<movername>`* **`-delete`** *`<ip_addr>`*
where:
<movername> = name of the Data Mover
<ip_addr> = IP address of the entry to be deleted
Example:
To delete an ARP entry on server_2, type:

Action
`$ server_arp server_2 -delete 172.24.101.254`
Output
`server_2 : deleted: 172.24.101.254 at 0:d0:3:f9:37:fc`

Manage the neighbor cache

The neighbor cache is similar to the IPv4 Address Resolution Protocl (ARP). It resolves IPv6 address and MAC addresses.

Managing the neighbor cache consists of some or all of these tasks:

- View the neighbor cache on page 88
- Add an entry to the neighbor cache on page 90
- Delete an entry from the neighbor cache on page 92
- Clear all neighbor cache entries on page 93

View the neighbor cache

To view the neighbor cache, refer to:

- View the entire neighbor cache on page 88
- View a specific entry in the neighbor cache on page 89
- View a specific interface in the neighbor cache on page 90

View the entire neighbor cache

Action
To view the neighbor cache, use this command syntax: $ **server_ip** *<movername>* **-neighbor -list** where: *<movername>* = name of the Data Mover Example: To view the neighbor cache on server_2, type: **$ server_ip server_2 -neighbor -list**

Output

```
server_2:
Address                     Link Layer Address  Interface       Type   State
fe80::204:23ff:fead:4fd4     0:4:23:ad:4f:d4    cge1_0000_ll    host   STALE
fe80::216:9cff:fe15:c00      0:16:9c:15:c:0     cge1_0000_ll    router STALE

fe80::216:9cff:fe15:c00      0:16:9c:15:c:0     cge4_0000_ll    router STALE

3ffe::1                      0:16:9c:15:c:10     cge3_0000_ll    host
REACHABLE
```

where:

Address = Neighbor IPv6 address

Link layer address = Link-layer address of the neighbor

Interface = Name of the interface connecting to the neighbor

Type = Type of the neighbor (host or router)

State = State of the neighbor entry, such as REACHABLE, INCOMPLETE, STALE, DELAY, PROBE

REACHABLE	The neighbor was reachable recently.
INCOMPLETE	Address resolution is in progress.
STALE	The neighbor is no longer reachable.
DELAY	The neighbor is no longer reachable, and probes have been delayed briefly to allow upper layer protocols to provide reachability confirmation.
PROBE	The neighbor is no longer reachable, and unicast Neighbor Solicitation probes are being sent to verify reachability.

View a specific entry in the neighbor cache

Action

To view a specific entry in the neighbor cache, use this command syntax:

```
$ server_ip <movername> -neighbor -list <v6addr>
```

where:

<movername> = name of the Data Mover

<v6addr> = global unicast address (required).

Example:

To view fe80::216:9cff:fe15:c00 in the neighbor cache on server_2, type:

```
$ server_ip server_2 -neighbor -list fe80::216:9cff:fe15:c00
```

Output
```
server_2:
Address                    Link Layer Address Interface       Type   State
fe80::216:9cff:fe15:c00     0:16:9c:15:c:0     cge1_0000_ll    router STALE
```

View a specific interface in the neighbor cache

Action
To view a specific interface in the neighbor cache, use this command syntax:
`$ server_ip <movername> -neighbor -list <v6addr> -interface <ifname>`
where:
<movername> = name of the Data Mover
<v6addr> = global unicast address (required).
<ifname> = name of the interface (Optional). The name can consist of 1 to 63 characters including a-z, A-Z, 0-9, or underscore (_). No leading or trailing spaces are allowed.
Example:
To view cge1_0000_ll in the neighbor cache on server_2, type:
`$ server_ip server_2 -neighbor -list fe80::216:9cff:fe15:c00 -interface cge1_0000_ll`
Output
```
server_2:
Address                    Link Layer Address Interface       Type   State
fe80::216:9cff:fe15:c00     0:16:9c:15:c:0     cge1_0000_ll    router STALE
```

Add an entry to the neighbor cache

To add an entry to the neighbor cache, refer to:

- Add a global unicast address to the neighbor cache on page 90
- Add a link-local address to the neighbor cache on page 91

Add a global unicast address to the neighbor cache

Action
To permanently add a global unicast address to the neighbor cache, use this command syntax:
`$ server_ip <movername> -neighbor -create <v6addr> -lladdr <macaddr> [-interface <ifname>]`

Action
where: *<movername>* = name of the Data Mover *<v6addr>* = global unicast address (required). *<macaddr>* = link-layer address for the neighbor entry (11–17 characters, from 0-ff hex) (required). *<ifname>* = name of the interface (Optional). The name can consist of 1 to 63 characters including a-z, A-Z, 0-9, or underscore (_). No leading or trailing spaces are allowed. Example: To permanently add a global unicast address to the neighbor cache on server_2, type: `$ server_ip server_2 -neighbor -create 2002:8c8:0:2310::2 -lladdr 0:16:9c:15:c:15`
Output
`OK`

To verify the neighbor cache, refer to View the entire neighbor cache on page 88.

Add a link-local address to the neighbor cache

Action

To permanently add a link-local address to the neighbor cache, use this command syntax:
`$ server_ip <movername> -neighbor -create <v6addr> -lladdr <macaddr> [-interface <ifname>]`
where:
<movername> = name of the Data Mover
<v6addr> = link-local address (required).
<macaddr> = link-layer address for the neighbor entry (11–17 characters, from 0-ff hex) (required).
<ifname> = name of the interface (Optional). The name can consist of 1 to 63 characters including a-z, A-Z, 0-9, or underscore (_). No leading or trailing spaces are allowed.
Example:
To permanently add a link-local address to the neighbor cache on server_2, type:
`$ server_ip server_2 -neighbor -create fe80::2 -lladdr 0:16:9c:15:c:12 -interface cge1v6`

Output
`OK`

To verify the neighbor cache, refer to View the entire neighbor cache on page 88.

Delete an entry from the neighbor cache

To delete an entry from the neighbor cache, refer to:

- Delete a global unicast address from the neighbor cache on page 92
- Delete a link-local address from the neighbor cache on page 92
- Delete all addresses from the neighbor cache on page 93

Delete a global unicast address from the neighbor cache

Action

To permanently delete a global unicast address from the neighbor cache, use this command syntax:

```
$ server_ip <movername> -neighbor -delete <v6addr>
```

where:

<movername> = name of the Data Mover

<v6addr> = global unicast address (required)

Example:

To permanently delete a global unicast address from the neighbor cache on server_2, type:

```
$ server_ip server_2 -neighbor -delete 2002:8c8:0:2310:0:2:ac18:f401
```

Output

```
OK
```

To verify the neighbor cache, refer to View the entire neighbor cache on page 88.

Delete a link-local address from the neighbor cache

Action

To permanently delete a link-local address from the neighbor cache, use this command syntax:

```
$ server_ip <movername> -neighbor -delete <v6addr> [-interface <ifname>]
```

where:

<movername> = name of the Data Mover

<v6addr> = link-local address

<ifname> = name of the interface (Optional). The name can consist of 1 to 63 characters including a-z, A-Z, 0-9, or underscore (_). No leading or trailing spaces are allowed.

Example:

To permanently delete a link-local address from the neighbor cache on server_2, type:

Action
`$ server_ip server_2 -neighbor -delete fe80::1 -interface cge1v6`
Output
`OK`

To verify the neighbor cache, refer to View the entire neighbor cache on page 88.

Delete all addresses from the neighbor cache

Action

To permanently delete all addresses from the neighbor cache, use this command syntax:
`$ server_ip <movername> -neighbor -delete -all`
where:
`<movername>` = name of the Data Mover
Example:
To permanently delete all addresses from the neighbor cache on server_2, type:
`$ server_ip server_2 -neighbor -delete -all`

Output
`OK`

To verify the neighbor cache, refer to View the entire neighbor cache on page 88.

Clear all neighbor cache entries

Action

To permanently delete all entries from the neighbor cache, use this command syntax:
`$ server_ip <movername> -neighbor -delete -all`
where:
`<movername>` = name of the Data Mover
Example:
To permanently delete all entries from the neighbor cache on server_2, type:
`$ server_ip server_2 -neighbor -delete -all`

Output
`OK`

To verify the neighbor cache, refer to View the entire neighbor cache on page 88.

Manage Packet Reflect

Packet Reflect ensures that reply packets always leave through the same interface that the request packets entered. Packet Reflect is enabled by default and disabled by modifying the system parameter file for each Data Mover.

When using the server_param command, ensure that the case of parameter and facility names are correct because they are case-sensitive. Packet Reflect on page 17 provides more information.

Action
To disable Packet Reflect on a Data Mover, use this command syntax:
`$ server_param <movername> -facility ip -modify reflect -value 0`
where:
`<movername>` = name of the Data Mover
Example:
To disable the reflect parameter on server_2, type:
`$ server_param server_2 -facility ip -modify reflect -value 0`
Output
`server_2 : done`

View network statistics

Use the server_netstat command to view network statistics by protocol. The server_netstat command displays all the information of a Data Mover. Running the server_netstat command without arguments displays active TCP connections.

Action
To view Data Mover network statistics by protocol, use this command syntax:
`$ server_netstat <movername> -p <protocol>`
where:
`<movername>` = name of the Data Mover
`<protocol>` = option to limit display to the protocol specified
Other options to use with this command:
`-a` = displays the IP, ICMP, TCP, and UDP endpoints
`-i` = displays a summary of the state of the physical devices
`-r` = displays the routing table

Action
-s = displays the per-protocol statistics Example: To view statistics for IP on server_2, type: **$ server_netstat server_2 -s -p ip**
Output
ip: *** 1222024 total packets received 0 bad header checksums 0 with unknown protocol 0 fragments received 0 fragments dropped (dup or out of space) 0 fragments dropped after timeout 0 packets reassembled 2 packets forwarded 0 packets not forwardable 0 no routes 1222030 packets delivered 1157661 total packets sent 0 packets fragmented 0 packets not fragmentable 0 fragments created

Modify Control Station networking

The network on the Control Station is configured at system initialization. If your network environment changes, you might need to modify network settings on the Control Station by using some or all of these tasks:

- Using Unisphere
- Using nas_cs command
- Editing the Control Station configuration files directly

If you are changing the Control Station IP address, but remaining on the same network, then the SP IP addresses for an integrated model need not be modified. However, if you are changing to a different network, then the SP IP addresses need to be modified to be on the same physical network as the Control Station for the Integrated model. Use the clariion_mgmt -modify -network command to update the IP addresses on the SP, as it will also update the files and VNX database with the modified IP addresses.

The tasks to modify Control Station networking are:

- Set the Control Station IPv4 address and network mask on page 96
- Set the Control Station IPv4 gateway on page 97
- Modify the Control Station hostname on page 97

- Improve TCP performance on page 100

Set the Control Station IPv4 address and network mask

1. Log in to the Control Station as root.
2. Change the IPv4 address and network mask by using this command syntax:

```
# nas_cs -set -ip4address <ipv4_address> -ip4netmask<ipv4_netmask>
```

where:

<ipv4_address> = new IPv4 address

<ipv4_netmask> = new network mask

Note: A netmask value must not equal 0.0.0.0.

Example:

Change the IPv4 address and network mask by typing:

```
# nas_cs -set -ip4address 172.24.101.255 -ip4netmask 255.255.255.0
```

Output:

```
OK
```

Set the Control Station IPv6 address and network mask

1. Log in to the Control Station as root.
2. Change the IPv6 address and network mask by using this command syntax:

```
# nas_cs -set -ip6address <ipv6_address[/prefix_length]>
```

where:

<ipv6_address[prefix_length]> = new IPv6 address and prefix length. The */prefix_length* option sets the integer value, between 8 and 128, for the prefix length of the IPv6 address of the Control Station.

Example:

Change the IPv6 address and network mask by typing:

```
# nas_cs -set -ip6address 2002:ac18:af02:f4:20e:cff:fe6e:d524/64
```

Output:

```
OK
```

Set the Control Station IPv4 gateway

1. Log in to the Control Station as root.
2. Set the new IPv4 gateway address by using this command syntax:

   ```
   # nas_cs -set -ip4gateway <ipv4_gateway>
   ```

   where:

   <ipv4_gateway> = new IPv4 gateway address

   Example:

   Change the IPv4 gateway address by typing:

   ```
   # nas_cs -set -ip4gateway 128.221.252.0
   ```

   Output:

   ```
   OK
   ```

Set the Control Station IPv6 gateway

1. Log in to the Control Station as root.
2. Set the new IPv6 gateway address by using this command syntax:

   ```
   # nas_cs -set -ip6gateway <ipv6_gateway>
   ```

   where:

   <ipv6_gateway> = new IPv6 gateway address

   Example:

   Change the IPv6 gateway address by typing:

   ```
   # nas_cs -set -ip6gateway 2002:ac18:af02:f4:20e:cff:fe6e:d527
   ```

   Output:

   ```
   OK
   ```

Modify the Control Station hostname

Although you can change the Control Station's hostname by using the CLI, if you plan to use Unisphere for other tasks, EMC strongly recommends to use Unisphere to make this change.

CAUTION If you change the Control Station's hostname by using the CLI and plan to use Unisphere to manage VNX system, ensure that you use this procedure. If you do not, you cannot create file systems using Unisphere.

1. Add the new hostname to either DNS or VNX system. To make changes to DNS, check the DNS documentation. This procedure deals with VNX system only.

   Log in to the Control Station as root.

2. Verify your current environment:

   ```
   # hostname
   ```

   Output:

   ```
   Eng_1
   ```

3. Display information about the Control Station, including its hostname and ID by typing:

   ```
   # nas_cel -list
   ```

   Output:

   ```
   id  name  owner mount_dev channel net_path      CMU
    0  Eng_1 0                  172.24.101.100 APM04490091900
   ```

4. Open the /etc/hosts file with a text editor. You will see the entry for the current hostname. Add the entry for the new hostname.

   For example, add the new hostname cs100 to the file:

   ```
   172.24.101.100    Eng_1.nasdocs.emc.com Eng_1
   172.24.101.100    cs100.nasdocs.emc.com cs100
   ```

5. Save the file and exit.

6. Ping the new and the old Control Station hostnames by typing:

   ```
   # ping cs100
   ```

   Output:

   ```
   PING cs100.nasdocs.emc.com (172.24.101.100) from 172.24.101.100
   : 56(84) bytes of data.
   64 bytes from Eng_1.nasdocs.emc.com (172.24.101.100): icmp_seq=0
   ttl=255 time=436 usec
   ```

   ```
   # ping Eng_1
   ```

   Output:

   ```
   PING Eng_1.nasdocs.emc.com (172.24.101.100) from 172.24.101.100
   : 56(84) bytes of data.
   64 bytes from Eng_1.nasdocs.emc.com (172.24.101.100): icmp_seq=0
   ttl=255 time=220 usec
   ```

7. Change the hostname on the Control Station by typing:

   ```
   # nas_cs -set -hostname cs100
   ```

8. Verify the new hostname:

```
# hostname
```

Output:

```
cs100
```

9. Change the hostname to the new hostname in the /etc/sysconfig/network file by using a text editor. This makes the hostname permanent when there is a restart.

```
NETWORKING=yes
FORWARD_IPV4=false
GATEWAY=172.24.101.254
GATEWAYDEV=eth3
DOMAINNAME=nasdocs.emc.com
HOSTNAME=cs100
```

10. Save the file and exit.

11. Remove the old hostname from DNS or open the /etc/hosts file with a text editor to delete the old hostname.

    Example:

    There will be only one Control Station hostname entry cs100 in the file after you delete the old hostname:

```
172.24.101.100 cs100.nasdocs.emc.com cs100
```

12. Save the file and exit.

13. Update the local hostname by typing:

```
# nas_cel -update id=0
```

Output:

```
id         = 0
name       = cs100
owner      = 0
device     =
channel    =
net_path   = 172.24.101.100
celerra_id = APM04490091900
```

14. Confirm the hostname of the Control Station by typing:

```
# nas_cel -list
```

Output:

```
id  name owner mount_dev  channel net_path       CMU
0  cs100  0                172.24.101.100  APM04490091900
```

15. Change the SSL certificate for Apache by running:

```
# /nas/sbin/nas_config -ssl
```

Output:

```
Installing a new SSL certificate requires restarting the Apache web
server.
Do you want to proceed? [y/n]: y
New SSL certificate has been generated and installed successfully.
```

16. Refresh the Java server processes by typing:

    ```
    # /nas/sbin/js_fresh_restart
    ```

Improve TCP performance

By default, NewReno, SACK, and FACK algorithms are enabled on Data Movers to improve network performance in the event of data loss. You can disable the NewReno algorithm by using the tcp do_newreno parameter. Similarly, you can disable the SACK and FACK algorithms by using the tcp do_sack parameter. TCP performance on page 20 provides more information.

1. Disable the do_newreno parameter on a Data Mover by using this command syntax:

   ```
   $ server_param <movername> -facility tcp -modify do_newreno -value 0
   ```

   where:

   <movername> = name of the Data Mover

   Example:

   Disable the do_newreno parameter on server_2 by typing:

   ```
   $ server_param server_2 -facility tcp -modify do_newreno -value 0
   ```

   Note: Parameter and facility names are case-sensitive.

   Output:

   ```
   server_2 : done
   ```

2. Disable the do_sack parameter on a Data Mover by using this command syntax:

   ```
   $ server_param <movername> -facility tcp -modify do_sack -value 0
   ```

   where:

   <movername> = name of the Data Mover

   Example:

   Disable the do_sack parameter on server_2 by typing:

   ```
   $ server_param server_2 -facility tcp -modify do_sack -value 0
   ```

   Note: Parameter and facility names are case-sensitive.

   Output:

```
server_2 : done
```

3. Verify that do_newreno and do_sack are disabled by using this command syntax:

   ```
   $ server_param <movername> -facility <facility_name> -list
   ```

   where:

   <movername> = name of the Data Mover

   <facility_name> = name of the facility

   Example:

   Verify that do_newreno and do_sack are disabled by typing:

   ```
   $ server_param server_2 -facility tcp -list
   ```

   Output:

   ```
   server_2 :
   name                facility   default     current   configured
   do_newreno              tcp          1           0            0
   sndcwnd                 tcp          0           0
   fastRTO                 tcp          0           0
   ackpush                 tcp          0           0
   do_sack                 tcp          1           0            0
   backlog                 tcp        100         100
   maxburst                tcp          4           4
   maxStreams              tcp      65535       65535
   ```

After improving TCP performance

The RFCs and proceedings describe these TCP enhancements:

- TCP — RFC 793
- NewReno — RFC 3782
- SACK — RFC 2018 and RFC 2883
- FACK — Forward Acknowledgment: Refining TCP Congestion Control, Proceedings from SIGCOMM'96, August 1996

5 Troubleshooting

As part of an effort to continuously improve and enhance the performance and capabilities of its product lines, EMC periodically releases new versions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes.

If a product does not function properly or does not function as described in this document, contact your EMC Customer Support Representative.

Problem Resolution Roadmap for VNX contains additional information about using EMC Online Support and resolving problems.

Topics included in this chapter are:

- EMC E-Lab Interoperability Navigator on page 104
- Tips for general networking problems on page 104
- Tips for network interface problems on page 104
- Error messages on page 105
- EMC Training and Professional Services on page 105

EMC E-Lab Interoperability Navigator

The EMC E-Lab™ Interoperability Navigator is a searchable, web-based application that provides access to EMC interoperability support matrices. It is available on EMC Online Support at http://Support.EMC.com. After logging in, in the right pane under **Product and Support Tools**, click **E-Lab Navigator**.

Tips for general networking problems

- Check link lights on all NICs and switches.
- Check speed and duplex settings for all interfaces on Data Movers, switches, and clients.
- Check the interface configuration on Data Movers and clients:
  - IP address
  - Netmask
  - Broadcast addresses
- Check connectivity by using the server_ping command from a Data Mover to a client, by using IP addresses to avoid name resolution issues. Check connectivity by using ping from a client to the Data Mover. If ping works in one direction, but not the other, a netmask is probably set incorrectly.
- Check the server_log file for a similar message, indicating that the IP address is in use on the network: 1097151569: IP: 3: Duplicate IP address 172.24.101.37/0:3:47:30:e4:a9
- If the Data Mover and the client are on different subnets, check connectivity between the Data Mover and the router, and between the client and the router.
- Check the output of the server_netstat <movername> -i command for errors on the interfaces.
- Check if the network cables are connected properly.
- Contact EMC Customer Service.

Tips for network interface problems

- If access is not permitted to the internal interfaces, when using the server_ifconfig <movername> <if_name> down command, you receive the error message:

```
server_<x>: interface: invalid interface specified, you might be
 trying to disable one of the internal interfaces.
```

- If a physical device is used by a virtual device and you assign a new name to the interface, you receive the error message:

```
No such device or address. Use the server_sysconfig command to
ensure that the device is present on the Data Mover, and try
creating the interface again.
```

- If the Data Mover fails to reply to certain SNMP requests, you might receive the error message:

```
LOG_SYSLOG: 3: M198 - Unsupported Version 2 in PDU, PDU ignored.
 Agent supports Version 1 only.
```

This error message appears because the SNMP management platform is using a more current version of SNMP than is the SNMP agent. Change the SNMP management platform to SNMP version 1.

Error messages

All event, alert, and status messages provide detailed information and recommended actions to help you troubleshoot the situation.

To view message details, use any of these methods:

- Unisphere software:
  - Right-click an event, alert, or status message and select to view Event Details, Alert Details, or Status Details.
- CLI:
  - Type nas_message -info <MessageID>, where <MessageID> is the message identification number.
- *Celerra Error Messages Guide*:
  - Use this guide to locate information about messages that are in the earlier-release message format.
- EMC Online Support:
  - Use the text from the error message's brief description or the message's ID to search the Knowledgebase on EMC Online Support. After logging in to EMC Online Support, locate the applicable **Support by Product** page, and search for the error message.

EMC Training and Professional Services

EMC Customer Education courses help you learn how EMC storage products work together within your environment to maximize your entire infrastructure investment. EMC Customer Education features online and hands-on training in state-of-the-art labs conveniently located throughout the world. EMC customer training courses are developed and delivered by EMC experts. Go to EMC Online Support at http://Support.EMC.com for course and registration information.

EMC Professional Services can help you implement your system efficiently. Consultants evaluate your business, IT processes, and technology, and recommend ways that you can leverage your information for the most benefit. From business plan to implementation, you get the experience and expertise that you need without straining your IT staff or hiring and training new personnel. Contact your EMC Customer Support Representative for more information.

Appendix A

Linux Services on the Control Station

VNX Control Station Linux 7.2 users with root privileges can use the chkconfig utility for the services currently available on the system.

Topics included are:

- List Linux services on page 108
- Start Linux services on page 108
- Enable Linux services on page 108
- Stop Linux services on page 109
- Disable Linux services on page 109

List Linux services

Action

To list Linux services along with their current state, type:

```
# chkconfig --list
```

Output

This produces a list of services in two parts:

- Services controlled by the /etc/rc.d/init.d scripts
- Services controlled by xinetd

Start Linux services

Action

To start an init.d service, for example the network service, type:

```
# chkconfig network on

# service network start
```

Enable Linux services

Action

To enable an xinetd service, for example the time service, type:

```
# chkconfig time on
```

After enabling Linux services

You can use the chkconfig utility to enable other services. The man pages for the chkconfig command provide additional information about using the --level switch to manipulate the specific run levels of the init.d services.

Stop Linux services

Action

To stop an init.d service, for example the network service, type:
`# chkconfig network off`
`# service network stop`

Disable Linux services

Action
To disable an xinetd service, for example the time service, type: `# chkconfig time off`

After disabling services

You can use the chkconfig utility to disable other services. The man pages for the chkconfig command provide additional information about using the --level switch to manipulate the specific run levels of the init.d services.

Appendix B

VLANs to Support a Standby Data Mover

VLAN tagging enables a single Data Mover to be the standby for multiple primary Data Movers. Each Data Mover can be connected to a single switch or to multiple switches. Some applications of VLANs on VNX are described in:

- Configure a standby Data Mover by using a single switch on page 112
- Configure a standby Data Mover in a multiswitch environment on page 115

Configure a standby Data Mover by using a single switch

You can use a single Gigabit switch to configure a VLAN-supported standby Data Mover, as shown in Figure 1 on page 112.

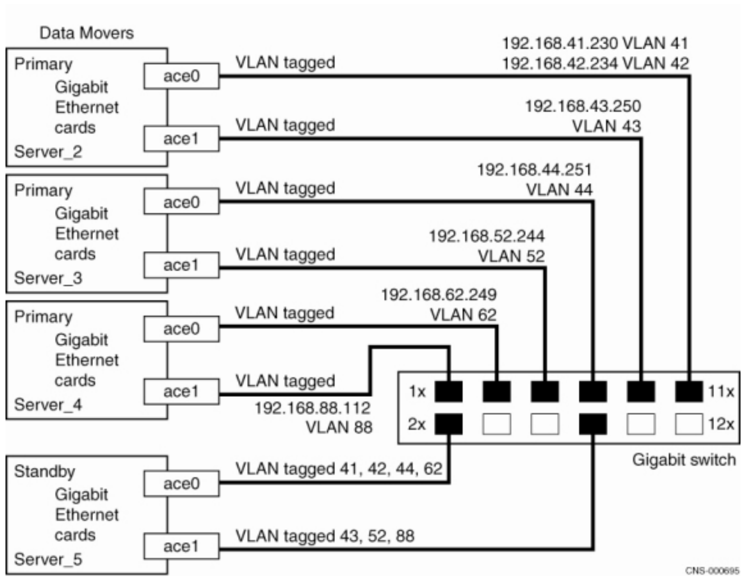

Figure 1. Standby Data Mover by using a single switch

For physical devices on only one network, VLAN tagging can be done either at VNX IP interface or by the network switch (802.1Q).

Note: Either a Layer 3 switch or a router is required for communication between VLANs.

---

1. Configure the interfaces on each of the primary Data Movers by using this command syntax:

   ```
   $ server_ifconfig <movername> -create -Device <device_name> -name <if_name>
   -protocol IP <ipaddr> <ipmask> <ipbroadcast>
   ```

   where:

   <movername> = name of the Data Mover on which you are configuring interfaces

   <device_name> = name of the device you are assigning

   <if_name> = name of the interface you are configuring

   <ipaddr> = IP address to assign to the device

   <ipmask> = subnet address for the device

   <ipbroadcast> = broadcast address for the device

   Example:

Configure the interfaces on server_2, server_3, and server_4 by typing these commands for each primary Data Mover:

```
$ server_ifconfig server_2 -create -Device ace0 -name ace0
-protocol IP 192.168.41.230 255.255.255.0 192.168.41.255
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_2 -create -Device ace0 -name ace0
-protocol IP 192.168.42.234 255.255.255.0 192.168.42.255
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_2 -create -Device ace1 -name ace1
-protocol IP 192.168.43.250 255.255.255.0 192.168.43.255
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_3 -create -Device ace0 -name ace0
-protocol IP 192.168.44.251 255.255.255.0 192.168.44.255
```

Output:

```
server_3: done
```

```
$ server_ifconfig server_3 -create -Device ace1 -name ace1
-protocol IP 192.168.52.244 255.255.255.0 192.168.52.255
```

Output:

```
server_3: done
```

```
$ server_ifconfig server_4 -create -Device ace0 -name ace0
-protocol IP 192.168.62.249 255.255.255.0 192.168.62.255
```

Output:

```
server_4: done
```

```
$ server_ifconfig server_4 -create -Device ace1 -name ace1
-protocol IP 192.168.88.112 255.255.255.0 192.168.88.255
```

Output:

```
server_4: done
```

2. Set the primary Data Mover interfaces to process VLAN traffic by using this command syntax:

```
$ server_ifconfig <movername> <if_name> vlan=<vlanID>
```

where:

<movername> = name of the Data Mover on which the interface exists

<if_name> = name of the interface on which you are changing the VLAN ID

<vlanID> = VLAN ID between 0 (the default, no VLAN tagging) and 4094

Example:

Set server_2, server_3, and server_4 to process VLAN traffic by typing these commands for each primary Data Mover:

```
$ server_ifconfig server_2 ace0 vlan=41
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_2 ace0 vlan=42
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_2 ace1 vlan=43
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_3 ace0 vlan=44
```

Output:

```
server_3: done
```

```
$ server_ifconfig server_3 ace1 vlan=52
```

Output:

```
server_3: done
```

```
$ server_ifconfig server_4 ace0 vlan=62
```

Output:

```
server_4: done
```

```
$ server_ifconfig server_4 ace1 vlan=88
```

Output:

```
server_4: done
```

3. Configure a standby for each primary Data Mover by using this command syntax:

```
$ server_standby <movername> -create mover=<source_movername> -policy <poli
cy_type>
```

where:

<movername> = name of the Data Mover on which the interface exists

<source_movername> = name of the Data Mover to be configured as standby

<policy_type> = Policy option to apply when a fault is detected (auto, retry, or manual)

Example:

Configure server_5 as the standby Data Mover for server_2, server_3, and server_4 by typing these commands:

```
$ server_standby server_2 -create mover=server_5 -policy auto
```

Output:

```
server_2 : server_5 is rebooting as standby
```

```
$ server_standby server_3 -create mover=server_5 -policy auto
```

Output:

```
server_3 : server_5 is rebooting as standby
```

```
$ server_standby server_4 -create mover=server_5 -policy auto
```

Output:

```
server_3 : server_5 is rebooting as standby
```

Configure a standby Data Mover in a multiswitch environment

You can configure a standby Data Mover to provide failover for one of several primary Data Movers in a multiswitch environment. Ensure that the interswitch links in a multiswitch environment pass the appropriate VLAN IDs, as shown in Figure 2 on page 115.

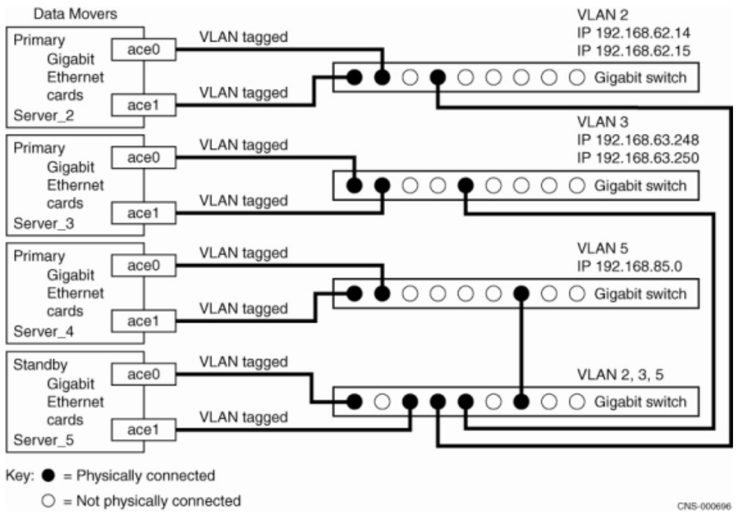
Data Movers
Primary Gigabit Ethernet cards Server_2
ace0
ace1
VLAN tagged
VLAN 2
IP 192.168.62.14
IP 192.168.62.15
Gigabit switch
Primary Gigabit Ethernet cards Server_3
VLAN 3
IP 192.168.63.248
IP 192.168.63.250
Primary Gigabit Ethernet cards Server_4
VLAN 5
IP 192.168.85.0
Standby Gigabit Ethernet cards Server_5
VLAN 2, 3, 5
Key: ● = Physically connected
○ = Not physically connected
CNS-000696

Figure 2. Standby Data Mover in a multiswitch environment

Note: Either a Layer 3 switch or a router is required for communication between VLANs.

1. Configure the interfaces on each of the primary Data Movers by using this command syntax:

```
$ server_ifconfig <movername> -create -Device <device_name> -name <if_name>
-protocol IP <ipaddr> <ipmask> <ipbroadcast>
```

where:

`<movername>` = name of the Data Mover

`<device_name>` = name of the device you are assigning

`<if_name>` = name of the interface you are configuring

`<ipaddr>` = IP address to assign to the device

`<ipmask>` = subnet address for the device

`<ipbroadcast>` = broadcast address for the device

Example:

Configure the interfaces on server_2, server_3, and server_4 by typing these commands:

```
$ server_ifconfig server_2 -create -Device ace0 -name ace0

-protocol IP 192.168.62.14 255.255.255.0 192.168.62.255
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_2 -create -Device ace1 -name ace1

-protocol IP 192.168.62.15 255.255.255.0 192.168.62.255
```

Output:

```
server_2: done
```

```
$ server_ifconfig server_2 -create -Device ace0 -name ace0

-protocol IP 192.168.63.248 255.255.255.0 192.168.63.25
```

Output:

```
server_3: done
```

```
$ server_ifconfig server_3 -create -Device ace1 -name ace1

-protocol IP 192.168.63.250 255.255.255.0 192.168.63.255
```

Output:

```
server_3: done
```

```
$ server_ifconfig server_3 -create -Device ace0 -name ace0

-protocol IP 192.168.85.22 255.255.255.0 192.168.85.255
```

Output:

```
server_4: done
```

```
$ server_ifconfig server_4 -create -Device ace1 -name ace1

-protocol IP 192.168.85.45 255.255.255.0 192.168.85.255
```

Output:

```
server_4: done
```

2. Set the Data Mover interfaces to process VLAN traffic by using this command syntax:

   `$ server_ifconfig <movername> <if_name> vlan=<vlanID>`

   where:

   `<movername>` = name of the Data Mover on which the interface exists

   `<if_name>` = name of the interface on which you are changing the VLAN ID

   `<vlanID>` = VLAN ID between 0 (the default, no VLAN tagging) and 4094

   Example:

   Set server_2, server_3, and server_4 to process VLAN traffic by typing these commands for each Data Mover:

   ```
   $ server_ifconfig server_2 ace0 vlan=2
   ```

   Output:

   ```
   server_2: done
   ```

   ```
   $ server_ifconfig server_2 ace1 vlan=2
   ```

   Output:

   ```
   server_2: done
   ```

   ```
   $ server_ifconfig server_3 ace0 vlan=3
   ```

   Output:

   ```
   server_2: done
   ```

   ```
   $ server_ifconfig server_3 ace1 vlan=3
   ```

   Output:

   ```
   server_2: done
   ```

   ```
   $ server_ifconfig server_4 ace0 vlan=5
   ```

   Output:

   ```
   server_2: done
   ```

   ```
   $ server_ifconfig server_4 ace1 vlan=5
   ```

   Output:

   ```
   server_2: done
   ```

3. Set each primary Data Mover by using this command syntax:

   `$ server_standby <movername> mover=<source_movername> -policy <policy_type>`

   where:

   `<movername>` = name of the Data Mover on which the interface exists

   `<source_movername>` = name of the Data Mover to be configured as standby

<policy_type> = Policy option to apply when a fault is detected (auto, retry, or manual)

Example:

Configure server_5 as the standby Data Mover for server_2, server_3, and server_4 by typing these commands for each primary Data Mover:

```
$ server_standby server_2 mover=server_5 -policy auto
```

Output:

```
server_2 : server_5 is rebooting as standby
```

```
$ server_standby server_3 mover=server_5 -policy auto
```

Output:

```
server_3 : server_5 is rebooting as standby
```

```
$ server_standby server_4 mover=server_5 -policy auto
```

Output:

```
server_4 : server_5 is rebooting as standby
```

Appendix C

Switch-Specific Guidelines for VLANs

The configuration guidelines for several vendor-specific network switches that support IEEE 802.1Q encapsulation are provided in this section.

Note: Ensure to reference each switch vendor's specific documentation for detailed configuration steps and cautions.

Topics included are:

- Configure VLANs on the Alteon 180 switch on page 120
- Configure VLAN tagging at the switch on page 120
- Configure VLAN tagging at VNX IP interface on page 121

Configure VLANs on the Alteon 180 switch

Table 4 on page 120 provides guidelines for configuring VLAN tagging for the Alteon 180 switch.

Table 4. Using 802.1Q encapsulation to set up VLANs on the Alteon 180 switch (Alteon 180 code version: 5.2.13)

Configuration steps	Menu commands
1. Assign one or more switch ports to a VLAN.	Create the needed VLAN by using the VLAN command in the Configure menu. Add port.
2. Verify the port VLAN membership, duplex mode, and speed.	Configure, VLAN, Current, /, Configure, Port.
3. Set the port to Tag.	Configure, Port, Tag.
4. Verify tagging configuration, port speed, and duplex mode.	Configure, Port, Current.

Configure VLAN tagging at the switch

Using 802.1Q for VLANs on Cisco 4000, 5000, or 6000 series switches (Cisco Catalyst 6509 NmpSW 7.3(2)) provides guidelines for configuring VLAN tagging at the switch (port-based tagging).

1. Assign one or more switch ports to a VLAN by using this command syntax:

   ```
   set vlan vlan_num mod_num/port_num
   ```

   Example:

   ```
   set vlan 560 4/10
   ```

2. Verify the port VLAN membership, duplex mode, and speed by using this command syntax:

   ```
   show vlan [vlan_num]
   ```

   Example:

   ```
   show vlan 560
   ```

   or

   ```
   show port [mod_num[/port_num]]
   ```

   Example:

   ```
   show port 4/10
   ```

Configure VLAN tagging at VNX IP interface

Using 802.1Q for VLANs on Cisco 4000, 5000, or 6000 series switches (Cisco Catalyst 6509 NmpSW 7.3(2) Code version needs to support 802.1Q) provides guidelines for configuring VLAN tagging at VNX IP interface.

1. Configure a 8021.Q trunk by using this command syntax:

   ```
   set trunk mod_num/port_num [on] dot1q
   ```

   Note: The Cisco set trunk command enables an 802.1Q VLAN. This command differs from VNX trk (trunk) option used to configure Ethernet channels and link aggregation.

   Example:

   ```
   set trunk 2/9 on dot1q
   ```

2. Verify the trunking configuration by using this command syntax:

   ```
   show trunk [mod_num/port_num]
   ```

   Example:

   ```
   show trunk 2/9
   ```

Appendix D

Configuring a Link-Local Address

You can manually configure a unique link-local address if your network configuration requires it. A link-local address always begins with a prefix fe80. The address must be unique or an error message appears.

- Configure a link-local address on page 124

Configure a link-local address

Action

To create a link-local address for a network device, use this command syntax:
 	
`$ server_ifconfig <movername> -create -Device <device_name> -name <if_name> -protocol IP6 <link_local_addr>`
 	
where:
 	
`<movername>` = name of the Data Mover where the device is located
 	
`<device_name>` = name of the network device
 	
`<if_name>` = name for the interface created by this command
 	
`<link_local_addr>` = IP address for the interface
 	
Example:
 	
To configure a link-local address for device cge1 with IPv6 address on server_3, type:
 	
`$ server_ifconfig server_3 -create -Device cge1 -name cge1_link_local -protocol IP6 fe80::260:16ff:fe0b:1234`

Output
`server_3 : done`

Verify a link-local address

Action

To verify that the link-local address is correct, use this command syntax:
 	
`$ server_ifconfig <movername> -all -ip6`
 	
where:
 	
`<movername>` = name of the Data Mover on which to verify the interfaces
 	
Example:
 	
Verify that the link-local address you configured is correct. To verify the settings for the cge1 interface for server_3, type:
 	
`$ server_ifconfig server_3 -all -ip6`

Output

```
server_3 :
el30 protocol=IP device=mge0
        inet=128.221.252.3 netmask=255.255.255.0 broadcast=128.221.252.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:b:b7:18 netname=lo-
calhost
el31 protocol=IP device=mge1
        inet=128.221.253.3 netmask=255.255.255.0 broadcast=128.221.253.255

        UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:b:b7:17 netname=lo-
calhost
loop6 protocol=IP6 device=loop
         inet=::1 prefix=128
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
loop protocol=IP device=loop
         inet=127.0.0.1 netmask=255.0.0.0 broadcast=127.255.255.255
         UP, Loopback, mtu=32768, vlan=0, macaddr=0:0:0:0:0:0 netname=lo-
calhost
cge1_link_local protocol=IP6 device=cge1
         inet=fe80::260:16ff:fe0b:1234 prefix=64
         UP, Ethernet, mtu=1500, vlan=0, macaddr=0:60:16:b:ed:89
```

After you finish

When you configure a global address after configuring a link-local address, another link-local address is not automatically created. Configure an IPv6 interface on page 37 provides more information about how to configure a global address.

Glossary

A

Address Resolution Protocol (ARP)
Protocol that translates IP addresses to MAC addresses.

B

broadcast
Data frame or packet transmitted to every node on the local network segment.

D

device
Two types of devices are physical (port) and virtual.

See also *virtual device*.

F

Fast Ethernet
Any Ethernet specification with a speed of 100 Mb/s. Based on the IEEE 802.3u specification.

frame tagging
Information added to a frame so the receiving device can identify the virtual local area network (VLAN) to which it belongs. IEEE 802.3q is a frame-tagging standard.

G

gateway
VNX for file that is capable of connecting to multiple systems, either directly (direct-connected) or through a Fibre Channel switch (fabric-connected).

gateway address
Host or IP address of the gateway machine through which network traffic is routed.

Gigabit Ethernet
Any Ethernet specification with a speed of 1000 Mb/s. IEEE 802.3z defines Gigabit Ethernet over fiber and cable, which has a physical media standard of 1000Base-X (1000Base-SX short wave, 1000Base-LX long wave) and 1000Base-CX shielded copper cable. IEEE 802.3ab defines Gigabit Ethernet over an unshielded twisted pair (1000Base-T).

I

Internet Control Message Protocol (ICMP)
Internet protocol that reports errors and provides control information related to IP-packet processing.

Internet Protocol (IP)
Network layer protocol that is part of the Open Systems Interconnection (OSI) reference model. IP provides logical addressing and service for end-to-end delivery.

Internet Protocol address (IP address)
Address uniquely identifying a device on any TCP/IP network. Each address consists of four octets (32 bits), represented as decimal numbers separated by periods. An address is made up of a network number, an optional subnetwork number, and a host number.

IP interface
Named logical entity representing a physical device (a port) or a virtual device (a combination of physical devices) and used to assign an IP address to the device. There may be multiple IP interfaces associated with a single device.

J

jumbo frames
Changes an Ethernet frame's Maximum Transmission Unit (MTU) from the IEEE 1500 bytes up to 9000 bytes. Jumbo frames should only be used when the packet to/from any combination of Ethernet devices can be handled without any Layer 2 fragmentation or reassembly.

L

link
Working data connection between systems on a network. Also, a connection between two or more ports.

M

Maximum Transmission Unit or Maximum Transfer Unit (MTU)
Largest frame size transmitted over the network. Messages longer than the MTU must be divided into smaller frames. The Layer 3 protocol (including IP, IPX) extracts the MTU from the Layer 2 protocol (including Ethernet), fragments the messages into that frame size, and makes them available to the lower layer for transmission.

media access control (MAC) address
Data link hardware address every physical device (port) or virtual device needs to connect to a network segment. These addresses are used by network devices to locate the logical devices.

multicast
Communication between one sender and multiple receivers. Multicast messages are sent to a defined subset of network addresses.

N

network interface card (NIC)
Circuit board that provides network communication capabilities to and from a computer system.

O

OpenLDAP
Open-source implementation of an LDAP-based directory service.

P

port
Physical connection point to a network or a number used at the transport layer to track host-to-host virtual circuits.

R

routing information protocol (RIP)
Routing protocol optimized for creating routes within one organization (interior gateway protocol). RIP is a distance vector protocol that uses hop count (max 15) as the metric. RIP-1 does not send the mask in updates. RIP-2 sends the mask in updates.

S

subnet
Any network that is part of a larger IP network as identified by the IP address and subnet mask.

subnet mask
32-bit address mask used in IP to identify the bits of an IP address used for the subnet address.

T

Transmission Control Protocol (TCP)
Connection-oriented transport protocol that provides reliable data delivery.

U

User Datagram Protocol (UDP)
Connectionless transport protocol in the TCP/IP stack that allows datagrams to be exchanged without acknowledgements or a delivery guarantee.

V

variable length subnet mask (VLSM)
Means of specifying different subnet masks for the same network on various subnets. VLSM maximizes the use of IP address space.

virtual device
Combination of multiple physical devices defined by a single MAC address.

virtual local area network (VLAN)
Group of devices physically residing on different network segments, but communicating as if they resided on the same network segment. VLANs are configured by management software and are based on logical connections as compared to physical connections for increased administrative flexibility.

Index

A

Address Resolution
    ARP table 86
address types
    link-local 23
    unicast 23
addressing
    IPv6 21
ARP table
    physical address
        MAC address 85
    viewing 86

C

chkconfig, Control Station Linux services 108
Control Station
    chkconfig command 108
    linuxconf 95

D

Data Mover
    configuring standbys using VLANs 19
    creating multiple IP interfaces 34, 36
    disabling RIP processing 59
    verifying connectivity of
        server_ping 50
    viewing ARP table 86
default link-local interface 23
device support
    IPv4 and IPv6 26, 39
directly connected routing 16
Duplicate Address Detection (DAD) 25
dynamic routing 16

E

EMC E-Lab Navigator 104
error messages 105

F

FACK algorithm 20
failover, using VLANs for 115
file system
    events 64
Forward Acknowledgement (FACK) algorithm 20

I

IEEE802.1Q 19
IP interface
    changing parameters for 65
    configuring 32, 34, 36
    deleting 65
    enabling
        RIP processing 84
    modifying VLAN tagging 67
    multiple interfaces for a device 34, 36
    speed and duplex 104
IP interface, deleting 66
IPv4 support 26
IPv6 addressing 21
IPv6 support 26

L

link-local address 23
    automatic creation 23
    default link-local interface 23

link-local address *(continued)*
    manual configuration 23
linuxconf, modifying Control Station network settings 95

M

MAC address
    for ARP table 85
messages, error 105
MTU
    defaults 48
    definition 48
    limits
        jumbo frames 48

N

Neighbor Discovery process 23
network
    statistics, viewing 94
    virtual local area networks (VLANs) 19
network device
    IP interfaces 16
    monitoring 64
    name prefixes
        default name 15
    types 15
NewReno algorithm 20

P

Packet Reflect
    and high-availability devices 18
    and routing 17
    multiple default routes 18
    param ip reflect 94
    using with VLANs 17
port status 28
prefixes 23

R

RIP
    disabling processing
        on a Data Mover 59
    enabling processing on an IP interface 84
    flushing the routing table 81
    processing status 59
    routed daemon 16
routed daemon
    and routing table 16
    definition 16
router configuration 24
routing
    and Packet Reflect 17
    automatic default router configuration 24
    directly connected 16
    dynamic 16
    static 16
routing table
    adding an entry 58
    definition
        when used 16
    deleting an entry 78
    removing all static routes 81
    RIP 16
    RIP and flushing the table 81
    routed daemon 16

S

SACK algorithm 20
Selective Acknowledgement (SACK) algorithm 20
server_arp
    viewing ARP table 86
server_ifconfig
    configuring IP interfaces 32, 34, 36
    deleting an IP interface 65
    modifying VLAN tagging 67
server_netstat
    IP interface errors 104
    viewing network statistics 94
server_ping, verifying Data Mover connectivity 104
server_rip
    displaying RIP status 59
    enabling processing 84
server_route
    default route, adding 55
    deleting a routing table entry 78
    removing all static routes 81
    routing table entry, adding 58
server_setup, disabling RIP processing 59
SNMP traps 64
standby Data Mover, and VLANs 19
static routing 16

T

TCP performance
    FACK algorithm 20
    improving in lossy networks 20
    NewReno algorithm 20
    SACK algorithm 20

troubleshooting 103

U

unicast address 23

V

verifying
    Data Mover connectivity 50
virtual device
    configuring IP interfacefor 31
VLAN (virtual local area network) 10, 19, 20, 67, 115
    and configuring standby Data Movers 19
    and switch configuration 20
    exporting file systems 10
    failover using 115
    frame tagging 19
    modify tagging from an IP interface 67
    protocol 19
VLSM (variable-length subnet mask)
    using 16