Chapter 1
Routing XNS

1

This chapter describes how to configure your router to perform routing for packets following the Xerox Network Systems (XNS) stack of protocols.

You will find information about the following topics and tasks:

- How to configure a routing process for XNS routing. This includes information on the principal XNS protocols, XNS addressing, how to configure your router to route XNS traffic, managing security issues, and maximizing performance.
- How to configure helper addresses for broadcast traffic.
- How to set up routes, including setting metrics.
- How to configure options for access lists and filters.
- Configuration restrictions and requirements for encapsulation of all important lower layer protocols, including Token Ring, FDDI, Ethernet, and others.

This chapter also contains configuration information on Ungermann-Bass' and 3Com's XNS-derived protocols.

The section "XNS Configuration Examples" later in this chapter includes examples of actual configurations for working XNS networks, including Ungermann-Bass and 3Com. For additional information on configuring access lists in 3Com networks, please consult the Application Note titled *A Detailed Look at Access Lists in 3Com XNS.*

Cisco's Implementation of XNS

Cisco provides a subset of the XNS protocol stack to support XNS routing on its routers. The same Cisco routers that route XNS can also route another protocol stack like TCP/IP or DECnet. At the physical and data link layers, XNS traffic can be routed over Ethernets, FDDI, Token Rings, or point-to-point serial lines running HDLC or LAPB.

DECnet

In previous releases of Cisco's routing software, it was not possible to run DECnet Phase IV and any of the XNS family of protocols simultaneously in a router that included both Ethernet and Token Ring interfaces. This restriction was removed in Release 8.2. There are no changes to the syntax of any configuration commands.

Ethernet and Token Ring

When XNS routing is enabled, the address is either the IEEE-compliant address specified in the XNS routing configuration command, or the first IEEE-compliant address in the system. The address is also used as the node address that non-LAN media (notably serial links) use for their XNS node addresses. This address is then used as the default XNS node address on non-LAN nodes.

XNS was originally designed by the same company (Xerox) that developed Ethernet, so it was designed to run over Ethernet. If you implement an XNS stack over Token Ring at the first two layers, you have to encapsulate the lower layers differently than you would for Ethernet. Encapsulation defines the kind of envelope (layer 1 and layer 2 bits) in which three through seven of the XNS protocol and data packet are wrapped prior to being transmitted. Because there is no agreed-upon mechanism for encapsulating XNS packets on a Token Ring network, many vendors have invented their own encapsulation methods. We will discuss this in detail in the section "Configuring XNS Over Token Ring" later in this chapter.

XNS Addresses

Both Data Link (MAC) and Network Layer addressing is needed in any network that supports routing; XNS is no exception. An XNS Network Layer address is composed of three fields.

- The network number uniquely identifies a network in an internet.
- The host number uniquely identifies a host on a network.
- The socket number uniquely identifies a socket within the operating system of the host. A socket is a transport address that is the source or destination of packets.

Network and Host Numbers

The network number is expressed in decimal format in Cisco configuration files and routing tables. When configuring a Cisco router, enter the network number in decimal.

Addresses must be unique throughout an XNS internet. Since both the network number and the host address are needed to deliver traffic to a host, addresses are usually given as network numbers, followed by host addresses, separated with dots. An example address follows.

Example:

```
47.0000.0c00.23fe
```

Here, the network number is 47 (decimal), and the host address is *0000.0c00.23fe* (hex).

Socket Numbers

An XNS socket number is a 16-bit field in the IDP header. Sockets are selected by the client processes of each host before a connection is established. Certain socket numbers are considered to be well-known sockets (WKS), which means that the service performed by the software using them is statically defined. Each system element supplying a specific well-known service does so at the same WKS. Socket numbers above the well-known range are arbitrary, which means that they can be selected and reused at random.

- A socket number of zero means all.
- A socket number of all ones (0xFFFF hex) means unknown.
- Well-known socket numbers range from 1 to 0x0BB8 hex (3000 decimal). All other socket numbers may be dynamically assigned and reused.

Configuring XNS

Follow these steps to configure XNS routing:

Step 1: Enable XNS routing using the **xns routing** command.

Step 2: Assign a unique XNS network number to each interface, using the **xns network** command.

Step 3: Optionally configure performance parameters and helper addresses (which help you manage broadcast traffic).

Step 4: Optionally configure access lists and filters.

Additionally, EXEC-level commands for monitoring and debugging the XNS network are available. These commands are described in the last few sections of this chapter, along with concise summaries of the global and interface-specific configuration commands.

Enabling XNS Routing

The first step in the configuration process is to specify XNS as the protocol you are enabling. Use the **xns routing** global configuration command to enable XNS routing. The full syntax of this command follows.

xns routing [*address*]

no xns routing

The optional argument *address* is the router interface's complete address, which is expressed in hexadecimal format. The **no xns routing** disables all XNS processing.

Example:

In the example below, an interface whose address is *0123.4567.abcd* is enabled for XNS routing.

```
xns routing 0123.4567.abcd
```

If the argument *address* is omitted, the router will use the first IEEE-compliant (Token Ring, FDDI, or Ethernet) interface hardware address it finds.

Your next step is to use the **xns network** interface subcommand to assign a decimal XNS network number to an interface and enables that interface to run XNS protocols. The full syntax of the command follows.

xns network *number*

no xns network

The argument *number* is the network number, in decimal format.

Interfaces not enabled to run XNS ignore any XNS packets that they receive. Every XNS interface in a system must have a unique XNS network number.

Example:

This example starts the routing process with no specific address specified, so the router will use the first IEEE-compliant interface hardware address it finds, then specifies network number 20.

```
xns routing
xns network 20
```

Configuring Static Routing

To add a static route from your router to a remote destination in the XNS routing table, use the **xns route** global configuration command. The full syntax follows.

xns route *network host-address*

no xns route *network host-address*

The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and a hexadecimal host number, separated by a dot.

Example:

The following example sets up a host (router) address of *51.0456.acd3.1243* as the static recipient of packets destined for network 25.

```
xns network 51
xns route 25 51.0456.acd3.1243
```

Managing Throughput

You have several options for managing throughput while dynamically routing. You can set up multiple paths and send packets over these paths in a round-robin fashion. You can also enable or disable the route cache. You can even adjust how often your router uses RIP to send routing table updates to its neighbors provided all the neighbors are Cisco routers.

Setting Multiple Paths

XNS allows your router to select from multiple paths to a destination in order to increase throughput in the network. The default assumes that the router will pick one best path and send all traffic on this path. You can tell your router to compile two or more paths that have equal cost (hop count in XNS' case) and balance the traffic load across all the available paths.

To set the maximum number of multiple paths, use the **xns maximum-paths** global configuration command. The full syntax follows.

xns maximum-paths *paths*

no xns maximum-paths

The argument *paths* is the number of paths to be assigned. The default value for *paths* is 1. Packets are distributed over the multiple paths in round-robin fashion on a packet-by-packet basis. To disable multiple paths, use the command with the default value of one. The **no xns maximum-paths** command restores the default.

The EXEC command **show xns route** displays the entire routing table, so you can examine your additional routes and the maximum path cost for each. See the section "Monitoring an XNS Network" when you are ready to use this command.

Example:

The following example asks the router to send packets over two alternate paths, if available.

```
xns maximum-paths 2
```

Enabling XNS Fast Switching

XNS fast-switching achieves higher throughput by using a cache created by previous transit packets. Fast-switching also provides load sharing on a per-packet basis. As soon as you enable XNS routing, fast switching is automatically enabled as well. Use the **xns route-cache** interface command to enable or disable fast switching. The full syntax of the command follows.

xns route-cache

no xns route-cache

Use the **no xns route-cache** command to disable fast switching and then the **xns route-cache** interface subcommand to re-enable fast switching.

Adjusting Timers

RIP sends routing table updates to its neighbor routers every 30 seconds, unless you specify some other time interval. You can reset the routing update timers on a per-interface basis using the **xns update-time** interface subcommand:

xns update-time *seconds*

XNS routing timers are affected by the value set for the *seconds* argument:

- XNS routes are marked invalid if no routing updates are heard within six times the value of the update timer.
- XNS routes are removed from the routing table if no routing updates are heard within eight times the value of the update timer.
- The granularity of the update timer is determined by the lowest value defined.
- The minimum is ten seconds.

If you want to return to the default condition, you cannot use a **no** variation of the command. Use the **xns update-time** command with a value for the *seconds* argument of 30.

Note: Be careful with this command. Use it only in an all-Cisco environment. Make sure that all timers are the same for all routers attached to the same network segment.

The EXEC command **show xns route** displays the value of these timers. See the section "Monitoring an XNS Network" when you are ready to use this command.

Configuring XNS Over Token Ring

There is no standard way of packaging an XNS packet for transit across an 802.5 Token Ring. XNS was designed to exist above Ethernet at the lowest two network layers. The general process of wrapping new header information around a packet so that it can traverse an unfamiliar network is called *encapsulation*. Each Token Ring vendor has its own method of encapsulating XNS packets. Cisco supports the encapsulation methods of LAN vendors Ungermann-Bass, 3Com, and IBM.

Use the **xns encapsulation** interface subcommand to select the encapsulation method:

xns encapsulation *keyword*

If your Token Ring is an IBM installation, you use the default *keyword* **snap**. Other options are:

- **ub** for Ungermann-Bass
- **3com** for older 3Com Corporation products. Some 3COM 3+ hosts do not recognize Token Ring packets with the source-route bridging RIF field set. You can work around this discrepancy by using the **no multiring xns** interface subcommand on Token Ring interfaces that are used for 3Com XNS routing. See the "Configuring Source-Route Bridging" chapter for more information.

Example:

In the following example, XNS is enabled, an interface is defined for Token Ring, and then Ungermann-Bass is specified as the encapsulation option, as shown:

```
xns routing
xns ub-routing
interface tokenring 0
xns network 1234
xns encapsulation ub
```

You will find more information on the Ungermann-Bass version of XNS in the following section.

Configuring Ungermann-Bass Net/One XNS

Ungermann-Bass's Net/One protocols were derived from the protocols of the Xerox Network Systems (XNS) stack and are very similar to them. However, Net/One is not equivalent to standard XNS. The following are the important differences between the Net/One implementation and standard XNS:

- Net/One routers use an Ungermann-Bass proprietary routing protocol instead of standard XNS RIP. Although they generate both Ungermann-Bass and standard RIP update packets, Net/One routers only listen to Ungermann-Bass updates. Cisco supports generation of Ungermann-Bass updates, and Cisco routers can interoperate with Ungermann-Bass routers in certain restricted configurations. Cisco routers do not presently listen to Ungermann-Bass updates, and illegal configurations can lead to routing loops.
- Net/One routers send periodic HELLO packets, which hosts use to discover the addresses of their local routers. Ordinary XNS hosts use RIP for this purpose. Cisco routers can be configured to generate Ungermann-Bass HELLO packets.
- Net/One equipment uses a non-XNS protocol for network software downloads. During the downloading process, XNS network numbers are embedded in the packets of this protocol. Ungermann-Bass routers pass the booting protocol from network to network, and modify the embedded network numbers. Cisco equipment does not understand the Net/One booting protocol, and Net/One NIUs cannot be booted through Cisco routers in XNS routing mode.

- The Ungermann-Bass Net/One Network Resource Monitor (a network management and monitoring tool) uses XNS packets whose destination host addresses are specific nodes, but whose destination network addresses are broadcast (network -1). These packets are sent as MAC-layer broadcasts, and are expected to be flooded throughout the XNS internet. Cisco does not support flooding of these packets, and this prevents the NRM from doing node discovery through a Cisco router.
- Net/One equipment uses a set of proprietary network management protocols. Cisco routers do not participate in these protocols.

Be aware that:

- The network topology must contain no Ungermann-Bass routers interconnected by media slower than Ethernets, or the topology may contain no loops which pass through both Cisco and Ungermann-Bass routers.
- The Network Resource Monitor cannot be used to manage a network through a Cisco router.
- No Cisco routers may fall in the path between a Net/One NIU and the Network Management Console from which that NIU downloads its software.

Ungermann-Bass Routing Protocol

The routing protocol used by Net/One routers is a distance-vector or Bellman-Ford protocol, similar but not identical to standard XNS RIP. The major difference between the two protocols lies in the metrics used. While standard RIP uses a hop count to determine the best route to a distant network, the Ungermann-Bass protocol uses a path delay metric. The standard RIP protocol maintains information only about hop counts, while the Ungermann-Bass protocol maintains information both about hop counts and about its own metrics. Ungermann-Bass routers generate standard RIP updates by extracting the hop-count values from the Ungermann-Bass routing protocol. Cisco routers generate Ungermann-Bass updates by computing a delay value from the hop count maintained by standard RIP, assuming that all hops are Ethernet LANs. Ungermann-Bass routers gather information only from Ungermann-Bass updates, while Cisco routers gather information only from standard RIP.

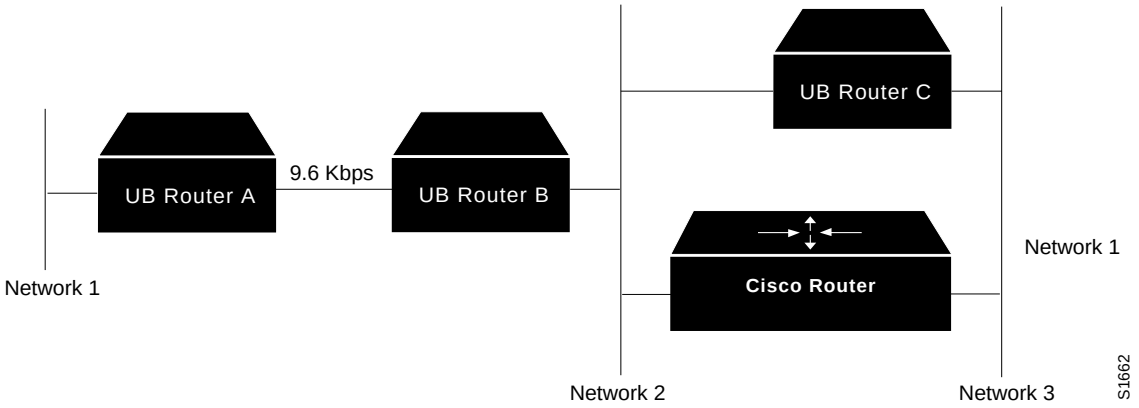
Ungermann-Bass routers do not implement the *split horizon* routing optimization; that is, they will advertize a route to a network through the interface via which packets for that network would actually be sent. Cisco software earlier than version 8.1(25) likewise did not implement split horizon for the Ungermann-Bass proprietary routing protocol. This caused packet forwarding loops in even very simple configurations. The forwarding loops typically manifested themselves as complete failures of the Ungermann-Bass routers.

Cisco software later than 8.1(25) does implement split horizon for the Ungermann-Bass routing protocol, and thus avoids forwarding loops in simpler configurations. Some stable forwarding loops are still possible, however, and transient forwarding loops caused by changes in the network topology (typically when links fail) are easy to create.

Avoiding Routing Loops

An example of a configuration in which both routing instability and a forwarding loop will occur is the following:

Figure 1-1 Sample Ungermann-Bass Routing Configuration



Here, Ungermann-Bass router *B* learns that it has a path to network *1* via Ungermann-Bass router A, in one hop, with a very high delay metric (introduced by the 9.6Kbps line). It re-advertises that route on network *2*, and the Cisco router learns that it can reach network *1* via Ungermann-Bass *B*. It advertises that route on network *3* using both standard RIP and the Ungermann-Bass routing protocol. Because of the metric conversions, the delay field in the Ungermann-Bass routing packets generated on network *3* by the Cisco router shows the delay of two Ethernet hops, rather than the delay of one Ethernet hop and one 9.6 Kbps hop.

Since this delay is less than the metric being advertised by Ungermann-Bass *B*, Ungermann-Bass *C* learns that the route to network 1 is via the Cisco's network *3* interface. It advertises this route onto network *2*, with a Ungermann-Bass delay metric equivalent to three Ethernet hops, and still much less than the delay of the 9.6 Kbps line. Ungermann-Bass *B* therefore switches its route to point to Ungermann-Bass *C*.

At this point, any packet destined for network *1* will be sent by Ungermann-Bass *B* to Ungermann-Bass *C*, which will send it to the Cisco router. The Cisco router will send it back to Ungermann-Bass *B*, and so forth until the packet exceeds the XNS maximum hop count of 16.

On the next update cycle, the Cisco router will learn the route through Ungermann-Bass *C* instead of through Ungermann-Bass *B* (since *B*'s hop count will now be greater than C's), and there will be a tight forwarding loop between the two. This will continue until the routes count to infinity and time out, at which point the process will begin again.

The best way to avoid problems like these is to avoid loops in the network topology which contain both Cisco and Ungermann-Bass routers. Since Cisco routers will assign Ethernet-level delays to all hops, an alternate strategy is to avoid links slower than Ethernets between Ungermann-Bass routers.

Configuring Ungermann-Bass Net/One Routing

To enable Ungermann-Bass Net/One routing, use the **xns ub-routing** global configuration command. The full syntax of this command follows:

xns ub-routing

no xns ub-routing

This command causes HELLO packets and routing updates in Ungermann-Bass format to be sent out through all the interfaces on which XNS is enabled. In addition, it changes the way some XNS broadcast packets are flooded. In normal operation, when XNS broadcast flooding is configured, packets with zero destination network numbers are flooded. With **xns ub-routing** in effect, such packets are not flooded.

There is an Ungermann-Bass configuration example in the "XNS Configuration Examples" section of this chapter.

Helper Addresses and Broadcast-Forwarding

You need to decide how you want the router to handle different kinds of broadcast packets. This is an excellent opportunity for you to reduce unnecessary network traffic, if you think about all your options carefully.

Many broadcasts occur when a node first becomes active on the network. A host will generate an Address Look-up packet when it does not know the current address of whatever other host is supposed to receive its next packet—the local server, for instance. It is generally not a good idea to place a router between users and the servers that carry their primary applications; you should minimize internet traffic. However, if you need that server configuration for some other reason, you need to ensure that users can broadcast between networks without cluttering the internet with unnecessary traffic.

Normally, a packet is sent to a specific network or series of networks. A flooded broadcast packet, with a destination network number of -1, is sent to every network. By default, flooding is off.

You can configure your routers to block all broadcasts and that may be an appropriate option in some circumstances. You have more than simply this all-or-nothing choice, however, by using the xns helper-address and forward-protocol commands.

Using Helper Addresses

The **xns helper-address** interface subcommand causes flooded (all-nets) broadcasts to be forwarded to another address that you specify. This address is called a helper address. The full syntax of this command follows:

xns helper-address *host-address*

no xns helper-address *host-address*

The argument *host-address* is a dotted combination of the network and host addresses as explained in the **xns route** command.

Here is how different kinds of broadcasts are handled:

- Broadcasts specifically sent to another network are routed to that network, and then broadcasted. This means broadcast packets with a network address that is *not* any of the following:
  - All ones
  - Zero
  - The local network
- Broadcasts to all networks (network number of all ones) are forwarded regardless of protocol.
- Broadcasts directed to devices on the same connected network (network address equals the local network number or zero) are checked to see if the protocol type matches an entry in the forward protocol lists and, if a match is found, are forwarded.

Figure 1-2 Helper Addresses

Ignore the protocol-filtering issue for a moment and just consider some examples of how helper-addresses are used. In Figure 1-2, the E0 interface has a helper address set, with the helper on network 12, available through the E2 interface. Some broadcast packets are being received on this E0 interface:

- A broadcast packet with a network address of 5 will be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 0 will also be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 13 will be sent through the E1 interface directly to network 13. It will not be sent to the helper address.

All Nets Broadcasts

To configure the interface for "all nets broadcast flooding," define the XNS **helper-address** for the interface as:

```
xns helper-address -1.FFFF.FFFF.FFFF
```

If you use this address, your router will flood packets with a destination network address of -1 coming in on that interface. Packets will flood to all networks except the source network. Although flooding always creates some duplicates, packets will not loop forever.

Forwarding Specific Protocols

The **xns forward-protocol** global configuration command allows you to specify which XNS protocols will be forwarded when the router receives a broadcast packet. The interface must already have an XNS helper address set. The full syntax of the command is shown below:

xns forward-protocol *type*

no xns forward-protocol *type*

The argument *type* is a decimal number corresponding to an appropriate XNS protocol. See the documentation accompanying your XNS implementation to determine the protocol type number.

Use the **no xns forward-protocol** command and the appropriate argument to remove this function.

In Figure 1-2, the E0 interface is set to a helper address and forwarding for protocol *1* only. (The protocol numbers will vary depending on your XNS implementation.) Broadcast packets are *arriving* on the E0 interface from network *5*:

- A broadcast packet destined for network *5* and protocol *1* will be sent to the helper address.
- A broadcast packet destined for network *5* and another protocol is discarded because it fails the protocol test.
- A broadcast packet destined for network *0* and protocol *1* is sent to the helper address.
- A broadcast packet destined for network *0* and a protocol other than *1* is discarded.

Please note that:

- A broadcast packet destined to network *12* is sent out on the E2 interface to network *12*. This has nothing to do with the helper-address or protocol forwarding.
- A broadcast packet destined to a network of all-ones is flooded to networks *12* and *13* on the E1 and E2 interfaces, regardless of protocol type. It is not sent to the broadcast address on network *5*, because it originated on network *5* and all-nets broadcasts do not flood onto the originating network.

Example:

In this example, a helper address corresponding to the local host is specified and protocol type 2 is forwarded to that host.

```
interface ethernet 0
xns helper-address  26.FFFF.FFFF.FFFF
xns forward-protocol 2
```

Configuring XNS Access Lists and Filters

You may configure XNS access lists to filter traffic on XNS interfaces. If you are specifically interested in 3Com access lists, please consult the Application Note titled *A Detailed Look at Access Lists in 3Com XNS.*

XNS access lists are numbered from 400 to 499 and filter on the source and destination addresses only. This means that they can prevent traffic from going to specific hosts or coming from specific hosts. Extended XNS access lists are numbered from 500 to 599 and filter on XNS protocol and socket fields. The extended filters can prevent entire classes of packets (SPP, Echo, and so on) from passing a router interface and they can also filter traffic going to or coming from specific processes.

Configuring XNS Access Lists

To configure an access list, use the **access-list** global configuration command, with the following syntax:

access-list *number* {**deny**|**permit**} *XNS-source-network*.[*source-address*[*source-mask*]]*XNS-destination-network*.[*destination-address*[*destination-mask*]]

no access-list *number*

Note: For typographic reasons access list command examples are shown on multiple lines; it must be on a single line when given as a configuration command.

The argument *number* must be a number between 400 and 499.

The only required parameter for standard XNS access lists is the XNS source network address. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered. (Note that XNS uses MAC addresses as the node ID; see the examples for further clarification.)

Example 1:

This example denies access from source network *–1* (all XNS networks) to destination network 2.

```
access-list 400 deny -1 2
```

Example 2:

This example denies access from XNS source address *21.0000.0c00.1111*. The destination network does not make any difference:

```
access-list 400 deny 21.0000.0c00.1111
```

Example 3:

This example denies access from all hosts on network *1* that have a source address beginning with *0000.0c*:

```
access-list 400 deny 1.0000.0c00.0000 0000.00ff.ffff
```

Example 4:

In this example, we deny access from source address *11.1622.15* on network *21* to destination address *01D3.020C.0022* on network *31*:

```
access-list 400 deny 21.011.1622.0015 0000.0000.0000 31.01D3.020C.0022
0000.0000.0000
```

Configuring Extended Access Lists

For extended access lists, the **access-list** command again must be typed on one line using this syntax (the negative form of the command is also included):

access-list *number* {**deny**|**permit**} *xns-protocol source-network*.[*source-address* [*source-mask*]] *source-socket destination-network*.[*destination-address* [*destination-mask*]] *destination-socket*

no access-list *number*

The argument *number* must be a number between 500 and 599.

The source and destination addresses and masks are optional. The protocol number *xns-protocol* is the only required parameter. A network number of –1 matches all networks; a socket number of 0 matches all sockets.

Example 1:

To deny access to packets with protocol *1* from source network *1*, source socket 1234 that are trying to be routed to destination network *2*, destination socket *1234*:

```
access-list 500 deny 1 1 1234 2 1234
```

Example 2:

This example adds masks for the source and destination networks:

```
access-list 500 deny 1 21.110011.1622.001500.0000.0000 31.01D3.020C.0022
0000.0000.0000. 1234
```

Filtering Outgoing Packets

An XNS access list group number is assigned with the **xns access-group** interface subcommand. The full syntax of this command follows.

xns access-group *number*

no xns access-group *number*

The argument *number* specifies the access list number defined by the XNS global **access list** command. This command causes all XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If the packet fails the access list, it is not transmitted, but is discarded instead.

Example:

This example uses the **xns access-list** command to deny access to protocol *1* from source network *1*, source socket *1234* to destination network *2*, destination socket *1234*, then assign number 500 to a group to be filtered:

```
access-list 500 deny 1 1 1234 2 1234
!
interface ethernet 0
xns access-group 500
```

Filtering XNS Routing Updates

This section describes the filtering commands that use access lists to control what routing information is accepted, or passed on, within XNS networks. The commands filter incoming traffic and outgoing routing information, and specific routers.

Each access list entry contains only one address parameter and the list must be in the range 400 to 499. How this address is interpreted is defined by the command that will use the list.

As with all other Cisco access lists, an implicit *deny everything* is defined at the end of the list. If this is not desired, an explicit *permit everything* definition must be included at the end of the list. Complex configuration examples are shown in the "XNS Configuration Examples" section.

Input Filters: Adding to the Routing Table

To control which networks are added to *your* router's routing table, use the **xns input-network-filter** interface subcommand.

xns input-network-filter *access-list-number*

no xns input-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In this example, access list 476 controls which networks are added to the routing table when RIP packets are received. The address in the access list is the address of the network that you want to be able to receive routing updates about.

```
access-list 476 permit 16
interface ethernet 1
xns input-network-filter 476
```

This set of configuration commands assures that network 16 is the only network whose information will be added to the routing table from Ethernet 1.

Output Filters: Controlling the List of Networks

To control the list of networks that are sent out in routing updates by your router, use this interface subcommand:

xns output-network-filter *access-list-number*

no xns output-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In the following example, access list 496 controls which networks are sent out in routing updates. The second line specifies interface serial 1, and the third line causes network *27* to be the only network advertised in routing update packets. Information about other networks will not be advertised in routing updates (for the specified interface only; other interfaces may advertise the full routing table).

```
access-list 496 permit 27
interface serial 1
xns output-network-filter 496
```

Router Filters

To control the list of routers from which data will be accepted, use the **xns router-filter** interface subcommand:

xns router-filter *access-list-number*

no xns router-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In this example, access list 466 defines the only router that data will be accepted from. In this case, the address parameter is the address of a router.

```
access-list 466 permit 26.0000.000c0.047d
interface serial 0
xns router-filter 466
```

Information from a disallowed router is ignored.

XNS Configuration Examples

This section includes examples of common configurations, designed to help you put all the specific command information into a complex, real-world configuration file. We start with basic configuration examples and move on to protocol forwarding, helper address and both simple and complex access lists. Some of the examples refer to 3Com or Ungermann-Bass XNS, rather than standard XNS; all examples are clearly marked.

Creating a Routing Process

The following example establishes XNS routing on a Cisco router (creating a routing process), then three interfaces are named and given their individual network numbers.

Example:

```
xns routing
!
interface ethernet 0
xns network 1
!
interface ethernet 1
xns network 44
!
interface serial 1
xns network 23
```

Setting Timers

This example creates a routing process by specifying a specific address. Next we specify the interfaces and give them network numbers. The update timers for the serial and the Ethernet interfaces have also been changed. The granularity for the Ethernet interface becomes 20 because that is the lowest value specified for that protocol.

Example:

```
xns routing 0000.0C53.4679
!
interface ethernet 0
xns network 1
xns update-time 20
!
interface serial 0
xns network 51
xns update-time 40
!
interface ethernet 1
xns network 27
xns update-time 25
```

Configuring for Multi-Protocol Routing

What do you do if you want to enable XNS and another protocol as well? This example illustrates one way to do this. Do a pencil copy of your proposed configuration commands and check them against the other protocol chapters in this manual before you proceed with configuring more than one protocol in a session.

Example:

```
xns routing
novell routing
interface ethernet 0
xns network 200
novell network 4e
interface ethernet 1
xns network 205
novell network 6bb
interface serial 0
xns network 301
novell network 4ad
```

Configuring for Ungermann-Bass Routing

In this example, basic Ungermann-Bass Net/One routing is enabled by first enabling XNS routing, then specifying Ungermann-Bass routing, then defining the interfaces and networks.

Example:

```
xns routing
xns ub-routing
interface ethernet 0
xns network 1234
interface ethernet 1
xns network 567
```

3Com Access List

This example permits and denies specific services between networks 1002 and 1006 in a 3Com network. Echo and error packets can go from 1002 to 1006, as well as all SPP and PEP (normal data traffic). However, all NETBios requests are denied. The final three lines are blanket permissions for RIP, SPP and PEP, because access lists will assume you wish to deny anything you do not specifically permit. These blanket permissions must come at the end of the list, as shown in the example configuration.

Example:

```
access-list 524 permit 2 1002 0x0000 1006 0x0000
!  permit Echo from 1002 to 1006
access-list 524 permit 3 1002 0x0000 1006 0x0000
!  permit Error from 1002 to 1006
access-list 524 deny 5 -1 0x0000 -1 0x046B
!  deny all NetBIOS
access-list 524 permit 4 1002 0x0000 1006 0x0000
!  permit PEP from 1002 to 1006
access-list 524 permit 5 1002 0x0000 1006 0x0000
!  permit SPP from 1002 to 1006
access-list 524 permit 1
!  permit all RIP
!
!These are needed if you want PEP and SPP to be permitted from
!networks other than 1002
access-list 524 permit 4
!  permit all PEP
access-list 524 permit 5
!  permit all SPP
```

There are additional access list examples in the Application Note titled *A Detailed Look at Access Lists in 3Com XNS.*

Monitoring an XNS Network

Use the EXEC commands described in this section to obtain displays of activity on your XNS network.

Displaying Cache Entries

Use the **show xns cache** command to displays a list of fast-switching cache entries. Enter this command at the EXEC prompt:

show xns cache

In the following sample output, the router responds to a request for cache information with a version number.

```
XNS routing cache version is 14
```

Displaying Interface Parameters

Use the **show xns interface** command to display interface-specific XNS parameters. Enter this command at the EXEC prompt:

show xns interface [*name*]

The optional argument *name* may be used to request a display a particular interface.

The following sample output shows a display of all interface statistics.

```
Ethernet 0 is up, line protocol is up
XNS address is 60.0000.0c00.1d23
xns encapsulation is ARPA
Helper address is 912.ffff.ffff.ffff
Outgoing address list is not set
INput filter list is not set
Output filter list is not set
Router filter list is not set
Update timer is not set
XNS fast-switching enabled

Ethernet 1 is administratively down, line protocol is down
XNS protocol processing disabled

Serial 1 is up, line protocol is up
XNS protocol processing disabled
```

In this display, the Ethernet 0 interface has a helper address set but all other parameters are set to the defaults. The Update Timer refers to itself as not set when it is set to default value. The Ethernet 1 interface is down and XNS processing is disabled. The serial 1 interface is up but it is not processing XNS packets.

For the Ethernet 1 and serial 1 interfaces, the **no xns network** command is in force. You can enable XNS processing by using the **xns network** configuration command.

Displaying the Routing Table

Use the EXEC command **show xns route** to display the entire XNS routing table. Enter this command at the EXEC prompt:

show xns route *network-number*

The optional network number argument specifies a particular network.

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 1 learned routes

Maximum allowed path(s) are/is 1
C Net 14 is directly connected, 0 uses, Ethernet0
C Net 15 is directly connected, 0 uses, Ethernet1
R Net 16 [1/0] via 14.0000.0c00.3e3b,  10 sec, 0 uses, Ethernet0
```

In this display, RIP-derived are indicated by the letter R. Networks that are reachable are listed in numerical order within each category—RIP or connected, in this case.

The routing table also tells you the address of the router that constitutes the first hop in the route (always the same router in this case), the round-trip delay encountered on the route and the interface that the route is available through.

Displaying Traffic Statistics

Use the EXEC command **show xns traffic** to display packet statistics, including packets sent, received, and forwarded. Enter this command at the EXEC prompt:

show xns traffic

Sample output follows. Table 1-1 describes the fields displayed.

```
Rec: 3968 total, 0 format errors, 0 checksum errors, 0 bad hop count, 3968 local
destination,  0 multicast
Bcast: 2912 received, 925 sent
Sent:  5923 generated, 500 forwarded, 0 encapsulation failed, 0 not routable
Errors:  10 received, 20 sent
Echo:  Recd:  100 requests, 89 replies   Sent:  20 requests, 20 replies
Unknown:  5 packets
```

Table 1-1 XNS Traffic Statistics Field Descriptions

Field	Description
Rec:	The total number of packets received on the interface.
format errors	Number of packets received with format errors in the header; they were discarded.
checksum errors	Number of packets received and discarded because of checksum error.
bad hop count	Number of packets discarded because the hop count field was equal to or greater than 16.
local destination	Number of packets received on the interface that had a local MAC address.
multicast	Number of packets received with a multicast list address.
B cast:	Number of broadcast packets received and sent (both directed and flooded).
Sent:	
generated	Total number of packets sent out on the interface.
forwarded	Number of packets sent to another router for forwarding to a remote network.
encapsulation failed	Number of packets discarded because encapsulation was needed (token ring, for example) and couldn't be accomplished.
not routable	Number of packets bridged or discarded because they did not conform to any protocol this router could route.
Errors:	Number of Error packets sent and received.
Echo:	Number of Echo packets received and sent, specifying how many replies were received.
Unknown:	Number of packets that failed for a reason not listed above; the packet conformed to nothing the router understood.

Debugging an XNS Network

Use the EXEC commands described in this section to troubleshoot and monitor your XNS network. For each **debug** command listed, there is a corresponding **undebug** command that turns off the message logging.

debug xns-packet

The command **debug xns-packet** enables logging of XNS packet traffic, including the addresses for source, destination, and next hop router of each packet.

debug xns-routing

The command **debug xns-routing** displays XNS routing transactions.

XNS Global Configuration Command Summary

Following is an alphabetically arranged list of the XNS global configuration commands.

access-list *number* {**deny**|**permit**} *XNS-source-network*.[*source-address*[*source-mask*]]*XNS-destination-network*.[*destination-address*[*destination-mask*]]

Configures an XNS access list. The argument *number* is an access list number in the range 400 and 499. The keyword **permit** or **deny** specifies the filtering action. The arguments *XNS-source-network* and *XNS-destination-network* are the only required addresses. The source and destination masks are optional.

access-list *number* {**deny**|**permit**} *XNS-protocol source-network*.[*source-address* [*source-mask*]] *source-socket destination-network*.[*destination-address* [*destination-mask*]] *destination-socket*

Configures an extended XNS access list. The argument number is an access list number in the range 500 and 599. The argument *XNS-protocol* is the XNS protocol number. See previous description for remaining argument and keyword descriptions.

no access-list *number*

Removes the specified access list.

[**no**] **xns forward-protocol** *type*

Determines which protocol types will be forwarded when a broadcast is received on an interface that has an XNS helper address set. The argument *type* is a decimal number corresponding to an appropriate XNS protocol.

[**no**] **xns maximum-paths** *paths*

Sets the maximum number of equal-cost paths the router will use. The default value of *paths* is one.

[**no**] **xns route** *network host-address*

Adds a static route to the XNS routing table. The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and the hexadecimal host number.

[**no**] **xns routing** [*address*]

Enables XNS routing. The optional argument *address* is the XNS address the router is to use. If the argument *address* is omitted, the first Token Ring, FDDI, or Ethernet interface hardware address found is used. The **no** form of the command disables all XNS packet processing.

[**no**] **xns ub-routing**

Enables or disables Ungermann-Bass Net/One routing.

XNS Interface Subcommand Summary

Following is an alphabetically arranged list of the XNS interface subcommands. These commands follow an **interface** command.

[**no**] **xns access-group** *number*

Assigns an access list number to an interface, and causes all XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If the packet fails the access list, it is not transmitted, but is discarded instead. The argument *number* specifies the access list number.

xns encapsulation *keyword*

Selects encapsulation for a Token Ring interface. Choices for *keyword* are: **snap, ub,** and **3com**.

[**no**] **xns helper-address** *host-address*

Sets a helper address to forward broadcasts. The argument *host-address* is a dotted combination of the network and host addresses.

[**no**] **xns input-network-filter** *access-list-number*

Controls which networks are added to *your router*'s routing table. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

[**no**] **xns output-network-filter** *access-list-number*

Controls the list of networks that are sent out *in routing updates by your router*. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

[**no**] **xns network** *number*

Assigns a decimal XNS network number to an interface.

[**no**] **xns route-cache**

Enables fast-switching. The **no** keyword disables fast switching.

[**no**] **xns router-filter** *access-list-number*

Controls the list of routers from which data will be accepted. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

[**no**] **xns update-time** *seconds*

Sets the XNS routing update timers to the value assigned to the *seconds* argument.