Chapter 35

# AppleTalk

# Introduction

This chapter describes the AppleTalk routing protocol, support for AppleTalk on the router, and how to configure and operate the router to act as a wide area AppleTalk router.

The AppleTalk network architecture was developed by Apple Computer Inc. to provide internetworking of Macintosh computers and other peripheral devices using LocalTalk media, and to allow seamless access to network services such as file servers and printers from the familiar Macintosh desktop environment. The open nature of the architecture has enabled the AppleTalk network system to be extended to support other media types (e.g. EtherTalk for Ethernet media), and a mixture of both Apple and non-Apple network devices on the same AppleTalk network.

Some interface and port types mentioned in this chapter may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the Hardware Reference.

# AppleTalk Protocol Architecture

The AppleTalk protocol architecture is a layered protocol architecture with well defined interfaces between layers (Figure 35-1 on page 35-5). Each protocol makes use of the services provided by a lower layer protocol, to provide an enhanced service to a higher layer protocol.

The AppleTalk protocols can be considered within the framework of the OSI seven-layer Reference Model. The physical layer is represented by AppleTalk network hardware, including LocalTalk, EtherTalk and TokenTalk. The data link layer is represented by the AppleTalk link access protocols — LocalTalk Link Access Protocol (LLAP), EtherTalk Link Access Protocol (ELAP) and TokenTalk Link Access Protocol (TLAP), respectively. The network layer is represented by the Datagram Delivery Protocol (DDP). The transport layer is represented by the Routing Table Maintenance Protocol (RTMP) and the Name Binding Protocol (NBP), while the session layer is represented by the Zone Information Protocol (ZIP).

# AppleTalk Nodes and Networks

An AppleTalk node is any device, such as a personal computer, file server or printer, connected to an AppleTalk network (LAN).

An AppleTalk network can be set up using a range of different media types including LocalTalk™, EtherTalk® and TokenTalk. Different physical networks can be interconnected via routers and switches to create arbitrarily large AppleTalk internets.

## LocalTalk

LocalTalk hardware is built into every Macintosh computer, Apple computer, LaserWriter printer, and many other peripheral devices. LocalTalk networks use 230.4 Kbps twisted pair cable in a bus topology. A single AppleTalk network can span up to 300 metres and support a maximum of 32 devices. The data link layer protocol used to deliver data packets between nodes on a LocalTalk network is the *LocalTalk Link Access Protocol (LLAP)*. LLAP provides a "best effort" delivery service of error free packets. It does not guarantee delivery, but every packet that is delivered is guaranteed to be error free.

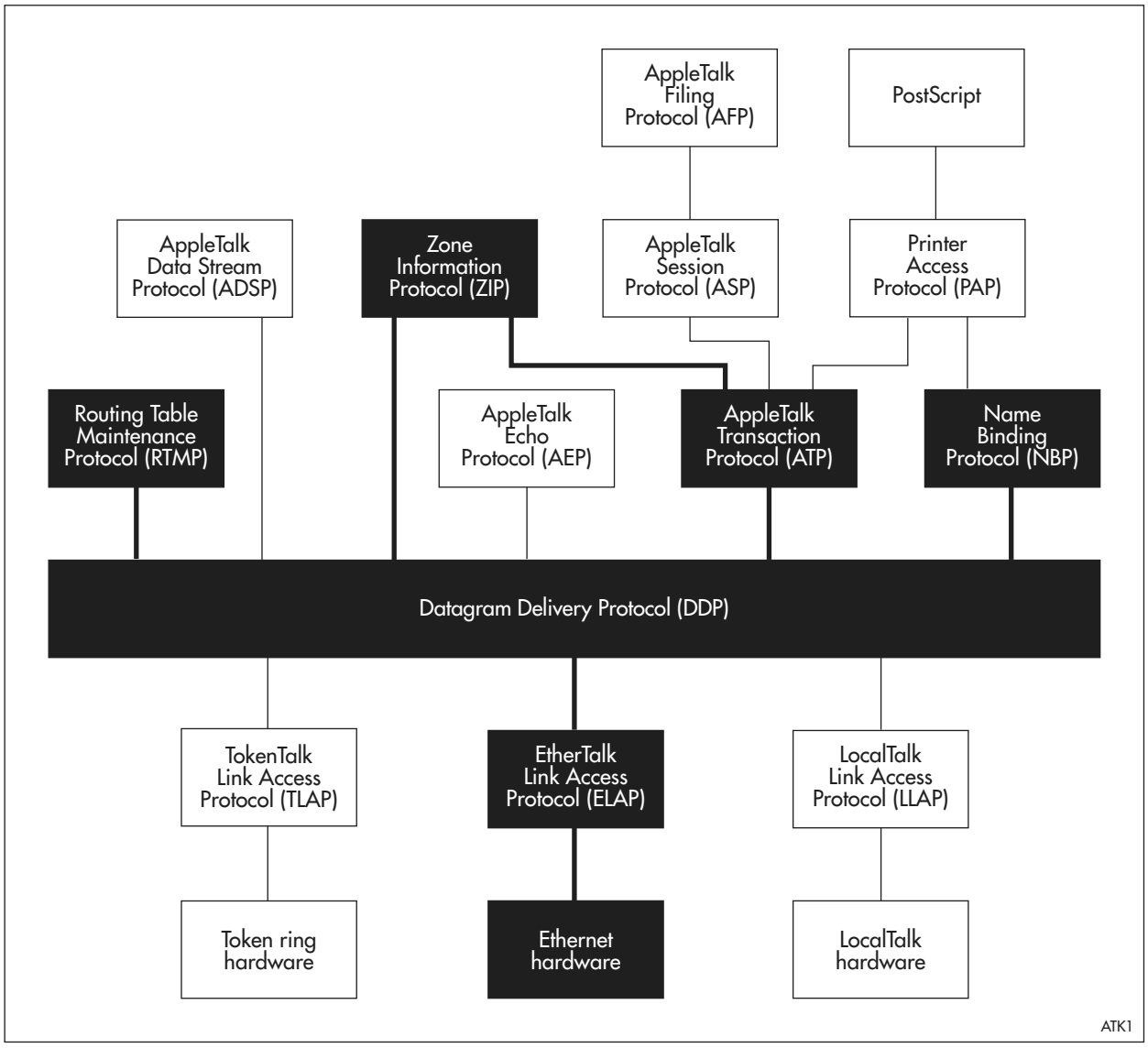
Each node on a LocalTalk network is assigned an 8-bit node identifier number (node ID) as its data link address, using a dynamic assignment mechanism. When a new node is activated, it selects a *provisional* node ID and then verifies the uniqueness of the *provisional* node ID by transmitting an *LLAP Enquiry control* packet to the provisional node address. If the provisional node ID is in use by another node, that node responds with an *LLAP Acknowledge control* packet. The new node then selects another provisional node ID and repeats the process.

If an Acknowledgement control packet is not received after a number of retransmissions of the Enquiry control packet, the provisional node ID is taken to be unique and used until the device is switched off. Node IDs are divided into two classes, user node IDs and server node IDs, to limit the number of enquiry packets required to confirm the provisional node ID (Table 35-1 on page 35-4). Each LLAP packet includes the node IDs of the source and destination nodes, used by the network hardware to ensure the packet is delivered to the correct node.

Table 35-1: Classes of LocalTalk Link Access Protocol node IDs

| Node ID Range | Description |
|---|---|
| 0 | Not allowed (unknown) |
| 1-127 | User node IDs |
| 128-254 | Server node IDs |

Figure 35-1: AppleTalk protocol architecture, with the elements supported by the router highlighted




## EtherTalk and TokenTalk

EtherTalk provides high speed connection of devices using standard 10 Mbps Ethernet technologies. EtherTalk can support as many concurrently active AppleTalk devices as can be connected to an Ethernet network. The data link layer protocol used to deliver data packets between nodes on an EtherTalk network is the *EtherTalk Link Access Protocol (ELAP)*.

TokenTalk provides connection of devices using standard Token Ring technologies. TokenTalk can support as many concurrently active AppleTalk devices as can be connected to a Token Ring network. The data link layer protocol used to deliver data packets between nodes on an TokenTalk network is the TokenTalk Link Access Protocol (TLAP).

Node addressing on EtherTalk and TokenTalk networks differs from node addressing in LocalTalk networks in that the physical hardware addressing schemes support a much larger number of devices, and therefore a physical address cannot be mapped directly to the node ID of an AppleTalk protocol address. Both ELAP and TLAP rely on the AppleTalk Address Resolution Protocol (AARP) to provide translations between hardware addresses and AppleTalk node IDs, and for dynamic address-acquisition.

# AppleTalk Address Resolution Protocol (AARP)

A *protocol stack* running within a node has its own unique *protocol address* to identify the stack among its peers on the internet and to communicate with these peer entities. A protocol address comprises a 16-bit *network number* and an 8-bit *node ID*. On a LocalTalk network, which supports a maximum of 254 devices, the AppleTalk data link address can be used directly as the lower 8 bits of the protocol address without the need for any special address resolution mechanism. On other media, such as EtherTalk, the data link address space is much larger, and cannot be used directly as a portion of the protocol address. The *AppleTalk Address Resolution Protocol (AARP)* provides a mechanism for mapping protocol addresses to hardware addresses. When the protocol stack transmits a data packet it specifies the destination by its protocol address. The data link layer uses AARP to translate the protocol address into the hardware node address of the destination node.

AARP performs three related address resolution functions — translating protocol addresses into hardware addresses; dynamically determining the node's protocol address; and filtering incoming packets.

## Address Translation

On each node, AARP maintains a cache of mappings between protocol and hardware addresses, called the *Address Mapping Table (AMT)*, which it uses to translate a protocol address to a hardware address. When a new address mapping is discovered it is added to the cache. When the cache becomes full, the least recently used entries are removed. Entries are aged to remove obsolete mappings — if an entry is not refreshed within a certain period it is deemed obsolete and removed.

New mappings are discovered by gleaning information from incoming data packets intended for the node's protocol stack (which contain both the hardware and protocol address of the sender), and from responses to specific *AARP Request* packets. When a required hardware address is not found in the AMT, AARP broadcasts an *AARP Request* packet containing the protocol address for which a hardware address is required to all nodes on the network. When a node receives an AARP Request packet containing a protocol address that matches its own protocol address, it returns an *AARP Response* packet containing its own hardware address.

## Dynamic Assignment of Protocol Addresses

On EtherTalk and TokenTalk networks, a protocol stack's address can be assigned dynamically by AARP. AARP selects a provisional address that is not in the AMT, and broadcasts an *AARP Probe* packet. When a node receives a Probe packet containing a provisional protocol address matching its own protocol address, it returns an *AARP Response* packet. When the probing node receives a Response packet, it selects another provisional protocol address and repeats the process. When a Response packet is not received after a specified number of retries, then AARP accepts the provisional address as the node's protocol address.

# Datagram Delivery Protocol

LLAP, ELAP and TLAP provide a best-effort, node-to-node delivery of packets on a single AppleTalk network. The Datagram Delivery Protocol (DDP) extends the node-to-node delivery service of data link layer protocols to a process-to-process, best-effort delivery service across an internet. Processes operating within a node can attach themselves to addressable entities called *sockets*. DDP manages the exchange of packets, called *datagrams*, between any two sockets in an internet. A process attached to a socket is a *network-visible entity (NVE)*, and can be accessed from anywhere in the internet.

A socket is a logical, addressable entity within a node, identified by an 8-bit *socket number*. Sockets 0 and 255 are reserved. Sockets 1 to 127 are statically assigned sockets, reserved for clients such as AppleTalk protocols (NBP, RTMP). Sockets 128 to 254 are assigned dynamically by DDP upon request from client processes in the node. Socket numbers are unique within a node. The *internet socket address*, which combines the network number, node ID and socket number, provides an internet-wide unique address.

Each network in an AppleTalk internet is assigned a unique range of 16-bit *network numbers*. No two networks in an internet may have overlapping ranges. Network number 0 is reserved and by default refers to the local network where the node is attached. Certain node IDs have special meaning to DDP and should not be used as part of an AppleTalk node address. Node ID 255 (0xFF), when used in conjunction with a non-zero network number, defines a network-specific broadcast directed at the specified network. Node ID 255 (0xFF), in conjunction with network number 0, defines a network-wide broadcast. Node ID 0 identifies any router on the network specified by the network number portion of the node address. Node ID 254 (0xFE) is reserved on EtherTalk and TokenTalk networks and should not be used as a node ID.

Every AppleTalk data packet includes a network number within the range of network numbers of the destination network. Routers use the destination network number in the data packet, and information from routing tables, to making routing decisions. The data packet is forwarded from router to router until it reaches the destination network where the appropriate data link protocol delivers the packet to the destination node. Routers use the Routing Table Maintenance Protocol (RTMP) to create and maintain AppleTalk routing tables.

DDP is responsible for acquiring a node's AppleTalk address at startup. This address must be unique across the AppleTalk internet. On a non-extended network (such as LocalTalk) with one network number, each 8-bit AppleTalk node ID is unique. The underlying data link layer (such as LLAP) is used to dynamically assign the node ID and the node's network number is then obtained from a router using an RTMP Request packet. If a router is not present on the network, the network number is set to 0.

On an extended network (such as EtherTalk) with a range of network numbers, nodes are differentiated by unique network number/node ID pairs, and address acquisition is a two stage process. First, a *provisional node address* is obtained through the data link layer. The node ID portion is chosen at random, and the network number portion is chosen from the startup range (0xFF00 to 0xFFFE). The network number and node ID used the last time the node was startup is used as a "hint" or "first guess". The provisional node address is used to communicate with a router on the network and discover the network number range for the network where the node is attached. The node's actual network number and node ID are then obtained through the underlying data link layer.

# Routing Table Maintenance Protocol (RTMP)

A datagram is transmitted from its source socket to its destination socket over the internet by routers. If the destination network number is on the local network, DDP uses the data link layer to deliver the datagram to its destination node. If the destination is not on the local network, DDP uses information from routing tables to forward the datagram to another router on the route to the destination network. At the destination network, the datagram is delivered to the destination node by the local data link layer.

Routers use the Routing Table Maintenance Protocol (RTMP) to establish and maintain the routing tables used by DDP to forward datagrams from any source socket to any destination socket on an internet.

The hardware interfaces on a router in an AppleTalk network are referred to as ports, and are identified by a *port number*, starting with 1. A port may be connected directly to an AppleTalk network (e.g. EtherTalk), or to a wide area link. The port has an associated port descriptor that identifies the port number, the node address of the router, the network number range for the network where the port is attached, and whether the port is attached to an AppleTalk network. Port numbers are assigned dynamically by the router. The port node address and port network number range are relevant for ports connected to an AppleTalk network, and are meaningless for ports connected to a communication link.

A stable routing table contains one entry for each network that can be reached in the internet. The entry includes the port number where packets destined for that network must be forwarded, the node address of the next router where the packet must be sent, the distance to the destination network, and the state of the entry. The distance measure is a hop count, with each hop representing a router on the path to the destination network.

The hop count field in a DDP datagram is set to 0 by the source node. Each router along the path increments the hop count by 1. The hop count is limited to 15. If a router receives a datagram with the hop count field set to 15, and the destination node is not on a network directly connected to the router, the datagram is discarded. This mechanism is used to filter out packets circulating in loops.

When a router is switched on, the routing table is initialised with entries for each AppleTalk port connected to a network with a non-zero network number range. Each router in the internet then periodically broadcasts the contents of its routing table in *RTMP Data* packets through each port. RTMP Data packets received from other routers are used to extend or update the routing table. RTMP Data packets may include routes to new networks, which are added to the routing table, or a better route to a network that is already in the routing table, in which case the routing table is updated with the new information.

To reduce RTMP Data packet size in large internets, AppleTalk uses a *split horizon* algorithm. Entries in the routing table whose forwarding port in the routing table is equal to the port out which the entry is being sent are omitted from the RTMP Data packet.

Routing table entries are aged to purge the table of obsolete or bad entries. A periodic timer, called the *validity timer*, is used to manage this process. Each entry in the routing table is assigned a state of *good*, *suspect*, *bad*, or *bad-again*.

Each time the validity timer expires, all entries in the table have their state downgraded. *Good* entries are marked *suspect*, *suspect* entries are marked *bad*, etc. Entries that are already marked *bad-again* when the timer expires are deleted. The process is arrested by the reception RTMP Data packets. All entries added to the routing table or updated as the result of processing an incoming RTMP Data packet have their state set to *good*.

The propagation of this process through the internet is speeded up by RTMP using a technique called *notify neighbour*. Bad entries are identified in RTMP Data packets by a hop count of 31. When a router receives an RTMP Data packet with such an entry, the state of the entry in the routing table is automatically set to *bad*.

Network number ranges are configured as part of the port descriptors of the router ports and then propagated through the internet by RTMP. On any particular network, one router, the *seed router*, needs to have the network number range set into its corresponding port descriptor. If there is more than one seed router on a network, they must all use the same value for the network number range.

# Name Binding Protocol (NBP)

Network-visible entities (e.g. socket clients) can assign themselves one or more *entity names*. An entity name is a character string of the form `object:type@zone`. Each field may be up to 32 characters long. The `object` field describes the particular object. The `type` field specifies attributes of the object (for example, Mailbox, Printer), and the `zone` field identifies the location of the entity by zone name. Entity names are **not** case-sensitive. Wildcard characters can be used to match multiple names. For the `object` and `type` fields, an equal sign (=) matches all possible values. A single approximately equal sign (≈) matches zero or more characters in an `object` or `type` string. In the `zone` field, an asterisk signifies the default zone for the node specifying the name.

Names identify entities because they are easier for network users to remember and they remain constant. Numeric addresses, however, change when a device is moved or as a result of the dynamic address acquisition process at startup. For example, a portable computer has the same name regardless of where it is connected to an internet, but its address and the routes to the entity change as it is moved around the internet. A personal computer that remains in the same physical location always has the same name, but its address may change each time it is switched on as a result of the dynamic address acquisition process.

AppleTalk protocols rely on numeric addresses. Since an entity's name is constant but its address and routes to the entity can change, a mechanism is required to dynamically translate between names and addresses. This mechanism is provided by the *Name Binding Protocol (NBP)*.

Each node maintains a *names table* of mappings between entity names and internet socket addresses for all entities in that node. NBP is used to lookup an entity's address in the name table. The NBP process uses a statically assigned socket called the *name information socket (NIS)* to accept and service requests.

Large internets can potentially present the user with very long lists of entity names. AppleTalk internets can be subdivided into AppleTalk zones, using the *Zone Information Protocol (ZIP)*. Name lookup can then be restricted to particular zones.

On a single network, name lookup is a relatively simple process. When a client requests a name lookup, the NBP process in the client firsts checks the node's own name table. If a match is not found, NBP uses DDP to broadcast an *NBP Lookup* packet over the network to the NIS. All nodes on the network with an operational NBP process receive the Lookup packet and search their own names table for a match. If a match is found a *NBP Lookup Reply* packet containing the name and internet socket address mapping is returned to the address from which the Lookup packet was received.

On an internet, the lookup process is more complicated and relies on the use of zones and the active participation of routers. When a client requests a name lookup, the NBP process in the client sends a *NBP Broadcast Request* packet to the NIS of the local router, which retransmits the lookup request as a *Forward Request* packet to the NIS in any router directly connected to each network containing nodes in the target zone of the lookup request. When a router receives a Forward Request packet, it converts it to a NBP Lookup packet and broadcasts it to the NIS in all nodes in the target zone on the destination network. The NBP Lookup Reply is returned to the original requester.

# Zone Information Protocol (ZIP)

A *zone* is an arbitrary subset of the AppleTalk nodes in an internet. A particular network may contain nodes belonging to any number of zones, but a particular node may belong to just one zone at a time. Zones are defined in terms of network number ranges and therefore comprise one or more entire networks. A network may be a member of more than one zone, so zones may intersect. The sum of all zones is the entire internet.

Each AppleTalk network has an associated *zones list* that specifies the zone names that may be chosen by nodes on that network during the startup process. One of the zone names in the list is chosen as the default zone.

Routers maintain a complete mapping of all zone names to their corresponding networks in the *zone information table (ZIT)*. The Zone Information Protocol (ZIP*)* manages this process. The ZIP process monitors the routing table managed by RTMP. When a new network appears, ZIP attempts to determine the network's zone list and add it to the ZIT. When a network disappears from the routing table, ZIP removes the network and its zone list from the ZIT.

# AppleTalk on the Router

The router supports AppleTalk Phase 2, and enables EtherTalk (Ethernet LAN) networks to be interconnected using Frame Relay, X.25 wide area and Point-to-Point Protocol (PPP) links. Direct support is provided for the EtherTalk Link Access Protocol (ELAP), the Datagram Delivery Protocol (DDP), the Routing Table Maintenance Protocol (RTMP), the Zone Information Protocol (ZIP), portions of the AppleTalk Transaction Protocol required for ZIP, and the Name Binding Protocol (NBP).

To create and associate an AppleTalk interface or port with a physical interface, use the command:

```
add apple port interface=interface [seed=seed] [demand={on|
   off}]
```

The **seed** parameter is required when the interface is an Ethernet interface and the router must act as the seed router for the AppleTalk network where the port is attached. The value specified is the network number range for the AppleTalk network. To modify the **seed** parameter of an existing AppleTalk port, use the command:

```
set apple port=port [seed=seed] [demand={on|off}]
```

To delete an AppleTalk port, use the command:

```
delete apple port=port
```

If the physical interface is a Frame Relay interface or an X.25 interface, any Data Link Connections (DLCs) and MIOX circuits configured for the respective interface are available and can be used by AppleTalk. However, each DLC or MIOX circuit that is required must be explicitly added to the AppleTalk port, after the port has been added by using the respective commands:

```
add apple dlci=dlci port=port
add apple circuit=circuit port=port
```

To delete individual DLCs or MIOX circuits from the AppleTalk, use the respective commands:

```
delete apple dlci=dlci port=port
delete apple circuit=circuit port=port
```

To display information about the ports, interfaces, DLCs and MIOX circuits assigned to the AppleTalk routing module, use the commands:

```
show apple port[=port]
show apple dlci
show apple circuit
```

Routers play an essential role in the operation of the Zone Information Protocol (ZIP), especially in an AppleTalk internet. For this process to work, each seed router on an AppleTalk network must be configured with the list of zone names defined for the local network by using the commands:

```
add apple zone=zone-name port=port [default]
delete apple zone=zone-name port=port
```

To display a list of currently defined zones, use the command:

```
show apple zone
```

To temporarily disable or enable the AppleTalk routing module without affecting configuration information, use the commands:

```
disable apple
enable apple
```

To display the current state of the AppleTalk module, use the command:

```
show apple
```

To display the state of the AARP cache and the routing table, use the commands:

```
show apple aarp
show apple route
```

# AppleTalk Filtering

AppleTalk filters provide a mechanism for determining whether to process packets received over network interfaces. There are three types of AppleTalk filters:

- Datagram Delivery Protocol (DDP) filters
- Routing Update (RMTP) filters
- Zone (ZIP) filters

A filter is a collection of filter entries, where each filter entry specifies a pattern to match and an action to execute upon a successful match. No two filter entries with the same pattern are allowed in a filter. When a packet matches one of the patterns in the filter, the filter then determines what action to take with the packet.

For each of the three types of AppleTalk filters, 100 filter entries can be added. These filters are numbered from 0 to 99.

Within a filter, each filter entry is identified by an entry number, from 1 to 65535, and filter entries are ordered according to entry number (in increasing order). A filter entry with a low entry number has precedence over a filter entry with a high entry number. For example, a packet is matched against *filter entry1* before being matched against *filter entry2*. A new filter entry can be inserted between existing entries and the entry numbers adjust accordingly.

A filter may be associated with one or more interfaces. However, an interface may have only one type of filter associated with it.

When an interface receives, or is about to transmit a packet, the filter specific to the packet received or about to be transmitted is inspected to see if that packet is allowed to be received or transmitted on that interface.

The search process within the filter is done from the first filter entry until a filter entry with a matching pattern is found or no more filter entries are available. If a matching filter entry is found, action taken by the interface follows the action specified by the filter entry. Otherwise, the filters default action is implemented.

# DDP packet filtering

The Datagram Delivery Protocol (DDP) provides a process-to-process, best-effort delivery service across an internet. DDP or packet filters are configured to match packets against the following filter entries:

- source network number or source network range
- destination network number or destination network range
- source node number or node number range
- destination node number or node number range
- source socket number or source socket number range
- destination socket number or destination socket number range
- direction of the packet, i.e. one of incoming or outgoing

Each filter entry is assigned an action to execute upon a successful packet match – either allow or deny.

If an interface associated with a packet filter receives a packet that does not match any filter entry on the filter, then by default the packet is discarded. This means that in effect every DDP packet filter has an implicit 'discard all' at the end of the entry list.

The application of DDP packet filtering enables the logging and counting of packets allowed or denied by a filter. This allows a network manager to closely monitor packets received by the router.

To add a DDP packet filter to a router use the command:

```
add apple packetfilter=filter-id [entry=entry]
   [snetwork={network-range|any}] [dnetwork={network-range|
   any}] [snode={node-range|any}] [dnode={node-range|any}]
   [ssocket={socket-range|any}] [dsocket={socket-range|any}]
   [direction={in|out}] [log={yes|no|on|off|true|false}]
   action={deny|allow}
```

To modify a filter entry on a DDP packet filter use the command:

```
set apple packetfilter=filter-id entry=entry
   [snetwork={network-range|any}] [dnetwork={network-range|
   any}] [snode={node-range|any}] [dnode={node-range|any}]
   [ssocket={socket-range|any}] [dsocket={socket-range|any}]
   [direction={in|out}] [log={yes|no|on|off}] [action={deny|
   allow}]
```

To delete a filter entry on a DDP packet filter use the command:

```
delete apple packetfilter=filter-id [entry=entry]
```

To display information about a DDP packet filter use the command:

```
show apple packetfilter[=filter-id]
```

# RTMP or Routing Update filtering

The Routing Table Maintenance Protocol (RTMP) manages routing information for AppleTalk networks. RTMP or route filters determine that specific route information is accepted and/or advertised by the RTMP process on a router.

A RTMP packet filter entry is characterised by:

- network number or network number range to match
- direction, either send or receive
- action, either allow or deny

A RTMP filter is associated with one or more interface.

Each time a routing update packet is received on an interface each route listed in the packet is checked against the filter entries associated with that interface. The packet is then either dropped or processed further by the routing table. Note that a route entry in a routing update packet that is exactly the same as a filter entry, or is within the network range of a filter entry, is defined as a match. If a route entry in a routing update packet is beyond the network range of a filter entry this is not a match.

Each time a routing update packet is transmitted to an interface each routing entry advertised by the RTMP process is checked against filter entries associated with that interface before routes are transmitted in a routing update packet.

This filtering mechanism also applies to RTMP reply packets generated by the router as the result of a routing query from either a host and/or router.

If the no matching filter entry is found on a RMTP packet filter then by default the route to be filtered is allowed to be processed, either send or receive.

To add a RTMP packet filter to a router, use the command:

```
add apple routefilter=filter-id [entry=entry]
   network=network-range [direction={in|out}] [log={yes|no|
   on|off}] action={deny|allow}
```

To modify a filter entry on a RTMP packet filter, use the command:

```
set apple routefilter=filter-id entry=entry
   [network=network-range] [direction={in|out}] [log={yes|no|
   on|off}] [action={deny|allow}]
```

To delete a filter entry on a RTMP packet filter, use the command:

```
delete apple zonefilter=filter-id [entry=entry]
```

To display information about a RTMP packet filter, use the command:

```
show apple routefilter[=filter-id]
```

# Zone Filtering

The Zone Information Protocol (ZIP) manages the relationship between network numbers and zone names. ZIP or zone filters prevent hosts accessing higher layer services on other non-routers in other zones.

ZIP filtering does not affect the exchange of zone information between routers; it affects the exchange between routers and hosts. Filtering the exchange of zone information between routers would result in the abnormal operation of routers in the network. To prevent a router from accessing a particular zone(s), implement RMTP filtering for the network associated with that zone(s).

A ZIP filter entry is characterised by:

- zone name to match
- action, either allow or deny

A ZIP filter is associated with an interface. All replies to zone name requests coming from that interface are checked against the filter to see if the zone(s) in the reply are allowed to be advertised or not. If a match with a filter entry is made then the request is either allowed or denied. If no matching filter entry is found, then the zone is advertised in a reply packet.

To add a ZIP packet filter to a router, use the command:

```
add apple zonefilter=filter-id [entry=entry] zone=zone-name
   [log={yes|no|on|off}] action={deny|allow}
```

To modify a filter entry on a ZIP packet filter, use the command

```
set apple zonefilter=filter-id entry=entry [zone=zone-name]
   [log={yes|no|on|off}] [action={deny|allow}]
```

To delete a filter entry on a ZIP packet filter, use the command:

```
delete apple zonefilter=filter-id [entry=entry]
```

To display information about a ZIP packet filter, use the command:

```
show apple zonefilter[=filter-id]
```

# AppleTalk Dial-On-Demand

The router's implementation of AppleTalk supports dial-on-demand routing on AppleTalk ports associated with Point-to-Point Protocol (PPP) interfaces that use an ISDN call or a synchronous port controlling a modem as the physical interface. The PPP interface must be configured with the IDLE parameter set to a value other than OFF. See Chapter 15, Point-to-Point Protocol (PPP) for details of how to configure a PPP interface for dial-on-demand operation.

An AppleTalk port is configured for dial-on-demand operation with the **demand** parameter when the port is created by using the command:

```
add apple port interface=interface demand=on
```

An existing AppleTalk port can be configured for dial-on-demand operation by using the command:

```
set apple port=port demand=on
```

Routing broadcasts are suppressed for a port with the **demand** parameter set to **on**, so that user data is transmitted or received via the port. When there has been no user data transmitted or received within the time period specified by the **idle** parameter for the PPP interface, the PPP module automatically disconnects the WAN connection. When the AppleTalk routing module attempts to transmit data via the AppleTalk port, the PPP module automatically reactivates the WAN connection.

Because no routing information is propagated across an on-demand link, static routes must be added to the ports at each end of the link by using the command:

```
add apple route
```

Static routes can be added to ports associated with any type of interface — Frame Relay, X.25, Point-to-Point Protocol (PPP) and Ethernet. However, adding a static route to an Ethernet port is of limited use because routes originating from routers attached to the same Ethernet segment are always broadcast by the routers.

Static routes can be deleted by using the command:

```
delete apple route
```

The following command displays the contents of the AppleTalk routing table, including both static and dynamically learned routes:

```
show apple route
```

# Extended Ping for AppleTalk

The extended **ping** command on page 22-135 of Chapter 22, Internet Protocol (IP) supports AppleTalk networks and addressing and can be used test the connectivity between two AppleTalk nodes to determine whether each node can "see" the other node. *Echo Request* packets are sent to the destination address and responses are recorded. The command:

```
ping [[appleaddress=]network.node] [delay=seconds]
    [length=number] [number={number|continuous}]
    [pattern=hexnum] [sappleaddress=network.node]
    [screenoutput={yes|no}] [timeout=number]
```

initiates the transmission of ping packets.

Any parameters not specified use the defaults configured with a previous invocation of the command:

```
set ping [[appleaddress=]network.node [delay=seconds]]
    [length=number] [number={number|continuous]
    [pattern=hexnum] [sappleaddress=network.node]
    [screenoutput={yes|no}] [timeout=number]
```

As each response packet is received, a message is displayed on the terminal and the details are recorded. The default configuration and summary information can be displayed with the command:

```
show ping
```

To halt a ping in progress, use the command:

```
stop ping
```

# Configuration Example

This example illustrates the basic steps required to configure an AppleTalk internetwork. In this example, three separate AppleTalk LANs are to be interconnected via a Point-to-Point Protocol link and a Frame Relay network (Figure 35-2 on page 35-18, Table 35-2 on page 35-19). The procedure is similar for all three routers, the main differences being the configuration of the different types of WAN link.

Some interface and port types mentioned in this example may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the *AR400 Series Router Hardware Reference*.

Figure 35-2: Example network configuration for AppleTalk routing

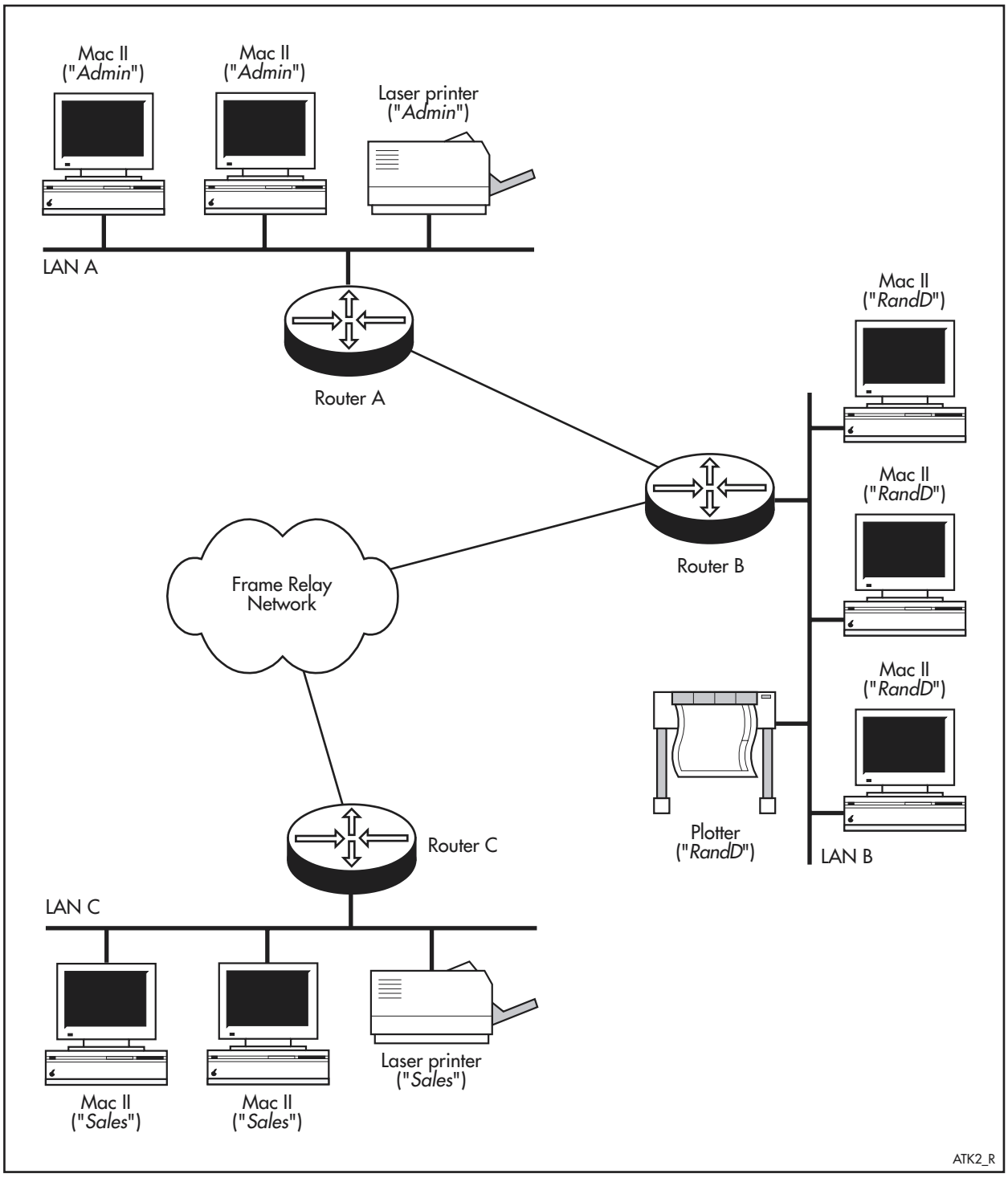


Table 35-2: Example configuration parameters for AppleTalk routing

| Parameter | Router A | Router B | Router C |
|---|---|---|---|
| Router Name | Router A | Router B | Router C |
| LAN Interface | eth0 | eth0 | eth0 |
| LAN Name | LAN A | LAN B | LAN C |
| LAN Network Number Range | 200–210 | 105–120 | 80–80 |
| LAN Zone Name | Admin | RandD | Sales |
| WAN Interfaces | ppp0 | ppp0, fr0 | fr0 |
| WAN Circuits | - | DLC 7 | DLC 3 |

### To configure Router A:

1. **Enable the AppleTalk module.**

   Enable the AppleTalk routing module, using the command:

   ```
   enable apple
   ```

2. **Create an AppleTalk port and an AppleTalk zone for the Ethernet interface.**

   Create an AppleTalk port on the local Ethernet interface with a network range of 200–210:

   ```
   add apple port int=eth0 seed=200-210
   ```

   Create an AppleTalk zone called "Admin" for the Ethernet network attached to the port, and make the zone the default zone for the network:

   ```
   add apple zone=admin port=1 default
   ```

3. **Create a PPP interface and an AppleTalk port to use the interface.**

   Create PPP interface 0 over synchronous interface 0:

   ```
   create ppp=0 over=syn0
   ```

   Create an AppleTalk port to use PPP interface 0:

   ```
   add apple port int=ppp0
   ```

4. **Check the configuration:**

   Check the AppleTalk configuration, using the commands:

   ```
   show apple port
   show apple zone
   show apple route
   ```

### To configure Router B:

1. **Enable the AppleTalk module.**

   Enable the AppleTalk routing module, with the command:

   ```
   enable apple
   ```

**2. Create an AppleTalk port and an AppleTalk zone for the Ethernet interface.**

Create an AppleTalk port on the local Ethernet interface with a network range of 105–120:

```
add apple port=eth0 seed=105-120
```

Create an AppleTalk zone called "RandD" for the Ethernet network attached to the port, and make the zone the default zone for the network:

```
add apple zone=randd port=1 default
```

**3. Create a PPP interface and an AppleTalk port to use the interface.**

Create PPP interface 0 over synchronous interface 0:

```
create ppp=0 over=syn0
```

Create an AppleTalk port to use PPP interface 0:

```
add apple port int=ppp0
```

**4. Create a Frame Relay interface and an AppleTalk port to use the interface.**

Create Frame Relay interface 0 over synchronous interface 1. Disable the LMI dialogue and reset the Frame Relay interface. Configure a static DLC with a DLCI of 7:

```
create fr=0 over=syn1
set fr=0 lmi=none
reset fr=0
add fr=0 dlc=7
```

Create an AppleTalk port to use Frame Relay interface 0, and add the static DLC:

```
add apple port int=fr0
add apple dlci=7 port=3
```

**5. Check the configuration:**

Check the AppleTalk configuration, using the commands:

```
show apple port
show apple zone
show apple dlci
show apple route
```

### To configure Router C:

**1. Enable the AppleTalk module.**

Enable the AppleTalk routing module, with the command:

```
enable apple
```

**2. Create an AppleTalk port and an AppleTalk zone for the Ethernet interface.**

Create an AppleTalk port on the local Ethernet interface with a network number of 80:

```
add apple port int=eth0 seed=80
```

Create an AppleTalk zone called "Sales" for the Ethernet network attached to the port, and make the zone the default zone for the network:

```
add apple zone=sales port=1 default
```

**3. Create a Frame Relay interface and an AppleTalk port to use the interface.**

Create Frame Relay interface 0 over synchronous interface 0. Disable the LMI dialogue and reset the Frame Relay interface. Configure a static DLC with a DLCI of 3:

```
create fr=0 over=syn0
set fr=0 lmi=none
reset fr=0
add fr=0 dlc=3
```

Create an AppleTalk port to use Frame Relay interface 0, and add the static DLC:

```
add apple port int=fr0
add apple dlci=3 port=2
```

**4. Check the configuration:**

Check the AppleTalk configuration, using the commands:

```
show apple port
show apple zone
show apple dlci
show apple route
```

# Command Reference

This section describes the commands available on the router to enable, configure, control and monitor the AppleTalk routing module.

Some interface and port types mentioned in this chapter may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the *AR400 Series Router Hardware Reference*.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page lxv of About this Software Reference in the front of this manual for details of the conventions used to describe command syntax. See Appendix A, Messages for a complete list of messages and their meanings.

# add apple circuit

**Syntax** `ADD APPle CIRCuit=`*`circuit`* `POrt=`*`port`*

where:

- *circuit* is a MIOX circuit name 1 to 15 characters long.
- *port* is an AppleTalk port number from 1 to 127.

**Description** This command adds a MIOX circuit to an AppleTalk port over an X25T interface. Up to 20 MIOX circuits may be attached to an AppleTalk port over an X25T interface.

The **circuit** parameter specifies the name of a MIOX circuit to be attached to the AppleTalk port. This circuit must already be configured for the X25T interface where the AppleTalk port is attached.

The **port** parameter specifies the AppleTalk port where the circuit is to be added. The AppleTalk port must already exist.

**Examples** To add the circuit "Head Office" to AppleTalk port 2, (assuming port 2 is an AppleTalk port with an X25T interface), use the command:

```
add app circ="head office" po=2
```

**Related Commands**

**add apple port**
**delete apple circuit**
**show apple circuit**
**show apple port**

# add apple dlci

**Syntax** `ADD APPle DLCi=`*`dlci`* `POrt=`*`port`*

where:

- *dlci* is the DLCI of a Frame Relay DLC from 0 to 1023.
- *port* is an AppleTalk port number from 1 to 127.

**Description** This command adds a Frame Relay DLCI to an AppleTalk port connected to a Frame Relay interface. When AppleTalk sends a broadcast to the port, then each DLCI receives a copy of the packet to be sent to the remote end.

The DLCI parameter specifies the Frame Relay DLCI to be added to the AppleTalk port. The DLCI must already exist or be configured on the Frame Relay interface where the AppleTalk port is attached.

The PORT parameter specifies the AppleTalk port where the DLCI is to be added. The AppleTalk port must already exist.

**Examples** To add DLCI 23 to an AppleTalk port 1 configured to Frame Relay interface 1, use the command:

```
add app dlc=23 po=1
```

**Related Commands**

**add apple port**
**delete apple dlci**
**show apple dlci**

# add apple packetfilter

**Syntax**

```
ADD APPle PACKETFilter=filter-id [ENTry=entry]
   [SNEtwork={network-range|ANY}]
   [DNEtwork={network-range|ANY}] [SNOde={node-range|ANY}]
   [DNOde={node-range|ANY}] [SSOcket={socket-range|ANY}]
   [DSOcket={socket-range|ANY}] [DIrection={IN|OUT}]
   [LOG={YES|NO|ON|OFF|TRUE|FALSE}]  ACtion={DENY|ALLOw}
```

where:

- *filter-id* is a decimal number range 0 to 99.
- *entry* is a decimal number range 1 to 65535.
- *network-range* is a network number range of the form of nnnnnnn-nnnnnnn. A network-range specified as “aaa” is treated as “aaa-aaa”. Network numbers must be decimal numbers from 0 to 65279.
- *node-range* is a node number range of the form of “nnnnnnn-nnnnnnn. A node-range specified as “aaa” is treated as “aaa-aaa”. Node numbers must be decimal numbers from 0 to 255.
- *socket-range* is a socket number range of the form of “nnnnnnn-nnnnnnn. A socket-range specified as “aaa” is treated as “aaa-aaa”. Socket numbers must be decimal numbers from 0 to 255.

**Description**

This command adds an AppleTalk DDP or packet filter to the router.

The **packetfilter** parameter specifies the filter number to be added.

The **entry** parameter specifies the entry number in the filter assigned to the new filter entry. Filter entries with a higher filter entry number, and an existing filter entry with the same entry number, are pushed down the filter. The default is to add a new filter entry to the end of the filter.

The **snetwork** parameter specifies the source network address of the packet. The default is **any**.

The **dnetwork** parameter specifies the destination network address of the packet. The default is **any**.

The **snode** parameter specifies the source node address of the packet. The default is **any**.

The **dnode** parameter specifies the destination node address of the packet. The default is **any**.

The **ssocket** parameter specifies the source socket number of the packet. The default is **any**.

The **dsocket** parameter specifies the destination socket number of the packet. The default is **any**.

The **direction** parameter specifies the direction of the packet to be filtered. If direction is **in**, then this entry is matched against an incoming packet on the specified interface. If direction is **out**, then this entry is matched against an outgoing packet on the specified interface. The default is **in**.

The **log** parameter specifies whether packets that match the filter entry are logged. The default is **off**.

The **action** parameter specifies what action is performed on any packets that match the filter entry. If **deny** is specified, matching packets are discarded immediately. If **allow** is specified, matching packets received or sent by the associated interface.

## Examples

To add an AppleTalk filter with ID equal 1 that denies all packets from all nodes with network number 120 directed at socket 56, use the command:

```
add app packetf=1 sne=120 dso=56 ac=deny
```

## Related Commands

**add apple port**
**add apple routefilter**
**add apple zonefilter**
**delete apple packetfilter**
**set apple packetfilter**
**set apple port**
**show apple packetfilter**
**show apple port**

# add apple port

**Syntax**

```
ADD APPle POrt INTerface=interface [DEMand={ON|OFF|YES|
   NO}] [HInt=network:node] [SEED=seed]
   [PACKETFilter={filter-id|NONE}]
   [ROUTEFilter={filter-id|NONE}] [ZONEFilter={filter-id|
   NONE}]
```

where:

- *interface* is a valid interface name formed by concatenating an interface type and an interface instance.
- *network* is an AppleTalk network number from 0 to 65279.
- *node* is decimal number from 0 to 253.
- *seed* is an AppleTalk network number range of the form of "nnnnnnn-nnnnnnn". A seed specified as "aaa" is treated as "aaa-aaa".
- *filter-id* is a decimal number range 0 to 99.

**Description**

This command adds a port to the AppleTalk routing module. A maximum of 127 AppleTalk ports can be configured at any one time.

The **interface** parameter specifies the lower layer interface to be used by the AppleTalk port. The interface must not already be used by another AppleTalk port. Valid interfaces are:

- eth (such as eth0)
- VLAN (such as vlan1)
- FR (such as fr0)
- X.25 DTE (such as x25t0)
- PPP (such as ppp0)

The interface must already exist. To see a list of all currently available interfaces, use the command **show interface**.

For Frame Relay and X25T interfaces, DLCIs and MIOX circuits associated with the interface must also be added, using the **add apple circuit** command on page 35-22 and **add apple dlci** command on page 35-23, before the interface can be used by the AppleTalk routing module.

The **port** parameter specifies the AppleTalk port to be added. The AppleTalk port must not already exist.

The **demand** parameter specifies whether the port is connected to a dial-on-demand PPP link that automatically disconnects when there has been no traffic for a period set by the **idle** parameter of the associated PPP interface. AppleTalk routing broadcasts are suppressed on AppleTalk ports with the **demand** parameter set to **on**. The default is **off**.

The **hint** parameter specifies the network and node to be used for the port when the network number falls within the cable range and the node number has not been used by any other device in the network. This parameter is valid for ETH and PPP interfaces and is required for AppleTalk ports using numbered PPP interfaces. By default, AppleTalk uses addressless PPP interfaces.

The **seed** parameter identifies the router as a seed router for the AppleTalk network with the specified network number. The **seed** parameter is valid for Ethernet interfaces.

The **seed** and **hint** parameters can be used in combination to influence the way AppleTalk assigns AppleTalk node numbers to devices on attached AppleTalk networks (Table 35-3 on page 35-27).

Table 35-3: Interaction of SEED and HINT parameters in setting AppleTalk node numbers

| SEED | HINT | Condition | Result |
|---|---|---|---|
| SEED=x-y | HINT=0:z | x <= w <= y | w:z |
| SEED=x-y | HINT=w:z | x <= w <= y | w:z |
| SEED=x-y | HINT=w:0 | x <= w <= y | w:nnn |
| Not specified | HINT=0:z | | mmm:z |

If **hint** is specified with a value of 0 for the network number (e.g. 0:8), then AppleTalk attempts to select a network number (w) in the range specified by **seed** with a free matching node number (8). If **hint** is specified with a non-zero value for both the network and node number (e.g. 2:8), then AppleTalk attempts to use the specified network and node number (2:8). If **hint** is specified with a value of 0 for the node number (e.g. 2:0), then AppleTalk uses the specified network number (provided it is in the range specified by **seed**) and select with a free node number (nnn) in that network. If **seed** is not specified, **hint** does not accept non-zero network numbers. In this case, AppleTalk selects a network number (mmm) from the network numbers already configured on the AppleTalk network. In all cases, when the required node number is in use, the next free node number in the same network is used. When all node numbers in the network are used, AppleTalk tries other network numbers in the range specified by **seed**.

The **packetfilter** parameter specifies the packet filter ID to be used on this interface. All incoming and outgoing packets through this interface are subject to filtering with the specified packet filter. If **none** is specified, then no packet filtering is done on this port.

The **routefilter** parameter specifies the route advertisement filter ID to be used on this interface. All incoming and outgoing route advertisement through this interface are subject to filtering with the specified route filter. If **none** is specified, then no route advertisement filtering is done on this port.

The **zonefilter** parameter specifies the zone advertisement filter ID to be used on this interface. All outgoing zone advertisement through this interface are subject to filtering with the specified route filter. If **none** is specified, then no zone advertisement filtering is done on this port.

## Examples

To add an AppleTalk port that uses Ethernet, use the command:

```
add app po int=eth0
```

## Related Commands

**add apple dlci**
**add apple circuit**
**add apple packetfilter**
**add apple routefilter**
**add apple zonefilter**
**delete apple port**
**set apple port**
**show apple port**

# add apple route

**Syntax**

```
ADD APPle ROUte=network POrt=port [{NEXThop=network:node|
   CIRCuit=circuit|DLCI=dlci}] [HOPs=hopcount]
```

where:

- *network* is an AppleTalk network number or an AppleTalk network number range of the form "nnnnnnn-nnnnnnn". Network numbers must decimal numbers from 0 to 65279.
- *port* is an AppleTalk port number from 1 to 127.
- *circuit* is the number of a MIOX circuit attached to an AppleTalk port over an X25T interface.
- *dlci* is the DLCI of a Frame Relay DLC attached to an AppleTalk port over a Frame Relay interface, from 0 to 1023.
- *node* is decimal number from 0 to 127.
- *hopcount* is decimal number from of 1 to 16.

**Description**

This command adds a static route to the AppleTalk routing table. The ROUTE parameter specifies the AppleTalk network number or network range for the route to be added. The route must not already exist.

The **port** parameter specifies the AppleTalk port number with which this route is associated. The port must already exist.

The **nexthop** parameter specifies the next hop router where packets are to be forwarded. The value must be the network number and node ID of an AppleTalk device. A route must exist to the network. This parameter is valid for routes associated with ports with Ethernet and PPP interfaces.

The **circuit** parameter specifies the number of a MIOX circuit associated with the X25T interface where the AppleTalk port is attached. This parameter is valid for AppleTalk ports attached to X25T interfaces.

The **dlci** parameter specifies the DLCI (Data Link Connection Identifier) of a DLC (Data Link Connection) associated with the Frame Relay interface where the AppleTalk port is attached. This parameter is valid for AppleTalk ports attached to Frame Relay interfaces.

There is little advantage in adding a static route to an AppleTalk port that is not attached to a PPP interface, since the **demand** option is currently supported on PPP interfaces only.

The **hops** parameter specifies this distance (number of hops) to the destination route. The default is 1.

**Examples**

To add a static route on port 2 with a network range of 200-210, and a distance of 2, use the command:

```
add app rou=200-210 po=2 hop=2
```

**Related Commands**

**delete apple route**
**show apple port**
**show apple route**

# add apple routefilter

**Syntax**

```
ADD APPle ROUTEfilter=filter-id [ENTry=entry]
   NETWork=network-range [DIrection={IN|OUT}] [LOG={YES|
   NO|ON|OFF}] ACtion={DENY|ALLOW}
```

where:

- *filter-id* is a decimal number from 0 to 99.
- *entry* is a decimal number from 1 to 65535.
- *network-range* is an AppleTalk network number range of the form of "nnnnnnn-nnnnnnn". A network specified as "aaa" is treated as "aaa-aaa". Network numbers must be decimal numbers from 0 to 65279.

**Description**

This command adds an AppleTalk RTMP or route filter to the router.

The **routefilter** parameter specifies the route filter number to be added.

The **entry** parameter specifies the entry number in the filter assigned to the new filter entry. Filter entries with a higher filter entry number, and an existing filter entry with the same entry number, are pushed down the filter. The default is to add a new filter entry to the end of the filter.

The **network** parameter specifies the network address range to be filtered.

The **direction** parameter specifies the direction of the route to be filtered. If direction is **in**, then this entry is matched against an incoming route advertised on the specified interface. If direction is **out**, then this entry is matched against an outgoing route advertised on the specified interface. The default is **in**.

The **log** parameter specifies whether a route that matches this entry is logged. The default is **off**.

The **action** parameter specifies what action is performed on any route matches the filter entry. If **deny** is specified, the matching route is discarded immediately. If **allow** is specified, the matching route is advertised or received by the associated interface.

**Examples**

To add an AppleTalk route filter with ID 4 a denies all outgoing routes for the network address range of 500-1000, use the command:

```
add app route=4 netw=500-1000 di=out ac=deny
```

**Related Commands**

**add apple packetfilter**
**add apple port**
**add apple zonefilter**
**delete apple routefilter**
**set apple port**
**set apple routefilter**
**show apple port**
**add apple routefilter**

# add apple zone

**Syntax** `ADD APPle ZOne=`*`zone-name`* `POrt=`*`port`* `[DEFault]`

where:

- *zone-name* is a character string 1 to 32 characters long. Valid characters are uppercase and lowercase letters, and decimal digits (0-9).
- *port* is an AppleTalk port number from 1 to 127.

**Description** This command is adds an AppleTalk zone to the zone list for a port. No more than 255 zones can be added to a particular network. If the router is a seed router for the network attached to the port, the port must have a default zone, specified by the **default** parameter.

The **default** parameter identifies the specified zone as the default zone for the network where the port is connected.

The **port** parameter specifies an existing AppleTalk port where the zone name is to be added.

The **zone** parameter specifies the zone name to be added to the zone list for the port. Seed routers must have zones defined. Ports connected to AppleTalk networks (such as Ethernet interfaces) support zone names.

**Examples** To add the zone name “Godzilla” to port 1 as a default zone for the attached AppleTalk network, use the command:

```
add app zo=godzilla po=1 def
```

**Related Commands**

**add apple port**
**delete apple zone**
**set apple zone**
**show apple zone**

# add apple zonefilter

**Syntax**

```
ADD APPle ZONEFilter=filter-id [ENTry=entry]
   ZOne=zone-name [LOG={YES|NO|ON|OFF}] ACtion={DENY|
   ALLOw}
```

where:

- *filter-id* is a decimal number range 0 to 99.
- *entry* is a decimal number range 1 to 65535.
- *zone-name* is a character string 1 to 32 characters long. Valid characters are any printable characters. If *zone-name* contains spaces, it must be in double quotes.

**Description**

This command adds an AppleTalk ZIP or zone filter to the router.

The **zonefilter** parameter specifies the zone filter number to be added.

The **entry** parameter specifies the entry number in the filter assigned to the new filter entry. Filter entries with a higher filter entry number, and an existing filter entry with the same entry number, are pushed down the filter. The default is to add a new filter entry to the end of the filter.

The **zone** parameter specifies the zone name to be filtered.

The **log** parameter specifies whether zones that match the filter entry are logged. The default is **off**.

The **action** parameter specifies what action is performed on any outgoing zone that matches the filter entry. If **deny** is specified, the matching zone is discarded immediately. If **allow** is specified, the matching zone is advertised by the associated interface.

**Examples**

To add an AppleTalk zone filter with ID 2 that denies all outgoing zones for the "Dungeon" zone with logging on, use the command:

```
add app zonef=2 zo="dungeon" log=yes ac=deny
```

**Related Commands**

**add apple packetfilter**
**add apple port**
**add apple routefilter**
**delete apple zonefilter**
**set apple port**
**set apple zonefilter**
**show apple port**
**show apple zonefilter**

# delete apple circuit

**Syntax** `DELete APPle CIRCuit=`*`circuit`* `POrt=`*`port`*

where:

- *circuit* is a MIOX circuit name 1 to 15 characters long.
- *port* is an AppleTalk port number from 1 to 127.

**Description** This command deletes a MIOX circuit from an AppleTalk port. The port must be attached to an X25T interface and the specified MIOX circuit must already exist on the X25T interface.

The **circuit** parameter specifies the name of the MIOX circuit to be deleted from the AppleTalk port. This circuit must already be configured for the X25T interface where the AppleTalk port is attached.

The **port** parameter specifies the AppleTalk port where the circuit is to be deleted. The AppleTalk port must already exist.

**Examples** To delete circuit "Head Office" from port 1, use the command:

```
del circ="head office" po=1
```

**Related Commands**
**add apple circuit**
**add apple port**
**show apple circuit**

# delete apple dlci

**Syntax** `DELete APPle DLCi=`*`dlci`* `POrt=`*`port`*

where:

- *dlci* is the DLCI of a Frame Relay DLC from 0 to 1023.
- *port* is an AppleTalk port number from 1 to 127.

**Description** This command deletes a Frame Relay DLCI from an AppleTalk port. The port must be attached to a Frame Relay interface and the specified DLCI must already exist on the Frame Relay interface.

The **dlci** parameter specifies the DLCI to be deleted from the AppleTalk port. This DLCI must already be configured for the Frame Relay interface where the AppleTalk port is attached.

The **port** parameter specifies the AppleTalk port where the DLCI is to be deleted. The AppleTalk port must already exist.

**Examples** To delete DLCI 4 from the port 2, use the command:

```
del app dlc=4 po=2
```

**Related Commands**

**add apple dlci**
**add apple port**
**show apple dlci**

# delete apple packetfilter

**Syntax** `DELete APPle PACKETFilter=`*`filter-id`* `[ENTry=`*`entry`*`]`

where:

- *filter-id* is a decimal number range 0 to 99.
- *entry* is a decimal number range 1 to 65535.

**Description** This command deletes the specified DDP or packet filter entry. If the **entry** parameter is not specified all entries belonging to the filter are deleted, therefore removing the packet filter.

The **packetfilter** parameter specifies the DDP filter to delete filter entries from.

The **entry** parameter specifies the entry number in the filter to be deleted.

**Examples** To delete an AppleTalk filter entry 20 with filter ID 1, use the command:

```
del app packetf=1 ent=20
```

**Related Commands**
**add apple packetfilter**
**add apple port**
**set apple packetfilter**
**set apple port**
**show apple packetfilter**
**show apple port**

# delete apple port

**Syntax** `DELete APPle POrt=`*`port`*

where *port* is an AppleTalk port number from 1 to 127

**Description** This command deletes a port from the AppleTalk routing module. A maximum of 127 AppleTalk ports can be configured at any one time.

The **port** parameter specifies the AppleTalk port to be deleted. The AppleTalk port must already exist.

**Examples** To delete AppleTalk port 2, use the command:

```
del app po=2
```

**Related Commands**
**add apple port**
**set apple port**
**show apple port**

# delete apple route

**Syntax** `DELete APPle ROUte=`*`network`*

where *network* is an AppleTalk network number or an AppleTalk network number in the "nnnnnnn-nnnnnnn" format. Network numbers must decimal numbers from 0 to 65279.

**Description** This command deletes an AppleTalk static route. The **route** parameter specifies the AppleTalk network number or network range for the route to be deleted. The route must already exist.

**Examples** To delete the static route to the network range 200-210, use the command:

```
del app rou=200-210
```

**Related Commands**
**add apple route**
**show apple route**

# delete apple routefilter

**Syntax** `DELete APPle ROUTEfilter=`*`filter-id`* `[ENTry=`*`entry`*`]`

where:

- *filter-id* is a decimal number range 0 to 99.
- *entry* is a decimal number range 1 to 65535.

**Description** This command deletes the specified RTMP or route filter entry. If the **entry** parameter is not specified all entries belonging to the filter are deleted, therefore removing the packet filter.

The **routefilter** parameter specifies the RTMP filter to delete entries from.

The **entry** parameter specifies the entry number in the filter to be deleted.

**Examples** To delete an AppleTalk route filter entry 20 with filter ID equal to 1, use the command:

```
del app routef=1 ent=20
```

**Related Commands**
**add apple port**
**add apple routefilter**
**set apple port**
**set apple routefilter**
**show apple port**
**show apple routefilter**

# delete apple zone

**Syntax** `DELete APPle ZOne=`*`zone-name`* `POrt=`*`port`*

where:

- *zone-name* is a character string 1 to 32 characters long. Valid characters are uppercase and lowercase letters, and decimal digits (0-9).
- *port* is an AppleTalk port number from 1 to 127.

**Description** This command is deletes an AppleTalk zone from the zone list for a port.

The **port** parameter specifies the AppleTalk port from which the zone name is to be deleted. The AppleTalk port must already exist.

The **zone** parameter specifies the zone name to be deleted from the zone list for the port. Ports connected to AppleTalk networks (i.e. Ethernet interfaces) support zone names.

**Examples** To delete the zone “Godzilla” from port 1, use the command:

```
del app zo=godzilla po=1
```

**Related Commands**

**add apple port**
**add apple zone**
**show apple zone**

# delete apple zonefilter

**Syntax** `DELete APPle ZONEFilter=`*`filter-id`* `[ENTry=`*`entry`*`]`

where:

- *filter-id* is a decimal number range 0 to 99.
- *entry* is a decimal number range 1 to 65535.

**Description** This command deletes the specified ZIP or zone filter entry. If the ENTRY parameter is not specified all entries belonging to the filter are deleted, therefore removing the packet filter.

The **zonefilter** parameter specifies the ZIP filter to delete entries from.

The **entry** parameter specifies the entry number in the filter to be deleted.

**Examples** To delete an AppleTalk zone filter entry 20 with filter ID 1, use the command:

```
del app zonef=1 ent=20
```

**Related Commands**
**add apple port**
**add apple zonefilter**
**set apple port**
**set apple zonefilter**
**show apple port**
**show apple zonefilter**

# disable apple

**Syntax** `DISable APPle`

**Description** This command disables AppleTalk routing in the router. This reinitialises all data structures used by the AppleTalk routing module.

**Examples** To disable AppleTalk routing, use the command:

```
dis app
```

**Related Commands** **enable apple**

# disable apple debug

**Syntax** `DISable APPle DEBug={ALL|AARP|PACket|PKT|ROUte|ZIp|ZOne}`

**Description** This command disables AppleTalk debugging on the specified sub-protocol.

The **debug** parameter specifies the AppleTalk protocol layer for which debugging is to be disabled. The options **packet** and **pkt** are synonyms for **ddp**.

**Examples** To disable DDP level debugging, use the command:

```
dis app deb=pac
```

**Related Commands** enable apple debug

# enable apple

**Syntax** `ENAble APPle`

**Description** This command enables AppleTalk routing on the router.

**Examples** To enable AppleTalk routing, use the command:

```
ena app
```

**Related Commands** disable apple

# enable apple debug

**Syntax** `ENAble APPle DEBug={ALL|AARP|PACket|PKT|ROUte|ZIp|ZOne}`

**Description** This command enables AppleTalk debugging on the specified sub-protocol.

The **debug** parameter specifies the AppleTalk protocol layer to debug. The options **packet** and **pkt** are synonyms for **ddp**.

**Examples** To enable DDP level debugging, use the command:

```
ena app deb=pac
```

**Related Commands** disable apple debug

# purge apple

**Syntax** `PURge APPle`

**Description** This command purges all AppleTalk configuration information, but does not change the current status of the AppleTalk routing module. If the AppleTalk routing module is currently enabled, it remains enabled after the **purge apple** command has been executed. Similarly, if the AppleTalk routing module is currently disabled, it remains disabled after the **purge apple** command has been executed.

**Examples** To purge the configuration of the AppleTalk routing module, use the command:

```
pur app
```

**Related Commands** **disable apple**
**enable apple**

# reset apple

**Syntax** `RESET APPle`

**Description** This command resets the AppleTalk routing module to its initial state, and is equivalent to disabling and then re-enabling the AppleTalk routing module.

The current configuration is retained but all dynamically learned information such as zones, routes and ARP entries are flushed.

**Examples** To reset the AppleTalk routing module, use the command:

```
reset app
```

**Related Commands** **disable apple**
**enable apple**

# set apple packetfilter

**Syntax**

```
SET APPle PACKETFilter=filter-id [ENTry=entry]
   [SNEtwork={network-range|ANY}]
   [DNEtwork={network-range|ANY}] [SNOde={node-range|ANY}]
   [DNOde={node-range|ANY}] [SSOcket={socket-range|ANY}]
   [DSOcket={socket-range|ANY}] [DIrection={IN|OUT}]
   [LOG={YES|NO|ON|OFF|TRUE|FALSE}]  ACtion={DENY|ALLOw}
```

where:

- *filter-id* is a decimal number range 0 to 99.
- *entry* is a decimal number range 1 to 65535.
- *network-range* is a network number in the "nnnnnnn-nnnnnnn" format. A network-range specified as "aaa" is treated as "aaa-aaa". Network numbers must be decimal numbers from 0 to 65279.
- *node-range* is a node number in the "nnnnnnn-nnnnnnn" format. A node range specified as "aaa" is treated as "aaa-aaa". Node numbers must be decimal numbers from 0 to 255.
- *socket-range* is a socket number in the "nnnnnnn-nnnnnnn" format. A socket range specified as "aaa" is treated as "aaa-aaa". Socket numbers must be decimal numbers from 0 to 255.

**Description**

This command changes the specified entry in a DDP or packet filter. Specified parameters cause corresponding fields in the entry to be modified.

The **packetfilter** parameter specifies the packet filter to modify.

The **entry** parameter specifies the entry number in the filter assigned to the new filter entry. The specified entry must exist.

The **snetwork** parameter specifies the source network address of the packet.

The **dnetwork** parameter specifies the destination network address of the packet.

The **snode** parameter specifies the source node address of the packet.

The **dnode** parameter specifies the destination node address of the packet.

The **ssocket** parameter specifies the source socket number of the packet.

The **dsocket** parameter specifies the destination socket number of the packet.

The **direction** parameter specifies the direction of the packet to be filtered. When direction is **in**, then this entry is matched against an incoming packet on the specified interface. When direction is **out**, then this entry is matched against an outgoing packet on the specified interface.

The **log** parameter specifies whether packets that match the filter entry are logged. The default is **off**.

The **action** parameter specifies what action is performed on any packets that match the filter entry. If **deny** is specified, matching packets are discarded immediately. If **allow** is specified, matching packets are sent or received by the associated interface.

**Examples** To modify an AppleTalk filter 1, entry number 10 with action **deny** to action **allow**, use the command:

```
set app packetf=1 sne=10 ac=allo
```

**Related Commands**

**add apple packetfilter**
**add apple port**
**delete apple packetfilter**
**set apple port**
**show apple packetfilter**
**show apple port**

# set apple port

**Syntax**

```
SET APPle POrt [DEMand={ON|OFF|YES|NO}]
   [HInt=network:node] [SEED=seed]
   [PACKETFilter={filter-id|NONE}]
   [ROUTEFilter={filter-id|NONE}] [ZONEFilter={filter-id|
   NONE}]
```

where:

- *port* is an AppleTalk port number from 1 to 127.
- *network* is an AppleTalk network number from 0 to 65279.
- *node* is decimal number from 0 to 253.
- *seed* is an AppleTalk network number range in the "nnnnnnn-nnnnnnn" format. A seed specified as "aaa" is treated as "aaa-aaa".
- *filter-id* is a decimal number from 0 to 99.

**Description**

This command modifies the configuration of an AppleTalk port.

The **reset apple** command on page 35-39 should be executed after a **set apple port** command to ensure that changes take effect immediately.

The **port** parameter specifies the AppleTalk port to be modified. The AppleTalk port must already exist.

The **demand** parameter specifies whether the port is connected to a dial-on-demand PPP link that automatically disconnects when there has been no traffic for a period set by the **idle** parameter of the associated PPP interface. AppleTalk routing broadcasts are suppressed on AppleTalk ports with the **demand** parameter set to **on**. The default is **off**.

The **hint** parameter specifies the network and node to be used for the port if the network number falls within the cable range and the node number has not been used by any other device in the network. This parameter is valid for ETH and PPP interfaces and is required for AppleTalk ports using numbered PPP interfaces. By default, AppleTalk uses addressless PPP interfaces.

The **seed** parameter identifies the router as a seed router for the AppleTalk network with the specified network number. The SEED parameter is valid for Ethernet interfaces.

The **seed** and **hint** parameters can be used in combination to influence the way AppleTalk assigns AppleTalk node numbers to devices on attached AppleTalk networks (Table 35-3 on page 35-27). If **hint** is specified with a value of 0 for the network number (e.g. 0:8), then AppleTalk attempts to select a network number (w) in the range specified by **seed** with a free matching node number (8). If **hint** is specified with a non-zero value for both the network and node number (e.g. 2:8) then AppleTalk attempts to use the specified network and node number (2:8). If **hint** is specified with a value of 0 for the node number (e.g. 2:0), then AppleTalk uses the specified network number (provided it is in the range specified by **seed**) and select with a free node number (nnn) in that network. If **seed** is not specified, **hint** does not accept non-zero network numbers. In this case AppleTalk selects a network number (mmm) from the network numbers already configured on the AppleTalk network. In all cases, when the required node number is in use, the next free node number in the same network is used. When all node numbers in the network are used, AppleTalk tries other network numbers in the range specified by **seed**.

The **packetfilter** parameter specifies the packet filter ID to be used on this interface. All incoming and outgoing packets through this interface are subject to filtering by the specified packet filter. If **none** is specified, then no packet filtering is done on this port. An existing filter can be removed by setting **packetfilter** to **none**.

The **routefilter** parameter specifies the route advertisement filter ID to be used on this interface. All incoming and outgoing route advertisement through this interface are subject to filtering by the specified route filter. If **none** is specified, then no route advertisement filtering is done on this port. An existing filter can be removed by setting **routefilter** to **none**.

The **zonefilter** parameter specifies the zone advertisement filter ID to be used on this interface. All outgoing zone advertisement through this interface are subject to filtering by the specified route filter. If **none** is specified, then no zone advertisement filtering is done on this port. An existing filter can be removed by setting **zonefilter** to **none**.

## Examples

To set the seed for port 2 to 123-125, and assign the port with route filter number 5, use the command:

```
set app po=2 seed=123-125 routef=5
```

## Related Commands

**add apple port**
**add apple packetfilter**
**add apple routefilter**
**add apple zonefilter**
**delete apple port**
**show apple port**

# set apple routeconvert

**Syntax** `SET APPle ROUTEConvert={ON|OFF|YES|NO}`

**Description** This command enables or disables the conversion of non-extended (LocalTalk) network routes to extended network routes when RTMP packets are generated. Route conversion is necessary on any internet with a LocalTalk network that must communicate with other AppleTalk networks on the same internet. Route conversion is enabled by default.

A *nonextended* network (e.g. a LocalTalk network) is assigned exactly one network number and can have a maximum of 254 active nodes, each uniquely identified by each node's 8-bit AppleTalk node ID. An *extended* network may have multiple network numbers and up to approximately 16 million nodes, each uniquely identified by network number/node ID pairs. An extended network can be thought of as a number of nonextended networks all residing on the same physical data link (e.g. a LAN with multiple network numbers accessed via a WAN link).

**Examples** To enable the conversion of non-extended network routes to extended network routes, use the command:

```
set app routec=on
```

**Related Commands** **show apple**

# set apple routefilter

**Syntax**

```
SET APPLE ROUTEFILTER=filter-id ENTry=entry
   [NETWork=network-range] [DIrection={IN|OUT}] [LOG={YES|
   NO|ON|OFF}] [ACtion={DENY|ALLOw}]
```

where:

- *filter-id* is a decimal number from 0 to 99.
- *entry* is a decimal number from 1 to 65535.
- *network-range* is an AppleTalk network number range in the "nnnnnnn-nnnnnnn" format. A network-range specified as "aaa" is treated as "aaa-aaa". Network numbers must be decimal numbers from 0 to 65279.

**Description**

This command changes the specified entry in a RTMP or route filter. Specified parameters cause the corresponding fields in the entry to be modified.

The **routefilter** parameter specifies the RTMP filter to modify.

The **entry** parameter specifies the entry number in the filter assigned to the new filter entry. The specified entry must exist.

The **network** parameter specifies the network address range to filter.

The **direction** parameter specifies the direction of the route to be filtered. When direction is **in**, then this entry is matched against an incoming route advertised on the specified interface. When direction is **out**, then this entry is matched against an outgoing route advertised on the specified interface.

The **log** parameter specifies whether packets that match the filter entry are logged.

The **action** parameter specifies what action is performed on any route that matches the filter entry. If **deny** is specified, the matching route is discarded immediately. If **allow** is specified, the matching route is advertised or received by the associated interface.

**Examples**

To modify an AppleTalk filter 2, entry number 5 to allow incoming advertisement of network numbers ranging from 100 to 200, use the command:

```
set app routef=2 ent=5 netw=10 di=in ac=allow
```

**Related Commands**

**add apple port**
**add apple routefilter**
**delete apple routefilter**
**set apple port**
**show apple port**
**show apple routefilter**

# set apple zone

**Syntax** `SET APPle ZOne=`*`zone-name`* `POrt=`*`port`* `[DEFault]`

where:

- *zone-name* is a character string 1 to 32 characters long. Valid characters are uppercase and lowercase letters, and decimal digits (0-9).
- *port* is an AppleTalk port number from 1 to 127.

**Description** This command sets the default zone for the AppleTalk network where the port is connected. If the router is a seed router for the network attached to the port, the port must have a default zone, specified by the **default** parameter.

The **default** parameter identifies the specified zone name as the default zone for the network where the port is to be connected.

The **port** parameter specifies the AppleTalk port where the zone is to be the default zone. The AppleTalk port must already exist.

The **zone** parameter specifies the name of the zone to be the default zone for the AppleTalk network attached to the port. Ports connected to AppleTalk networks (i.e. Ethernet interfaces) support zone names.

**Examples** To set zone “Godzilla” on port 2 to be the default zone, use the command:

```
set app zo=godzilla po=2 def
```

**Related Commands**
**add apple zone**
**delete apple zone**
**show apple zone**

# set apple zonefilter

**Syntax**

```
SET APPle ZONEFilter=filter-id ENTry=entry
   [ZOne=zone-name] [LOG={YES|NO|ON|OFF}] [ACtion={DENY|
   ALLOw}]
```

where:

- *filter-id* is a decimal number range 0 to 99.
- *entry* is a decimal number range 1 to 65535.
- *zone-name* is any printable character string 1 to 32 characters long. If *zone name* contains spaces, it must be in double quotes.

**Description**

This command changes the specified entry in a ZIP or zone filter. Specified parameters cause the corresponding fields in the entry to be modified.

The **zonefilter** parameter specifies the ZIP filter to modify. The specified entry must exist.

The **entry** parameter specifies the entry number in the filter assigned to the new filter entry. The specified entry must exist.

The **zone** parameter specifies the zone name to be filtered.

The **log** parameter specifies whether packets that match the filter entry are logged. The default is **off**.

The **action** parameter specifies what action is performed on any outgoing zone that matches the filter entry. If **deny** is specified, the matching zone is discarded immediately. If **allow** is specified, the matching zone is distributed by the associated interface.

**Examples**

To modify an AppleTalk filter 2, entry number 5 to allow outgoing advertisement of zone “Cave”, use the command:

```
set app zonef=2 ent=5 zo="cave" ac=allow
```

**Related Commands**

**add apple port**
**add apple zonefilter**
**delete apple zonefilter**
**set apple port**
**show apple port**
**show apple zonefilter**

# show apple

**Syntax** `SHow APPle`

**Description** This command displays the current status of the AppleTalk routing module (Figure 35-3, Table 35-4).

Figure 35-3: Example output from the **show apple** command

```
Appletalk Module Configuration
---------------------------------------
Module Status ................. Enabled
LocalTalk Route Conversion ..... On
---------------------------------------
```

Table 35-4: Parameters in output of the **show apple** command

| Parameter | Meaning |
|---|---|
| Module Status | Whether the AppleTalk routing module is enabled. |
| LocalTalk Route Conversion | Whether LocalTalk non-extended routes are converted to extended routes in RTMP packets. |

**Examples** To show the status of the AppleTalk routing module, use the command:

```
sh app
```

**Related Commands** **enable apple**
**disable apple**

# show apple aarp

**Syntax** `SHow APPle AARP`

**Description** This command displays the contents of the AARP cache (Figure 35-4, Table 35-5).

Figure 35-4: Example output from the **show apple aarp** command

```
Apple Address Resolution Table

  Network  Node  Physical
  -------------------------------
  22       124   00-00-c0-c9-c6-7b
  -------------------------------
```

Table 35-5: Parameters in output of the **show apple aarp** command

| Parameter | Meaning |
|---|---|
| Network | AppleTalk network number of the AppleTalk device. |
| Node | AppleTalk node number of the AppleTalk device. |
| Physical | Physical (MAC) address of the AppleTalk device on the network. |

**Examples** To display the current AARP cache, use the command:

```
sh app aarp
```

# show apple circuit

**Syntax** `SHow APPle CIRCuit`

**Description** This command displays all MIOX circuits used and available for use by the AppleTalk routing module (Figure 35-5, Table 35-6).

Figure 35-5: Example output from the **show apple circuit** command

```
AppleTalk Circuits
-------------------------------------------------
Port  Interface  Circuit Name     Number  State
-------------------------------------------------
   1  x25t0      ToB                   0  Active
```

Table 35-6: Parameters in output of the **show apple circuit** command

| Parameter | Meaning |
|---|---|
| Port | Number of the AppleTalk port. |
| Interface | X25T interface used by the AppleTalk port. |
| Circuit Name | Name of a MIOX circuit defined for the X25T interface. |
| Number | PVC or SVC circuit number of the MIOX circuit. |
| State | Whether the MIOX circuit is Active or Allowed. Active circuits can be used for AppleTalk routing. |

**Examples** To display the available MIOX circuits, use the command:

```
sh app circ
```

**Related Commands**

**add apple circuit**
**delete apple circuit**

# show apple counter

**Syntax**

```
SHow APPle COUnter={ALL|AARP|ATP|DDp|NBp|PACket|PKT|POrt|
   ROUte|RTMp|ZIp|ZOne}
```

**Description**

This command displays all AppleTalk counters or all counters from the specified group of counters.

If **aarp** is specified, counters for the AppleTalk Address Resolution Protocol (AARP) are displayed (Figure 35-6, Table 35-7).

If **atp** is specified, counters for the AppleTalk Transaction Protocol (ATP) are displayed (Figure 35-7 on page 35-52, Table 35-8 on page 35-52).

If **ddp**, **pkt** or **packet** is specified, counters for the AppleTalk Datagram Delivery Protocol (DDP) are displayed (Figure 35-8 on page 35-52, Table 35-9 on page 35-53).

If **nbp** is specified, counters for the AppleTalk Name Binding Protocol (NBP) are displayed (Figure 35-9 on page 35-54, Table 35-10 on page 35-54).

If **port** is specified, counters for traffic transmitted and received via the AppleTalk ports on the router are displayed (Figure 35-10 on page 35-54, Table 35-11 on page 35-54).

If **route** or **rtmp** is specified, counters for the AppleTalk Routing Table Maintenance Protocol (RTMP) are displayed (Figure 35-11 on page 35-55, Table 35-12 on page 35-55).

If **zip** or **zone** is specified, counters for the AppleTalk Zone Information Protocol (ZIP) are displayed (Figure 35-12 on page 35-56, Table 35-13 on page 35-56).

Figure 35-6: Example output from the **show apple counter=aarp** command

```
AARP Counter
-------------
  aarpLookups ..................... 241
  aarpHits ........................ 137
```

Table 35-7: Parameters in output of the **show apple count=aarp** command

| Parameter | Meaning |
|---|---|
| aarpLookups | Number of times the AARP cache for this entity was searched. |
| aarpHits | Number of times an entry was searched for and found in the AARP cache for this entity. |

Figure 35-7: Example output from the **show apple counter=atp** command

```
ATP Counter
------------

  atpInPkts ......................... 1
  atpOutPkts ........................ 1
  atpTRequestRetransmissions ......... 0
  atpTResponseRetransmissions ........ 0
  atpReleaseTimerExpiredCounts ....... 0
  atpRetryCountExceededs ............. 0
```

Table 35-8: Parameters in output of the **show apple count=atp** command

| Parameter | Meaning |
|---|---|
| atpInPkts | Number of ATP packets received by this entity. |
| atpOutPkts | Number of ATP packets sent by this entity. |
| atpTRequestRetransmissions | Number of times that a timeout occurred and a Transaction Request packet needed to be retransmitted by this host. |
| atpTResponseRetransmissions | Number of times a timeout was detected and a Transaction Response packet needed to be retransmitted by this host. |
| atpReleaseTimerExpiredCounts | Number of times the release timer expired, as a result of which a Request Control Block had to be deleted. |
| atpRetryCountExceededs | Number of times the retry count was exceeded, and an error was returned to the client of ATP. |

Figure 35-8: Example output from the **show apple counter=ddp** command

```
DDP Counter
------------
  ddpOutRequests .................. 6285
  ddpOutShorts ....................... 0
  ddpOutLongs ..................... 6303
  ddpInReceives ................... 6250
  ddpInLocalDatagrams ............. 6231
  ddpNoProtocolHandlers ............... 4
  ddpTooShortErrors ................... 0
  ddpTooLongErrors .................... 0
  ddpShortDDPErrors ................... 0
  ddpChecksumErrors ................... 0
  ddpForwRequests .................. 18
  ddpOutNoRoutes ...................... 1
  ddpBroadcastErrors .................. 0
  ddpHopCountErrors ................... 0
```

Table 35-9: Parameters in output of the **show apple count=ddp** command

| Parameter | Meaning |
|---|---|
| ddpOutRequests | The total number of DDP datagrams that were supplied to DDP by local clients in request for transmission. This counter does not include any datagrams counted in ddpForwRequests. |
| ddpOutShorts | The total number of short DDP datagrams that were transmitted from this entity. |
| ddpOutLongs | The total number of long DDP datagrams that were transmitted from this entity. |
| ddpInReceives | The total number of input datagrams received by DDP, including those received in error. |
| ddpInLocalDatagrams | The total number of input DDP datagrams for which this entity was their final DDP destination. |
| ddpNoProtocolHandlers | The total number of DDP datagrams addressed to this entity that were addressed to an upper layer protocol for which no protocol handler existed. |
| ddpTooShortErrors | The total number of input DDP datagrams dropped because the received data length was less than the data length specified in the DDP header or the received data length was less than the expected DDP header. |
| ddpTooLongErrors | The total number of input DDP datagrams dropped because they exceeded the maximum DDP datagram size. |
| ddpShortDDPErrors | The total number of input DDP datagrams dropped because this entity was not their final destination and their type was short DDP. |
| ddpChecksumErrors | The total number of input DDP datagrams for which this DDP entity was their final destination, and that were dropped because of a checksum error. |
| ddpForwRequests | The number of input datagrams for which this entity was not their final DDP destination, as a result of which an attempt was made to find a route to forward them to that final destination. |
| ddpOutNoRoutes | The total number of DDP datagrams dropped because a route could not be found to their final destination. |
| ddpBroadcastErrors | The total number of input DDP datagrams dropped because this entity was not their final destination and they were addressed to the link level broadcast. |
| ddpHopCountErrors | The total number of input DDP datagrams dropped because this entity was not their final destination and their hop count would exceed 15. |

Figure 35-9: Example output from the **show apple counter=nbp** command

```
NBP Counter
-----------

  nbpInLookUpRequests ............... 0
  nbpInLookUpReplies ................ 0
  nbpInBroadcastRequests ............ 65
  nbpInForwardRequests .............. 0
  nbpOutLookUpReplies ............... 0
  nbpRegistrationFailures ........... 0
  nbpInErrors ....................... 0
```

Table 35-10: Parameters in output of the **show apple count=nbp** command

| Parameter | Meaning |
|---|---|
| nbpInLookUpRequests | Number of NBP LookUp Requests received. |
| nbpInLookUpReplies | Number of NBP LookUp Replies received. |
| nbpInBroadcastRequests | Number of NBP Broadcast Requests received. |
| nbpInForwardRequests | Number of NBP Forward Requests received. |
| nbpOutLookUpReplies | Number of NBP LookUp Replies sent. |
| nbpRegistrationFailures | Number of times this node experienced a failure in attempting to register an NBP entity. |
| nbpInErrors | Number of NBP packets received by this entity that were rejected for any error. |

Figure 35-10: Example output from the **show apple counter=port** command

```
Port Counter
------------

  Port        atportInPkts        atportOutPkts
  ---------------------------------------------
     1                6247                 6402
     2                   7                    8
  ---------------------------------------------
```

Table 35-11: Parameters in output of the **show apple count=port** command

| Parameter | Meaning |
|---|---|
| atportInPkts | Number of packets received by this entity on this port. |
| atportOutPkts | Number of packets transmitted by this entity on this port. |

Figure 35-11: Example output from the **show apple counter=route** command

```
RTMP Counter
-------------

  rtmpInDataPkts .................. 6163
  rtmpOutDataPkts ................. 6092
  rtmpInRequestPkts ................. 0
  rtmpNextIREqualChanges ......... 18481
  rtmpNextIRLessChanges .............. 1
  rtmpRouteDeletes ................... 0
  rtmpRoutingTableOverflows .......... 0
  rtmpOutRequestPkts ................. 0
  rtmpInVersionMismatches ............ 0
  rtmpInErrors ....................... 0
```

Table 35-12: Parameters in output of the **show apple count=route** command

| Parameter | Meaning |
|---|---|
| rtmpInDataPkts | The number of good RTMP data packets received by this entity. |
| rtmpOutDataPkts | The number of RTMP packets sent by this entity. |
| rtmpInRequestPkts | The number of good RTMP Request packets received by this entity. |
| rtmpNextIREqualChanges | The number of times RTMP changed the Next Internet Router in a routing entry because the hop count advertised in a routing table was equal to the current hop count for a particular network. |
| rtmpNextIRLessChanges | The number of times RTMP changed the Next Internet Router in a routing entry because the hop count advertised in a routing table was less than the current hop count for a particular network. |
| rtmpRouteDeletes | The number of times RTMP deleted a route because it was aged out of the table. This can help to detect routing problems. |
| rtmpRoutingTableOverflows | The number of times RTMP attempted to add a route to the RTMP table but failed due to lack of space. |
| rtmpOutRequestPkts | The number of RTMP Request packets sent by this entity. |
| rtmpInVersionMismatches | The number of RTMP packets received by this entity that were rejected due to a version mismatch. |
| rtmpInErrors | The number of RTMP packets received by this entity that were rejected for an error other than version mismatch. |

Figure 35-12: Example output from the **show apple counter=zip** command

```
RTMP Counter
-------------

  rtmpInDataPkts .................. 6163
  rtmpOutDataPkts ................. 6092
  rtmpInRequestPkts ................... 0
  rtmpNextIREqualChanges ......... 18481
  rtmpNextIRLessChanges ............... 1
  rtmpRouteDeletes .................... 0
  rtmpRoutingTableOverflows .......... 0
  rtmpOutRequestPkts .................. 0
  rtmpInVersionMismatches ............. 0
  rtmpInErrors ........................ 0
```

Table 35-13: Parameters in output of the **show apple count=zip** command

| Parameter | Meaning |
|---|---|
| zipInZipQueries | The number of ZIP Queries received by this entity. |
| zipInZipReplies | The number of ZIP Replies received by this entity. |
| zipInZipExtendedReplies | The number of ZIP Extended Replies received by this entity. |
| zipZoneConflictErrors | The number of times a conflict was been detected between this entity's zone information and another entity's zone information. |
| zipInObsoletes | The number of ZIP Takedown or ZIP Bringup packets received by this entity. Note that as the ZIP Takedown and ZIP Bringup packets have been obsoleted, the receipt of one of these packets indicates that a node sent it in error. |
| zipInGetNetInfos | The number of ZIP GetNetInfo packets received on this port by this entity. |
| zipOutGetNetInfoReplies | The number of ZIP GetNetInfo Reply packets sent out this port by this entity. |
| zipZoneOutInvalids | The number of times this entity has sent a ZIP GetNetInfo Reply with the zone invalid bit set in response to a GetNetInfo Request with an invalid zone name. |
| zipAddressInvalids | The number of times this entity had to broadcast a ZIP GetNetInfo Reply because the GetNetInfo Request had an invalid address. |
| zipOutGetNetInfos | The number of ZIP GetNetInfo packets sent out this port by this entity. |
| zipInGetNetInfoReplies | The number of ZIP GetNetInfo Reply packets received on this port by this entity. |
| zipZoneInInvalids | The number of times this entity has received a ZIP GetNetInfo Reply with the zone invalid bit set because the corresponding GetNetInfo Request had an invalid zone name. |
| zipInErrors | The number of ZIP packets received by this entity that were rejected for any error. |

**Examples** To display the **port** counter group, use the command:

```
sh app cou=po
```

**Related Commands**

show apple
show apple aarp
show apple circuit
show apple dlci
show apple port
show apple route
show apple zone

# show apple dlci

**Syntax**

```
SHow APPle DLCi
```

**Description**

This command displays all the DLCIs used and available for use by the AppleTalk routing module (Figure 35-13, Table 35-14).

Figure 35-13: Example output from the **show apple dlci** command

```
Port   Interface   DLCI    Allowed   Active   Usable
-----------------------------------------------------
  3    fr0           23    Yes       No       No
                     70    Yes       No       No
-----------------------------------------------------
```

Table 35-14: Parameters in output of the **show apple dlci** command

| Parameter | Meaning |
|---|---|
| Port | Number of the AppleTalk port. |
| Interface | Frame Relay interface used by the AppleTalk port. |
| DLCI | DLCI of a DLC defined for the Frame Relay interface. |
| Allowed | Whether the circuit is allowed to be used by the AppleTalk routing module. |
| Active | Whether Frame Relay indicates the circuit is active. |
| Usable | Whether the circuit can be used by the AppleTalk routing module. |

**Examples**

To display all available DLCIs, use the command:

```
sh app dlc
```

**Related Commands**

add apple dlci
delete apple dlci

# show apple packetfilter

**Syntax** `SHow APPle PACKETFilter[=`*`filter-id`*`]`

where *filter-id* is a decimal number range 0 to 99

**Description** This command displays information about DDP or packet filters. If a filter is specified, all the entries in the filter are displayed. If a filter is not specified, all entries in all filters are displayed (Figure 35-14, Table 35-15).

Figure 35-14: Example output from the **show apple packetfilter** command

```
AppleTalk Packet Filters
-------------------------------------------------------------------------------
Filt    Ent  Source Network  Node     Socket    Dir  Action  Log     Matches
             Dest   Network  Node     Socket
-------------------------------------------------------------------------------
  1       1  200-300         20-20    6-6       In   Deny    Yes          237
             200-300         Any      6-6
          2  Any             Any      Any       In   Allow   No          2434
             Any             Any      Any
    Requests:      23654     Allows:      23417     Denies:        237
-------------------------------------------------------------------------------
```

.
Table 35-15: Parameters in output of the **show apple filter** command

| Parameter | Meaning |
|---|---|
| Filt | Filter number. |
| Ent | The entry number. |
| Source Network | The source network address range. |
| Source Node | The source node number range. |
| Source Socket | Source socket number range. |
| Dest Network | Destination network number range. |
| Dest Node | The Destination node number range. |
| Dest Socket | Destination socket number range. |
| Dir | The direction of the packet to match against (In - incoming, Out - outgoing). |
| Action | The action to take when a match is found, one of Allow or Deny. |
| Log | Flag indicating whether packet matching this entry is logged. |
| Matches | The number of packets matching this entry. |
| Requests | The total number of packets filtered using this filter. |
| Allows | The number of packets allowed to be processed further. |
| Denies | The number of packets denied by this filter. |

**Examples** To display the packet filter number 5, use the command:

```
sh app packetf=5
```

**Related Commands**

add apple packetfilter
add apple port
delete apple packetfilter
set apple packetfilter
set apple port
show apple port
show apple routefilter
show apple zonefilter

# show apple port

**Syntax** `SHow APPle POrt[=`*port*`]`

where *port* is an AppleTalk port number from 1 to 127

**Description** This command displays information about AppleTalk ports. If a port number is specified, detailed information about the specified port is displayed (Figure 35-15, Table 35-16). If a port number is not specified, summary information about all ports is displayed (Figure 35-16, Table 35-16).

Figure 35-15: Example output from the **show apple port** command for a specified port

```
Appletalk Port Details
-----------------------------------
Port Number ............. 1
Interface ............... eth0
ifIndex ................. 1
Node ID ................. 217
Network Number .......... 22
Network Range Start ..... 22
Network Range End ....... 22
State ................... ACTIVE
Seed .................... NO
Seed Network Start ...... 0
Seed Network End ........ 0
Hint .................... YES
Hint Node ID ............ 179
Hint Network ............ 22
Default Zone ............ -
Zone List is Empty
Packet Filter ........... 1
Route Filter ............ None
Zone Filter ............. None
-----------------------------------
```

Figure 35-16: Example output from the **show apple port** command

```
Appletalk Port Summary
Port  Interface  Node     Network     State   Demand  Link  Route  Pf  Rf  Zf
-------------------------------------------------------------------------------
2     ppp0       0        0            ACTIVE  On      Down  Init   -   -   10
-------------------------------------------------------------------------------
```

Table 35-16: Parameters in output of the **show apple port** command

| Parameter | Meaning |
|---|---|
| Port Number | The number of the AppleTalk port. |
| Interface | The interface used by the port. |
| Node ID | The node ID of the port. |
| Network Number | The network number for the port. |
| Network Range Start | The first network number in the range of network numbers assigned to the network where the port is attached. |

Table 35-16: Parameters in output of the **show apple port** command (Continued)

| Parameter | Meaning |
|---|---|
| Network Range End | The last network number in the range of network numbers assigned to the network where the port is attached. |
| Network Address | The network address of the port. |
| State | The state of the port; one of "INITIAL", "PROVISIONAL", "ACTUAL" or "ACTIVE". |
| Demand | Whether the port is configured for dial-on-demand routing. This parameter is valid for PPP interfaces. |
| Link | The status of the link layer interface; one of "Up" (link layer interface is up), "RTMP" (RTMP packets have been transmitted via this port) or "Down" (link layer interface is down). This parameter is valid for PPP interfaces. |
| Route | The status of peer route updates; one of "Init" (no route updates have been seen), "RTMP" (RTMP packets have been seen since the link came up) or "Updt" (route updates have been seen since the first RTMP packet). This parameter is valid for PPP interfaces. |
| Seed | Whether this router is a seed router for the network where the port is attached. |
| Seed Network Start | If the Seed field is **yes**, the first network number in the range of seed network numbers for the network where the port is attached. |
| Seed Network End | If the Seed field is **yes**, the last network number in the range of seed network numbers for the network where the port is attached. |
| Hint | Whether this router has a hint protocol address that can be used in the address acquisition phase at startup. |
| Hint Node ID | f the Hint field is **yes**, the node ID portion of the hint protocol address for this router. |
| Hint Network | If the Hint field is **yes**, the network number portion of the hint protocol address for this router. |
| Default Zone | The default zone, if any, for the network where this port is attached. |
| Zone List | The lists of zones, if any, defined for the network where this port is attached. |
| Packet Filter | The packet filter used on this interface, or "NONE" if a filter has not been set. |
| Route Filter | The route advertisement filter used on this interface, or "NONE" if a filter has not been set. |
| Zone Filter | The zone advertisement filter used on this interface, or "NONE" if a filter has not been set. |

**Examples** To display all AppleTalk ports, use the command:

```
sh app po
```

**Related Commands**

**add apple port**
**delete apple port**

# show apple route

**Syntax** `SHow APPle ROUte`

**Description** This command displays the contents of the AppleTalk routing table, including both dynamic and static routes (Figure 35-17, Table 35-17).

Figure 35-17: Example output from the **show apple route** command

```
AppleTalk routing table

  Network range   Hops   Port   Interface   Next hop   Type
  ------------------------------------------------------------
  21              1      1      eth0        22:124     Remote
  22              0      1      eth0        0:0        Direct
  23              1      1      eth0        22:124     Remote
  200             2      2      ppp0        -          Static
  50000           1      1      eth0        22:124     Remote
  ------------------------------------------------------------
```

Table 35-17: Parameters in output of the **show apple route** command

| Parameter | Meaning |
|---|---|
| Network range | Network or network range of the destination network. |
| Hops | Number of hops to the destination network. |
| Port | AppleTalk port over which the destination network can be reached. |
| Interface | Lower layer interface used by the AppleTalk port. |
| Next hop | AppleTalk network number and node ID of the next router on the route to the destination network. |
| Type | Type of route; one of "Remote" (dynamically learned route to a remote network), "Direct" (dynamically learned route to a network where the router is directly connected) or "Static". |

**Examples** To display the current contents of the AppleTalk routing table, use the command:

```
sh app rou
```

**Related Commands** **add apple route**
**delete apple route**

# show apple routefilter

**Syntax** `SHow APPle ROUTEFilter[=`*`filter-id`*`]`

where *filter-id* is a decimal number range 0 to 99

**Description** This command displays information about RTMP or route filters. If a filter is specified, all the entries in the filter are displayed. If a filter is not specified, all entries in all filters are displayed (Figure 35-18, Table 35-18).

Figure 35-18: Example output from the **show apple routefilter** command

```
AppleTalk Route Filters
---------------------------------------------------------------
Filt    Ent   Network Range    Dir    Action    Log       Matches
---------------------------------------------------------------
  1       1  200-300           In     Deny      Yes           237
          2  200-300           Out    Allow     No            225
          3  Any               Out    Allow     No           1434
 Requests:       1896    Allows:       1659    Denies:       237
---------------------------------------------------------------
```

Table 35-18: Parameters in output of the **show apple routefilter** command

| Parameter | Meaning |
|---|---|
| Filt | Filter number. |
| Ent | Entry number. |
| Network Range | Route destination network address range. |
| Dir | Whether the direction of the route advertisement to match against is incoming or outgoing. |
| Action | Whether to allow or deny it when a match is found. |
| Log | Flag indicating whether route advertisement matching this entry is logged. |
| Matches | Number of route advertisement matching this entry. |
| Requests | Total number of route advertisement filtered using this filter. |
| Allows | Number of route advertisement allowed to be processed further. |
| Denies | Number of route advertisement denied by this filter. |

**Examples** To display all route filters, use the command:

```
sh app routef
```

**Related Commands**

**add apple port**
**add apple routefilter**
**delete apple routefilter**
**set apple port**
**set apple routefilter**
**show apple packetfilter**
**show apple port**
**show apple zonefilter**

# show apple zone

**Syntax** `SHow APPle ZOne`

**Description** This command displays information about all currently known and configured zones (Figure 35-19, Table 35-19).

Figure 35-19: Example output from the **show apple zone** command

```
Zone Information Table

  Zone                                Port  Network
  ------------------------------------------------
  Aslan Internal                      -     50000-50000
  BrickBat                            -     100-100
  Finance                             -     21-21
  Finance                             2     22-22
  ------------------------------------------------
```

Table 35-19: Parameters in output of the **show apple zone** command

| Parameter | Meaning |
|---|---|
| Zone | The name of the AppleTalk Zone. |
| Port | The AppleTalk port with which the zone is associated. |
| Network | The network number range(s) associated with the zone. |

**Examples** To display all AppleTalk zones, use the command:

```
sh app zo
```

**Related Commands** **add apple zone**
**delete apple zone**

# show apple zonefilter

**Syntax** `SHow APPle ZONEFilter[=`*`filter-id`*`]`

where *filter-id* is a decimal number range 0 to 99

**Description** This command displays information about ZIP or zone filters. If a filter is specified, all the entries in the filter are displayed. If a filter is not specified, all entries in all filters are displayed (Figure 35-20, Table 35-20).

Figure 35-20: Example output from the **show apple zonefilter** command

```
AppleTalk Zone Filters
------------------------------------------------------------------------
Filt    Ent   Zone Name                          Action  Log      Matches
------------------------------------------------------------------------
nnn   nnnnn   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   Xxxxx   Xxx   nnnnnnnnnn
------------------------------------------------------------------------
 10       1   Dungeon Of Doom Limited Area       Deny    Yes           23
          2                                       Allow   No            167
   Requests:        190    Allows:        167     Denies:         23
------------------------------------------------------------------------
```

Table 35-20: Parameters in output of the **show apple routefilter** command

| Parameter | Meaning |
|---|---|
| Filt | Filter number. |
| Ent | The entry number. |
| Zone Name | The zone name to be advertised. |
| Action | The action to take when a match is found, one of Allow or Deny. |
| Log | Flag indicating whether zone advertisement matching this entry is logged. |
| Matches | The number of zone advertisement matching this entry. |
| Requests | The total number of zone advertisement filtered using this filter. |
| Allows | The number of zone advertisement allowed to be processed further. |
| Denies | The number of zone advertisement denied by this filter. |

**Examples** To display all route filters, use the command:

```
sh app zonef
```

**Related Commands**

**add apple port**
**add apple zonefilter**
**delete apple zonefilter**
**set apple port**
**set apple zonefilter**
**show apple port**