# Chapter 1
# Configuring Transparent Bridging

1

This chapter describes the transparent bridging support available in the Cisco Systems network server product line. Topics described in this chapter include:

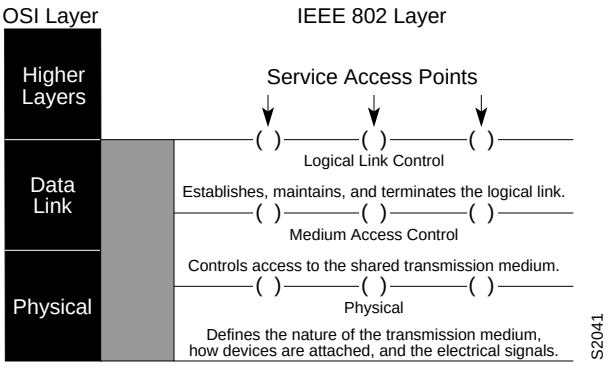
- An overview of bridging and Cisco's implementation of transparent bridging.
- How to define a spanning tree.
- How to adjust the spanning-tree parameters.
- How to filter packets based on type; supported types include: Ethernet address, protocol type, vendor code, and LAT multicast service announcements.

## Bridging Overview

The basic function of a bridge is to accept frames of data passing through a network, and then make a decision about whether to forward each frame based on information contained in the frame.

Bridges operate at the physical (Level 1) and data link (Level 2) layers of the Open Systems Interconnection (OSI) model. As defined by the IEEE 802.1 standard, the task of the physical layer is to provide the physical connection to the transmission medium. The task of the data link layer is to provide reliable transmission. To accomplish this effectively, the standards committee sublayered the data link layer into two distinct tasks, Logical Link Control (LLC) and Medium Access Control (MAC). Bridges, and specifically transparent bridges, function at the physical and MAC layers. However, the administrative filtering functions described later in this chapter make use of the LLC. Figure 1-1 illustrates these concepts.

*Figure 1-1* OSI Model and IEEE 802 Layers




The LLC, as its name implies, controls the logical link, that is, it serves to open, maintain, and terminate a link. It depends upon the physical layers to perform its tasks.

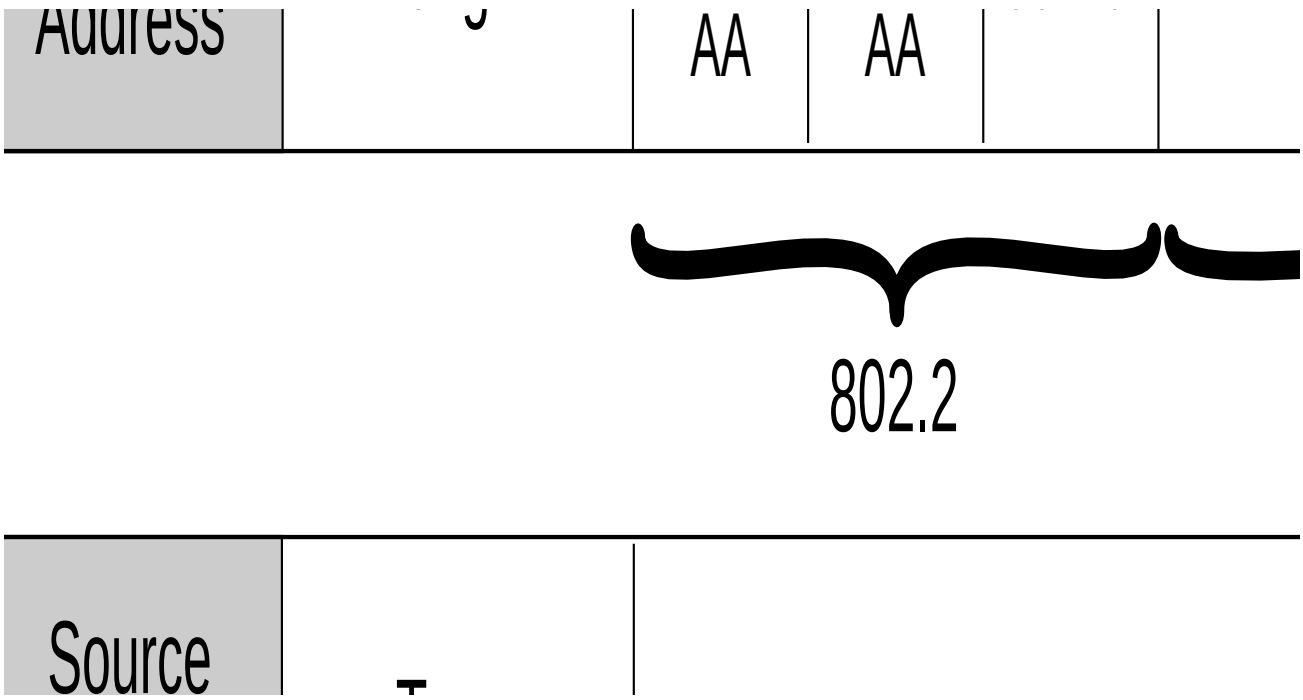
The MAC controls access to the media used to transmit the frames. It acts independently of the physical layer and provides a service interface to the upper layers of the network model. IEEE 802.3 and Ethernet use the same MAC layer, allowing both types of frames to coexist on the same cable. In this way, the same bridge can forward both Ethernet and IEEE 802.3 frames.

The standard identifies two basic elements to facilitate these tasks: a 48 bit address which is always at the same offset in the frame, and Service Access Points (SAPs), which can be thought of as ports through which a higher layer application may communicate with its lower layer.

Figure 1-2 illustrates the frame formats found on Ethernets.

The IEEE 802 standard allows any IEEE 802-compliant station to communicate with remote, bridged stations as if they were local. Although the networks connected by a bridge remain physically separate, they appear as one network to the rest of the devices on the internet.

*Figure 1-2* IEEE 802 Frame Formats




Bridges forward frames between networks using the same medium (two IEEE 802.3 networks, for example). Bridges can pass data using any protocol compatible with the media. Bridges receive all frames from a local area network (LAN) and examine each one to determine its destination. If the frame is destined for a station on the physical cable from which it was received, the bridge does not forward the frame. If the destination is unknown, the frame is forwarded to all ports except the one on which it was received.

Because bridges examine each frame, and because each frame contains a source address, bridges can easily learn the locations of stations on the network. However, if an address is unknown, their flooding behavior could also result in endless loops on the network, since all bridges that do not know the address will flood the frame looking for the address. To circumvent this, a tree-type algorithm has been developed that prevents loops and forces the frames into logical routes. Called a spanning-tree algorithm, it allows only one route between any two physical cables or networks.

Spanning tree bridges are based upon the concept of a *root* bridge which exchanges topology information with designated bridges to maintain the configuration. The root and designated bridges notify all other bridges in the network when topology changes are required, thereby preventing loops and providing a measure of defense against link failure.

Bridges may be further classified as *local* or *remote.* A local bridge typically connects two nearby LANs, for example, Ethernet-to-Ethernet or Token Ring-to-Token Ring. A remote bridge, on the other hand, links geographically distant LANs or LANs separated by some other media. Two Ethernets linked by a serial line, or two Token Rings linked by a TCP connection are examples of remote bridging.

## *Cisco's Implementation of Transparent Bridging*

Cisco's transparent bridging software implementation provides these features:

- Complies with the IEEE 802 standard.
- Provides two spanning-tree algorithms—an older version that is compatible with DEC and other LAN bridges for backwards compatibility, and a recent IEEE Standard implementation that provides for multiple domains for spanning trees. The software also provides for adjustments to the spanning-tree parameters.
- Allows configuration of filters to effectively block the passing of frames based on Ethernet address, protocol type, or the vendor code. Additionally, the bridging software can be configured to selectively include or exclude LAT multicast service announcements.
- Provides load balancing and redundancy by assigning a set of serial lines between two bridges to a *circuit group.*
- Provides ability to bridge over X.25 and frame relay networks.
- Provides for the compression of LAT frames to reduce LAT traffic through the network.

Additionally, the Cisco router/bridge combines the advantages of a spanning-tree bridge and a full multiprotocol router. This combination provides the speed and protocol transparency of an adaptive spanning-tree bridge, along with the functionality, reliability, and security of routers.

Network interfaces on the Cisco Multiport Communications Interface (MCI) and on the Multiport Ethernet Controller (MEC) cards on the cBus high-speed backplane, may be configured to serve as both multiprotocol routers and MAC-level bridges, bridging any traffic that cannot otherwise be routed. For example, a router/bridge routing the Internet Protocol can also bridge DEC's Local Area Transport (LAT) protocol, or NETBIOS traffic.

Remote bridging over synchronous serial lines is also supported. Ethernet frames are encapsulated in HDLC frames for transmission between Cisco bridges/routers. As with frames received on Ethernet interfaces, the learning process and any filtering is applied to HDLC-encapsulated Ethernet frames.

The transit bridging of Ethernet frames across an FDDI ring is also supported. The term *transit* refers to the fact that the source or destination of the frame cannot be on the FDDI ring itself. This allows the FDDI ring to act as a highly efficient backbone for the interconnection of many bridged Ethernets. The configuration of FDDI transit bridging is identical to the configuration of Ethernet transparent bridging.

## *Configuring Transparent Bridging*

Follow these steps to configure transparent bridging on a Cisco router/bridge:

*Step 1:* Define the spanning-tree protocol and assign a bridge group number using the **bridge** *group* **protocol** command. The Cisco bridging software supports the IEEE 802.1 spanning-tree protocol, and the earlier DEC protocol upon which the IEEE standard is based. Additionally, multiple domains are supported for the IEEE spanning tree.

*Step 2:* Enable each interface and configure any routed protocols. If you wish to bridge IP, you must disable IP routing.

*Step 3:* Assign the network interfaces to the spanning-tree group.

*Step 4:* Adjust the spanning-tree parameters, as necessary.

The bridging software also supports filtering of frames. Filtering can be done by Ethernet address, protocol type, vendor code, and based upon LAT group codes. Additionally, EXEC-level commands for monitoring and debugging the bridge are available. These tasks and commands are described in the following sections.

## *Defining the Spanning Tree Protocol*

Two spanning-tree protocols are supported: the IEEE 802.1 standard, and the earlier DEC spanning-tree protocol upon which the IEEE standard is based.

To define a spanning-tree protocol, use the **bridge** *group* **protocol** global configuration command. The command has this syntax:

**bridge** *group* **protocol** *{***ieee|dec***}*
**no bridge** *group* **protocol** *{***ieee|dec**}*

The argument *group* is a number between one and nine that you choose to refer to a particular set of bridged interfaces. Frames are bridged only among interfaces in the same group. You will use the group number you assign in subsequent bridge configuration commands.

The keyword **protocol** specifies the protocol to use and is either **ieee** for the Ethernet spanning-tree protocol, or **dec** for DEC spanning-tree protocol.

The **no bridge** command with the appropriate keywords and arguments deletes the bridge group.

*Example:*

This command defines bridge 1 as using the DECnet spanning-tree protocol.

```
bridge 1 protocol dec
```

## *Establishing Multiple Spanning Tree Domains*

The Cisco IEEE 802 bridging software supports multiple domain spanning-trees. You can place any number of router/bridges within the domain. The devices in the domain, and only those devices, will then share spanning-tree information.

This feature is used when multiple routers share the same cable, and you wish to use only certain, discrete subsets of those routers to share spanning tree information with each other. This function is most useful when running other router applications, such as IP UDP flooding, that use the IEEE spanning tree. It can also be used to reduce the number of global reconfigurations in large bridged networks.

---

*Note:* Use with care, since bridges in different domains do not share spanning tree information. This makes it possible for bridge loops to be created if the domains are not carefully planned.

---

Establish a domain by assigning it a value between one to ten using this variation of the **bridge** global configuration command:

**bridge** *group* **domain** *domain-number*

The argument *group* is the bridge group number, and must be the same as that specified by the **protocol ieee** keywords in the previously described **bridge** *group* command.

The keyword **domain** specifies a domain; the argument *domain-number* is a domain number you choose. The default domain number is zero (0), and this is the required domain number when communicating to IEEE bridges that do not support this domain extension.

---

*Note:* This command works only when the bridge group is running the IEEE spanning-tree protocol.

---

*Example:*

This command places bridge group *1* in bridging domain *3*. Only other Cisco routers that are in domain *3* will accept spanning-tree information from this router.

```
bridge 1 domain 3
```

## *Assigning the Interface to a Spanning Tree Group*

Assign each network interface to a spanning-tree group using the **bridge-group** *group* interface subcommand, which has this syntax:

**bridge-group** *group*

The simplest form of the **bridge-group** command takes just a spanning-tree group number

as an argument. More complex forms of the **bridge-group** command are described later.

---

*Note:* Restrictions apply as to which interfaces can be configured for bridging. Only the Cisco MCI, SCI, MEC HSCI, and FDDI interface controller cards are capable of supporting simultaneous bridging and routing. Bridging can be configured between interfaces on different cards, although the performance is lower compared with interfaces on the same card. Also note that serial interfaces must be running with HDLC, X.25, or frame relay encapsulation.

---

All protocols except IP are bridged by a router/bridge unless their routing is explicitly enabled. Refer to the chapter "Routing IP" for the procedures to enable routing of individual protocols. IP is normally routed by the router/bridge.

Also note that bridging and routing are done on a per-system basis. If a protocol is being routed, it must be routed on all interfaces that are handling that protocol. This is similar for bridging. You cannot route IP on one interface and bridge it on another interface.

*Example:*

The following is an example of a basic bridging configuration. The system has two Ethernets and one serial line on the same card. The Internet Protocol (IP) is being routed and everything else is being bridged. The DEC-compatible bridging algorithm with default parameters is being used.

```
interface ethernet 0
ip address 192.31.7.26  255.255.255.240
bridge-group  1
!
interface serial 0
ip address 192.31.7.34  255.255.255.240
bridge-group  1
!
interface ethernet 1
ip address 192.31.7.65  255.255.255.240
bridge-group  1
!
bridge 1 protocol dec
```

## *Bridging and Routing IP*

To bridge IP you must disable IP routing by giving the following global configuration command:

**no ip routing**

Assign an IP address (the *same* IP address) to all network interfaces to manage the system with Telnet, TFTP, SNMP, ICMP (ping) and so forth. Once bridging is enabled, all IP and ARP frames not intended for the Cisco router/bridge are handled according to standard bridging and spanning tree rules. IP routing processes (such as IGRP or RIP) must not be running.

## Adjusting Spanning-Tree Parameters

Under some circumstances, adjustments to certain spanning-tree parameters may be needed. Parameters affecting the entire spanning tree are configured with variations of the **bridge** *group* global configuration command. Interface-specific parameters are configured with variations of the **bridge-group** *group* interface subcommand. Global adjustments to the entire spanning tree are described first.

---

*Note:* These adjustments must be done carefully, and only by network administrators with a good understanding of how bridges and the spanning-tree protocol work. Badly planned adjustments to these parameters can have a negative impact on performance. A good source on bridging is the IEEE 802.1d specification; see the recommended reading list at the end of this publication for other references.

---

### Electing the Root Bridge

The priority of an individual bridge, or the likelihood that it will be selected as the root bridge, can be configured with the **bridge** *group* **priority** global configuration command. The command has this syntax:

**bridge** *group* **priority** *number*

The argument *number* can range from 1 to 65,000. The lower the value of *number*, the more likely the bridge will be chosen as root. The default priority is 128.

*Example:*

This command establishes this bridge as a likely candidate to be the root bridge.

```
bridge 1 priority 100
```

### Adjusting the Interval Between HELLO BPDUs

The interval between HELLO Bridge Protocol Data Units (BPDUs) is specified with the **bridge** *group* **hello-time** global configuration command. The command has this syntax:

**bridge** *group* **hello-time** *seconds*

The argument *seconds* can be any value between one and ten seconds. The default value is one second.

*Example:*

This command sets the interval to five seconds.

```
bridge 1 hello-time 5
```

*Note:* Each bridge in a spanning tree adopts the **hello-time**, **forward-time**, and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

## *Defining the Forward Delay Interval*

The forward delay interval is the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. Use the **bridge** *group* **forward-time** global configuration command to specify this interval. The command has this syntax:

**bridge** *group* **forward-time** *seconds*

The argument *seconds* can be any value between 10 and 200 seconds. The default value is 30 seconds.

*Note:* Each bridge in a spanning tree adopts the **hello-time**, **forward-time**, and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

*Example:*

This command sets the forward delay interval to 60 seconds.

```
bridge 1 forward-time 60
```

## *Defining the Maximum Idle Interval*

If a bridge does not hear BPDUs from the root bridge within a specified interval, it assumes that the network has changed and recomputes the spanning-tree topology. This interval is 15 seconds by default. It can be changed with the **bridge** *group* **max-age** global configuration command. The command has this syntax:

**bridge** *group* **max-age** *seconds*

The argument *seconds* is the interval the bridge will wait to hear BPDUs from the root bridge.

*Note:* Each bridge in a spanning tree adopts the **hello-time**, **forward-time**, and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

*Example:*

This command increases the maximum idle interval to 20 seconds.

```
!
```

```
bridge 1 max-age 20
```

## *Assigning Path Costs*

Each interface has associated with it a path cost. By convention, the path cost is 1000/data rate of the attached LAN, in mbps. Table 1-1 lists some common path cost values.

*Table 1-1* Media and Path Cost Values

| **Media** | **Path Cost** |
|---|---|
| Ethernet | 100 |
| FDDI | 10 |

Use the interface subcommand **bridge-group** *group* **path-cost** to set a different path cost. The command syntax is as follows:

**bridge-group** *group* **path-cost** *cost*

The path cost can range from 0 to 65,535, with higher values indicating higher costs.

The default path cost is computed from the interface's bandwidth setting.

*Example:*

This command changes the default path cost for interface Ethernet 0.

```
interface ethernet 0
bridge-group 1 path cost 250
```

## *Setting an Interface Priority*

A priority can also be associated with an interface. In the event that two bridges tie for position as the root bridge, this priority is used in tie-breaking when computing a network topology. Use the **bridge-group** *group* **priority** interface subcommand to do this. The command has this syntax:

**bridge-group** *group* **priority** *number*

The argument *number* can range from 0 to 255. The default value is zero, and the lower the number, the more likely it is that the bridge on this interface will be chosen as the root.

*Example:*

This command increases the likelihood that the root bridge will be the one on Ethernet 0, in spanning-tree group 1.

```
interface ethernet 0
bridge-group 1 priority 0
```

## Establishing Administrative Filtering

It is the job of a bridge to examine frames and transmit them through the internet according to their destination address; that is, a bridge will not forward a frame back to its originating network segment. The Cisco bridge software allows specific administrative filters to be configured that block frames based upon information other than paths to their destinations. This administrative filtering can be done by:

- MAC-layer address
- Protocol type—Ethernet or IEEE 802.3
- Vendor code
- LAT multicast service announcement

When setting up administrative filtering, remember that there is virtually no performance penalty in filtering by Ethernet address or vendor code, but there can be a significant performance penalty when filtering by protocol type.

## Administrative Filtering by MAC-layer Address

Blocking transmission of frames based on MAC-layer address is configured by variations of the **bridge** *group* global configuration command, as described in the following sections.

### Filtering on Ethernet Address

To filter frames with a particular MAC-layer station source or destination address, use the global configuration command **bridge** *group* **address**. The full syntax of this command is as follows:

**bridge** *group* **address** *ethernet-address* [**forward|discard**] [*interface*]
**no bridge** *group* **address** *ethernet-address*

The argument *group* is the group number you assigned to the spanning tree.

The argument *ethernet-address i*s a 48-bit dotted-triple hardware address such as that displayed by the EXEC **show arp** command. An example is:

```
0800.cb00.45e9
```

The argument *ethernet-address* is either a station address, the broadcast address, or a multicast destination address.

If the optional **forward** keyword is specified, a frame sent from or destined to the specified address is forwarded, as appropriate.

If the optional **discard** keyword is specified, a frame sent from or destined to the specified address is discarded without further processing.

The argument *interface* is an optional interface specification, such as *Ethernet 0*, and is added after the **discard** or **forward** keyword to indicate the interface on which that address can be

reached.

Any number of addresses can be configured into the system without a performance penalty.

Use the **no bridge** *group* **address** command followed by the Ethernet address to disable the forwarding ability.

*Examples:*

This command enables filtering of frames with Ethernet address *0800.cb00.45e9*.

```
bridge 1 address 0800.cb00.45e9 forward ethernet 1
```

The frame is forwarded through interface Ethernet 1.

This command disables the ability to forward frames with Ethernet address *0800.cb00.45e9*.

```
no bridge 1 address 0800.cb00.45e9
```

## *Preventing the Forwarding of Dynamically Determined Stations*

Normally the system forwards any frames for stations that it has learned about dynamically. This default can be changed with the **no bridge** *group* **acquire** global configuration command. The full syntax of this command follows:

**no bridge** *group* **acquire**
**bridge** *group* **acquire**

The bridge filters out all frames except those whose sourced-by or destined-to addresses have been statically configured into the forwarding cache.

The **bridge** *group* **acquire** global configuration command restores the default behavior.

*Example:*

This command prevents the fowarding of dynamically determined source and destination addresses.

```
no bridge 1 acquire
```

## *Fowarding the Multicast Addresses*

The bridging support may be configured to allow the forwarding, but not the learning, of multicast source addresses. Use this variation of the **bridge** *group* global configuration command to configure this function (the negative form is also listed):

**bridge** *group* **multicast-source**
**no bridge** *group* **multicast-source**

List the bridge group using the *group* argument.

## *Administrative Filtering by Protocol Type*

Filtering by protocol type is done using the access list mechanism and by specifying a protocol type code. The bridge **access-list** command specifies an element in an access list. The order in which **access-list** commands are entered affects the order in which the access conditions are checked. Each condition is tested in succession. A matching condition is then used to execute a permit or deny decision. If no conditions match, a deny decision is reached.

### *Establishing Protocol Type Access Lists*

Type code access lists are built with this bridge **access-list** global configuration command:

**access-list** *list {***permit**|**deny***} type-code wild-mask*

The argument *list* is a user-selectable number between 200 and 299 that identifies the list.

The keyword **permit** permits the frame; the keyword **deny** denies the frame.

The argument *type-code* is a 16-bit hexadecimal number written with a leading "0x", for example, 0x6000. You may specify either an Ethernet type code for Ethernet-encapsulated packets, or a DSAP/SSAP pair for 802.3-encapsulated packets. Ethernet type codes are listed in the appendix "Ethernet Type Codes."

The argument *wild-mask* is another 16-bit hexadecimal number whose ones bits correspond to bits in the *type-code* argument that should be ignored when making a comparison. (A mask for a DSAP/SSAP pair should always be at least 0x0101. This is because these two bits are used for purposes other than identifying the SAP codes.)

*Examples:*

The following access list permits only LAT frames (type 0x6004) and filters out all other frame types.

```
access-list  201  permit  0x6004  0x0000
access-list  201  deny    0x0000  0xFFFF
```

The following access list filters out only type codes assigned to DEC (0x6000 through 0x600F) and lets all other types pass.

```
access-list  202  deny     0x6000  0x000F
access-list  202  permit   0x0000  0xFFFF
```

It is always a good idea to use the last item of an access list to specify a default action, for example, permit everything else or deny everything else. If nothing else in the access matches, the default action is normally to deny access, that is, filter out all other type codes.

---

*Note:* Type code access lists can have an impact on system performance, therefore, keep the lists as short as possible and use wild card bit masks whenever possible.

---

## *Filtering Ethernet- and SNAP-Encapsulated Packets on Input*

You may filter Ethernet- and SNAP-encapsulated packets on input. The access list specifying the type codes to be filtered is given in this variation of the **bridge-group** interface subcommand:

**bridge-group** *group* **input-type-list** *list*

This access list is then applied to all Ethernet and SNAP frames received on that interface prior to the bridge learning process. SNAP frames must also pass any applicable IEEE802.3 DSAP/SSAP access lists.

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

### *Example:*

The following example illustrates how to configure a system with two Ethernet interfaces and one serial interface. The system is routing both IP and DECnet. Each interface has an access list that allows only the LAT protocol to be bridged. The bridging software has also been instructed to discard frames sent to or from the address *AB00.0C00.AE35*.

```
decnet address 34.88
!
interface ethernet 0
ip address 192.31.7.26  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface serial 0
ip address 192.31.7.34  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface ethernet 1
ip address 192.31.7.65  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
bridge 1 protocol dec
bridge 1 address AB00.0C00.AE35 discard
!
access-list 201 permit 0x6004  0x0000
access-list 201 deny   0x0000  0xFFFF
```

## *Filtering Ethernet- and SNAP-Encapsulated Packets on Output*

You may filter Ethernet- and SNAP-encapsulated packets on output. The access specifying the type codes to be filtered is given by this variation of the **bridge-group** interface subcommand:

**bridge-group** *group* **output-type-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

This access list is then applied just before sending out a frame to an interface.

*Example:*

This command specifies access list 202 on interface Ethernet 0.

```
interface ethernet 0
bridge-group 2 output-type-list 202
```

## *Filtering IEEE 802.3-Encapsulated Packets on Input*

You may filter IEEE 802.3-encapsulated packets on input. The access list specifying the type codes to be filtered is given by this variation of the **bridge-group** interface subcommand:

**bridge-group** *group* **input-lsap-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

This access list is applied to all IEEE 802.3 frames received on that interface prior to the bridge-learning process. SNAP frames must also pass any applicable Ethernet type-code access list.

*Example:*

This command specifies access list 203 on interface Ethernet 1:

```
interface ethernet 1
bridge-group 3 input-lsap-list 203
```

## *Filtering IEEE 802.3-Encapsulated Packets on Output*

You may filter IEEE 802.3-encapsulated packets on output. The access list specifying the type codes to be filtered is given by this variation of the **bridge-group** interface subcommand:

**bridge-group** *group* **output-lsap-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command. SNAP frames must also pass any applicable Ethernet type-code access list. This access list is applied just before sending out a frame to an interface.

*Note:* For performance reasons, it is not a good idea to have both input and output type code filtering on the same interface.

Access lists for Ethernet- and IEEE 802.3-encapsulated packets affect only bridging functions. It is not possible to use such access lists to block frames with protocols that are being routed.

*Example:*

This command specifies access list 204 on interface Ethernet 0.

```
interface ethernet 0
bridge-group 4 output-lsap-list 204
```

## Administrative Filtering by Vendor Code

The bridging software supports administrative filtering of Ethernet addresses. These lists support filtering groups of Ethernet addresses, including those with particular vendor codes. The lists are defined with bridge **access-list** global configuration command and **bridge-group** interface subcommand. There is no noticeable performance loss in using these access lists. The lists can be of indefinite length. The following sections describe how to set these lists up.

### Establishing Vendor Code Access Lists

Use the bridge **access-list** global configuration command to establish Ethernet and IEEE 802.3 address access lists. The command has the following form:

**access-list** *list {***permit|deny***} address mask*
**no access-list** *list {***permit|deny***} address mask*

The argument *list* is an integer from 700 to 799 that you select for the list.

The argument *address* and *mask* are 48-bit Ethernet addresses written in dotted triplet form. The ones bits in the *mask* argument are the bits to be ignored in *address*.

See the section "Filtering Source Addresses" for an example of use of this command.

### Filtering Source Addresses

Use the **bridge-group** *group* **input-address-list** interface subcommand to assign an access list to a particular interface for filtering on the Ethernet or IEEE 802.3 source addresses of packets received on that interface. The command has this syntax:

**bridge-group** *group* **input-address-list** *list*
**no bridge-group** *group* **input-address-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

*Example:*

This configuration example assumes you want to disallow the bridging of Ethernet packets of all SUN workstations on Ethernet 1. Software assumes that all such hosts have Ethernet addresses with the vendor code 0800.2000.0000. The first line of the access list denies access to all SUN workstations while the second line permits everything else. You then assign the access list to the input side of Ethernet 1.

```
access-list 700 deny 0800.2000.0000  0000.00FF.FFFF
access-list 700 permit 0000.0000.0000  FFFF.FFFF.FFFF
interface ethernet 1
bridge-group 1 input-address-list 700
```

### *Filtering Destination Addresses*

Use the **bridge-group** *group* **output-address-list** interface subcommand to assign an access list to a particular interface for filtering the Ethernet or IEEE 802.3 destination addresses of packets that would ordinarily be forwarded out that interface. The command has this syntax:

**bridge-group** *group* **output-address-list** *list*
**no bridge-group** *group* **output-address-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge **access-list** command.

*Example:*

This command assigns access list 703 to interface Ethernet 3.

```
interface ethernet 3
bridge-group 5 output-address-list 703
```

## *Administrative Filtering of LAT Service Announcements*

The Cisco bridging software provides filtering of LAT frames. LAT bridge filtering allows the selective inclusion or exclusion of LAT multicast service announcements, on a per-interface basis.

In the DEC LAT protocol, a *group code* is defined as a decimal number in the range 0 to 255. Some of the Cisco LAT configuration commands take a list of group codes; this is referred to as a *group code list*. The rules for entering numbers in a group code list follow:

- Entries can be individual group code numbers separated with a space. (The DEC LAT implementation specifies that a list of numbers be separated by commas; however, Cisco's implementation expects the list to be separated by a space.)
- Entries may also specify a range of numbers. This is done by separating an ascending

order range of group numbers with a hyphen.

- Any number of group codes or group code ranges can be listed in one command; just separate each with a space.

In LAT, each node transmits a periodic service advertisement message, which announces its existence and availability for connections. Within the message is a group code list; this is a mask of up to 256 bits. Each bit represents a group number.

In the traditional use of LAT group codes, a terminal server will only connect to a host system when there is an overlap between the group code list of the user on the terminal server, and the group code list in the service advertisement message.

While some believe this to be a security feature, it is in fact present to allow you to partition your physical network. This is because most DEC terminal servers do not have enough memory to cope with a large number of services. Group codes do not actually provide any real security; they are trivial to defeat.

In an environment with many bridges and many LAT hosts, the number of multicast messages with which each system has to deal becomes unreasonable. The 256 group codes may not be enough to allocate local assignment policies, such as giving each DECserver 200 device its own group code, in large bridged networks.

LAT group code filtering allows you to have very fine control over what multicast messages actually get bridged. Through a combination of input and output permit and deny lists, many different LAT control policies can be implemented. Use the EXEC command **show span** to report the group code filtering in effect.

## *Specifying LAT Group Code Service Filtering*

Use the **bridge** *group* **lat-service-filtering** global configuration command to specify LAT group code filtering. The command has this syntax:

**bridge** *group* **lat-service-filtering**

The command informs the system that LAT service advertisements require special processing. Use the *group* argument to specify the bridge group in which this special processing is to take place.

### *Example:*

This command specifies that LAT service announcements travelling across bridge group 1 require some special processing.

```
bridge 1 lat-service-filtering
```

## *Specifying Deny Conditions for LAT Group Codes on Input*

Use the **bridge-group** *number* **input-lat-service-deny** interface subcommand to specify the group codes by which to deny access upon input  The command has this syntax:

**bridge-group** *number* **input-lat-service-deny** *grouplist*

This command causes the system to not bridge any LAT service advertisement which has any of the specified groups set. The argument *number* is the previously chosen spanning-tree group number. Enter the list of LAT service groups with the *grouplist a*rgument.

*Example:*

This command causes any advertisements with groups 6, 8, and 14 through 20 to be dropped.

```
interface ethernet 0
bridge-group 1 input-lat-service-deny 6 8 14-20
```

## *Specifying Permit Conditions for LAT Group Codes on Input*

Use the **bridge-group** *number* **input-lat-service-permit** interface subcommand to specify the group codes by which to permit access upon input  The command has this syntax:

**bridge-group** *number* **input-lat-service-permit** *grouplist*

This command causes the system to bridge only those service advertisements which match at least one group in the group list specified by the *grouplist* argument.

---

*Note:* If a message specifies group codes in both the deny and permit list, the message is not bridged.

---

*Example:*

This command bridges any advertisements from groups 1, 5, and 12 through 14.

```
interface ethernet 1
bridge-group 1 input-lat-service-permit 1 5 12-14
```

## *Specifying Deny Conditions for LAT Group Codes on Output*

Use the **bridge-group** *number* **output-lat-service-deny** interface subcommand to specify the group codes by which to deny access upon output  The command has this syntax:

**bridge-group** *number* **output-lat-service-deny** *grouplist*

This command causes the system to not bridge onto this output interface any service advertisements which contain groups matching any of these in the group list. The LAT service advertisements are specified with the argument *grouplist*.

The argument *number* is the previously chosen spanning-tree group number.

*Example:*

This command prevents bridging of LAT service announcements from groups 12 through 20.

```
interface ethernet 0
bridge-group 1
bridge-group 1 output-lat-service-deny 12-20
```

### *Specifying Permit Conditions for LAT Group Codes on Output*

Use the **bridge-group** *number* **output-lat-service-permit** interface subcommand to specify the group codes by which to permit access upon output. The command has this syntax:

**bridge-group** *number* **output-lat-service-permit** *grouplist*

This command causes the system to bridge onto this output interface only those service advertisements that match at least one group in the specified group code list. The service advertisements are specified with the argument *grouplist*.

The argument *number* is the previously chosen spanning-tree group number.

---

*Note:* If a message matches both a deny and a permit condition, the message will not be bridged.

---

*Example:*

This command allows only LAT service announcements from groups 5, 12, and 20 on this bridge.

```
interface ethernet 0
bridge-group 1 output-lat-service-permit 5 12 20
```

## *Special Bridging Configurations*

This section describes some special bridging configurations, including how to configure load balancing over serial lines, and how to compress LAT traffic.

### *Establishing Load Balancing*

In the normal operation of the spanning-tree algorithm, parallel network segments cannot all be carrying traffic at the same time. This is necessary to prevent the looping of frames. In the case of serial lines, however, there is often a desire to increase the available bandwidth by using multiple, parallel serial lines.

To modify the spanning-tree algorithm's handling of serial lines, a set of serial lines between two bridges may be grouped in an association called a *circuit group.* If the spanning-tree algorithm allows any serial interface in the circuit group to forward packets, then all interfaces can be used for forwarding traffic. Ordering problems are avoided by assigning each

destination address to a particular serial interface. Reassignment is done dynamically if interfaces go down or come back up.

To establish load balancing, use the **bridge-group** *group* **circuit i**nterface subcommand. The command has this syntax:

**bridge-group** *group* **circuit** *number*

The command marks a serial interface as belonging to circuit group. The argument *group* is the spanning-tree group number. The argument *number* defines the circuit group number and is a small integer less than ten.

The parallel serial interfaces on both bridges must all be marked as being members of the same circuit group.

*Example:*

To load share over the two parallel serial links in this example, each router would have the configuration shown.

```
interface ethernet 0
bridge-group 1
!
interface serial 0
bridge-group 1
bridge-group 1 circuit 1
!
interface serial 1
bridge-group 1
bridge-group 1 circuit 1
!
bridge 1 protocol dec
```

## *Configuring X.25 Bridging*

Cisco's transparent bridging software supports bridging of packets in X.25 frames. This ability is useful, as an example, for transmitting packets from proprietary protocols across an X.25 network.

To configure this capability, use this variation of the **x25 map** interface subcommand in the bridging configuration file:

**x25 map bridge** *X.121-address* **broadcast** [*options-keywords*]

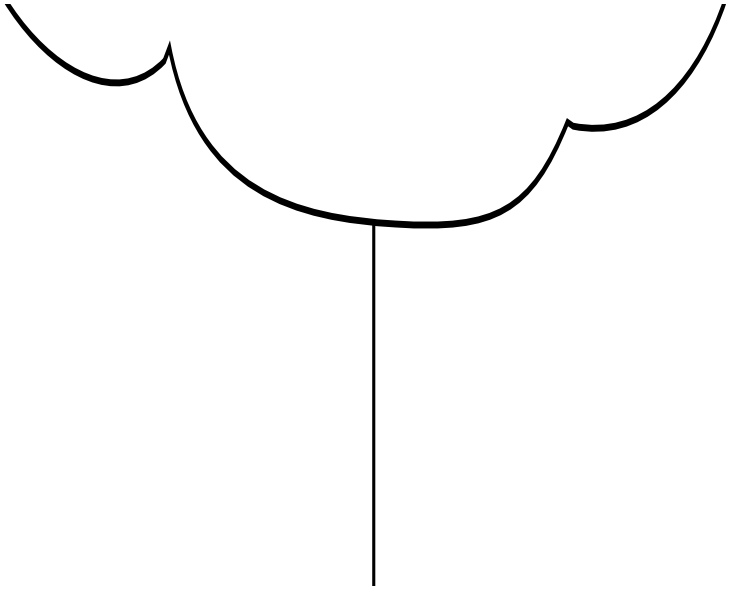
The keyword **bridge** specifies bridging over X.25. The argument *X.121-address* is the X.121 address. The keyword **broadcast** is required for bridging over X.25. The optional argument *options-keywords* are the services that may be added to this map, and are listed in the section "Setting Address Mappings" in the chapter "Configuring Packet-Switched Software."

The X.25 bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated in X.25 frames and transmitted across X.25 media. The command specifies Internet-to-X.121 address mapping and maintains a table of both the Ethernet and X.121 addresses.

## *X.25Bridging Example*

The following is an example configuration illustrating three Cisco bridges connected to each other through an X.25 network.

*Figure 1-3* X.25 Bridging Example



Following are the configuration commands for each of the bridges depicted in Figure 1-3.

*Example for Bridge 1:*

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9  255.255.255.0
!
interface serial 0
encapsulation x25
x25 address 31370019027
bridge-group 5
x25 map bridge 31370019134 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 5 protocol ieee
```

*Example for Bridge 2:*

```
interface serial 1
encapsulation x25
x25 address 31370019134
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 4 protocol ieee
```

*Example for Bridge 3:*

```
interface serial 0
encapsulation x25
x25 address 31370019565
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019134 broadcast
!
bridge 5 protocol ieee
```

## *Configuring Frame Relay Bridging*

Cisco's transparent bridging software supports bridging of packets over frame relay networks. This ability is useful, as an example, for transmitting packets from proprietary protocols across a frame relay network.

Bridging over a frame relay network is supported on both networks supporting a multicast facility and those that do not support such a facility. To configure bridging in a network not supporting a multicast facility, use this variation of the **frame-relay map** interface subcommand:

**frame-relay map bridge** *DLCI* **broadcast**

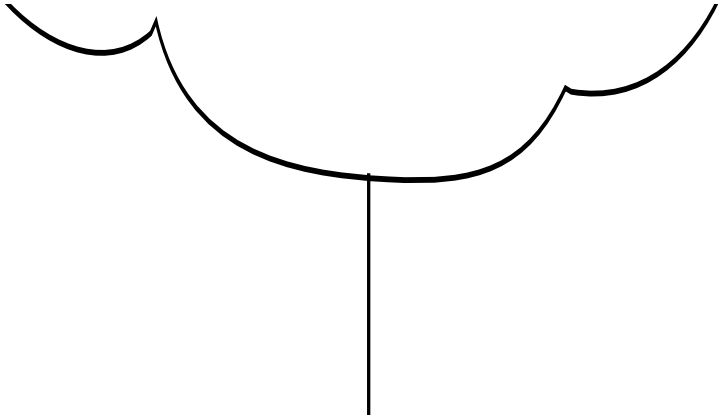
The keyword **bridge** specifies bridging over frame relay. The argument *DLCI* is the DLCI of the destination bridge. The keyword **broadcast** is required for bridging.

The frame relay bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated for transmission across a frame relay network. The command specifies Internet-to-DLCI address mapping and maintains a table of both the Ethernet and DLCIs.

### *Frame Relay Bridging Examples*

The following is an example configuration illustrating three Cisco bridges connected to each other through a frame relay network.

*Figure 1-4* Frame Relay Bridging Example



Following are the configuration commands for each of the bridges depicted in Figure 1-4.

*Example for Bridge 1:*

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9  255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 134 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
```

*Example for Bridge 2:*

```
interface serial 1
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
```

*Example for Bridge 3:*

```
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 134 broadcast
!
bridge 5 protocol ieee
```

## *Bridging in a Frame Relay Network with Multicasts*

The following example illustrates how to configure bridging in a frame relay network which supports the multicast facility.

*Example for Bridge 1:*

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9  255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
!
```

*Example for Bridge 2:*

```
interface serial 1
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
```

*Example for Bridge 3:*

```
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
```

In the above example, the multicast facility is used to learn about the other bridges on the network eliminating the need for the **frame-relay map** commands.

## *Configuring LAT Compression*

The Local Area Transport (LAT) protocol used by DEC and DEC-compatible terminal servers is one of the common protocols that lacks a well-defined network layer (Level 3), and so must always be bridged.

To reduce the amount of bandwidth LAT traffic consumes on serial interfaces, a LAT-specific form of compression may be specified. This is done with the **bridge-group** *group* **lat-compression** interface subcommand. The command has this syntax:

**bridge-group** *group* **lat-compression**

The argument *group* is the spanning-tree group number.

Compression is applied to LAT frames being sent out the router/bridge through the interface in question.

LAT compression may be specified only for serial interfaces. For the most common LAT operations (user keystrokes and acknowledgment packets), LAT compression reduces LAT's bandwidth requirements by nearly a factor of two.

*Example:*

This command compresses LAT frames on the bridge assigned to group 1.

```
bridge-group 1 lat-compression
```

# *Transparent Bridging Configuration Examples*

This section provides example configurations that you may use to configure your bridging environment.

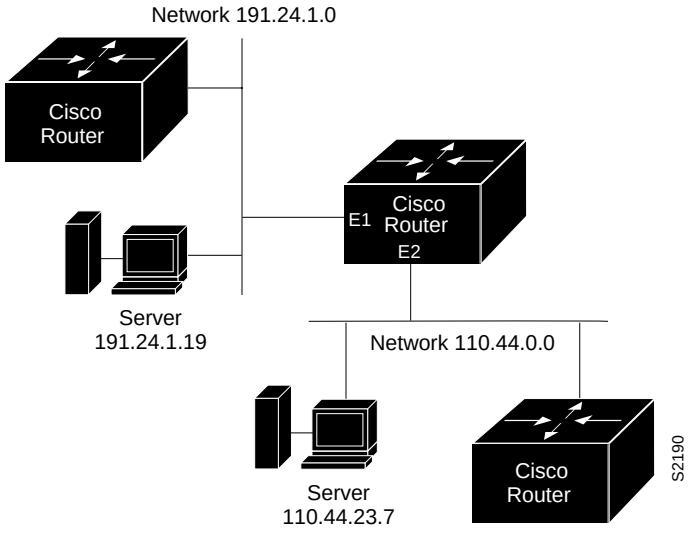
## *Configuring Ethernet Bridging*

In this example, two buildings have networks that must be connected via a T1 link. For the most part, the systems in each building use either IP or DECnet and, therefore, should be routed. There are some systems in each building that must communicate, but they can use only a proprietary protocol.

The example places two Ethernets in each building. One of the Ethernets will be used to attach to the rest of the building network that speaks IP and DECnet. The other Ethernet will be attached to the hosts that use a proprietary protocol. This Ethernet will be enabled for bridging to the serial line and to the other building. Figure 1-5 shows an example configuration.

The interfaces marked with an asterisk (*) are to be configured as part of spanning tree 1. The routers will be configured to route IP and DECnet. This configuration permits hosts on any Ethernet to communicate with hosts on any other Ethernet using IP or DECnet. In addition, hosts on Ethernet 1 in either building can communicate using protocols not supported for routing.

*Figure 1-5* Ethernet Bridging Configuration Example




The configuration file for the router/bridge in Building 1 would be as follows. (Note that no bridging takes place over Ethernet 0. Both IP and DECnet routing are enabled on all interfaces.)

```
decnet address 3.34
interface ethernet 0
ip address 128.88.1.6  255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.1  255.255.255.0
bridge-group  1
decnet cost 10
!
interface ethernet 1
ip address 128.88.3.1  255.255.255.0
bridge-group  1
decnet cost 10
!
bridge 1 protocol dec
```

The configuration file for the router/bridge in Building 2 is similar.

```
decnet address 3.56
!
interface ethernet 0
ip address 128.88.11.9  255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.2  255.255.255.0
bridge-group  1
decnet cost 10
!
interface ethernet 1
ip address 128.88.16.8  255.255.255.0
bridge-group  1
decnet cost 10
```

```
!
bridge 1 protocol dec
```

## Maintaining the Transparent Bridge

Use the **clear bridge** command to remove any learned entries from the forwarding database and zero the transmit and receive counts for any statically configured forwarding entries. The command has this syntax:

**clear bridge** *group*

The argument *group* is the number you chose to specify a particular spanning tree.

## Monitoring the Transparent Bridge

This section describes the EXEC commands you use to obtain displays of activity on the bridged network.

### Viewing Entries in the Forwarding Database

Use the **show bridge** command to view classes of entries in the bridge forwarding database. The command has this syntax:

**show bridge** [*group*] [*interface*]

**show bridge** [*group*] [*address* [*mask*]]

The optional argument *group* is the number you chose that specifies a particular spanning tree.

The optional argument *interface* is a specific interface, such as Ethernet 0.

The optional argument *address* is a 48-bit Ethernet address. This may be entered with an optional mask of bits to be ignored in the address, which is specified with the *mask* argument.

In the sample display, below, the first command would display all entries for hosts reachable via interface Ethernet 0, the second command would display all entries with the vendor code of 0000.0c00.0000, and the third command displays the entry for address 0000.0c00.0e1a. In the fourth command, all entries in the forwarding database would be displayed. In all four examples, the bridge group number has been omitted.

```
show bridge ethernet 0
show bridge 0000.0c00.0000  0000.00FF.FFFF
show bridge 0000.0c00.0e1a
show bridge
```

The following is sample output of the **show bridge** command:

```
Total of 300 station blocks, 295 free
BG Hash   Address        Action  Interface Age  RX count  TX count
 1 00/0   FFFF.FFFF.FFFF discard     -       P         0         0
 1 09/0   0000.0C00.0009 forward Ethernet0   0         2         0
 1 49/0   0000.0C00.4009 forward Ethernet0   0         1         0
 1 CA/0   AA00.0400.06CC forward Ethernet0   0        25         0
```

The first line of the **show bridge** output lists the total number of forwarding database elements in the system and the number in the free list. The total number of forwarding elements is expanded dynamically, as needed. Other field descriptions follow.

*Table 1-2* Forwarding Database Display Field Descriptions

| Field | Description |
|---|---|
| BG | Indicates the bridging group to which the address belongs. |
| Address | Is the MAC address. |
| Action | Is the action to be taken when that address is looked up; choices are to discard or forward the datagram. |
| Interface | Indicates the interface, if any, on which that address was seen. |
| Age | Indicates the number of minutes since a frame was received from or sent to that address. The letter “P” indicates a permanent entry. The letter “S” indicates the system as recorded by the router. On the modular systems, this is typically the broadcast address and the router’s own hardware address; on the IGS, this field will also include certain multicast addresses. |
| RX count | Displays count of the number of frames received from that address. |
| TX count | Displays count the number of frames sent to that address. |

## *Displaying the Known Spanning Tree Topology*

Use the **show span** command to display the spanning-tree topology known to the router/bridge. The display includes whether or not LAT group code filtering is in effect. The command has this syntax:

**show span**

The following is a sample output of the **show span** command. The first part of the display lists global spanning-tree parameters, followed by port-specific parameters.

```
Bridge Group 1 is executing the DEC compatible spanning tree protocol
  Bridge Identifier has priority 127, address 0000.0c00.4369
  Configured hello time 1, max age 15, forward delay 30
  We are the root of the spanning tree
  Acquisition of new addresses is enabled
  Forwarding of multicast source addresses is disabled
  LAT service filtering is disabled
  Topology change flag not set, detected flag not set
  Times:  hold 1, topology change 30, notification 30
          hello 1, max age 15, forward delay 30
  Timers: hello 1, topology change 0, notification 0
 --More--
Port 5 (Ethernet0) of bridge group 1 is forwarding. Path cost 10, pri-
ority 0
   Designated root has priority 127, address 0000.0c00.4369
   Designated bridge has priority 127, address 0000.0c00.4369
   Designated port is 5, path cost 0
   Timers: message age 0, forward delay 0, hold 1
   LAT compression is not set
   Input LAT service deny group code list is not set
   Input LAT service permit group code list is not set
   Output LAT service deny group code list is not set
   Output LAT service permit group code list is not set
   Access list for input filtering on type is 201; for LSAP is not set
   Access list for input address filter is not set
   Access list for output filtering on type is not set; for LSAP is not
set
   Access list for output address filter is not set
 --More--
Port 6 (Serial4) of bridge group 1 is forwarding. Path cost 64, priority
0
   Designated root has priority 127, address 0000.0c00.4369
   Designated bridge has priority 127, address 0000.0c00.4369
   Designated port is 6, path cost 0
   Timers: message age 0, forward delay 0, hold 1
   LAT compression is set
   Input LAT service deny group code list is not set
   Input LAT service permit group code list is not set
   Output LAT service deny group code list is not set
   Output LAT service permit group code list is not set
   Access list for input filtering on type is 202; for LSAP is not set
   Access list for input address filter is not set
   Access list for output filtering on type is 202; for LSAP is not set
   Access list for output address filter is not set
 --More--
```

```
Port 7 (Ethernet1) of bridge group 1 is down. Path cost 10, priority 0
   Designated root has priority 127, address 0000.0c00.4369
   Designated bridge has priority 127, address 0000.0c00.4369
   Designated port is 7, path cost 0
   Timers: message age 0, forward delay 0, hold 1
   LAT compression is not set
   Input LAT service deny group code list is not set
   Input LAT service permit group code list is not set
   Output LAT service deny group code list is not set
   Output LAT service permit group code list is not set
   Access list for input filtering on type is 201; for LSAP is not set
   Access list for input address filter is not set
  Access list for output filtering on type is not set; for LSAP is not
set
   Access list for output address filter is not set
```

## *Debugging the Transparent Bridge*

This section describes the EXEC debugging commands you use to debug the transparent bridge. For each **debug** command, there is a corresponding **undebug** command to disable the reports.

### debug span

Use the **debug span** command to track changes in the spanning-tree topology. This command is useful for verifying correct operation of the spanning-tree protocol.

### debug lat

Use the **debug lat** command on a bridge to show group code filtering actions.

### debug lat-packet

Use the **debug lat-packet** command to list all LAT service advertisements which were forwarded.

## *Transparent Bridging Global Configuration Command Summary*

This section provides a summary of the transparent bridging-specific global configuration commands.

[**no**] **access-list** *list* {**permit**|**deny**} *type-code wild-mask*

Prepares access control information for filtering of frames by protocol type. The argument *list* is a user-selectable number between 200 and 299 that identifies the list. The keyword **permit** permits the frame; the keyword **deny** denies the frame. The argument *type-code* is a 16-bit hexadecimal number written with a leading "0x." The argument *wild-mask* is another 16-bit hexadecimal number whose ones bits correspond to bits in the *type-code* argument that should be ignored when making a comparison.

[**no**] **access-list** *list* {**permit**|**deny**} *address-mask*

Prepares access control information for filtering of frames by Ethernet and IEEE 802.3 address. The argument *list* is an integer from 700 to 799 selected for the list. The argument *address* and *mask* are 48-bit Ethernet addresses written in dotted triplet form. The ones bits in the m*ask* argument are the bits to be ignored in *address*.

[**no**] **bridge** *group* **acquire**

The negative form of this command disables the dynamic learning process and is the default. The argument *group* is the spanning-tree group number.

[**no**] **bridge** *group* **address** *ethernet-address* [**forward**|**discard**] [*interface*]

Adds or removes an address from the forwarding database. The argument *group* is the spanning-tree group number. The argument *ethernet-address i*s a 48-bit dotted triplet hardware address such as those displayed by the EXEC **show arp** command. The argument *ethernet-address* is either a station address, the broadcast address, or a multicast destination address. The optional **forward** keyword enables forwarding of a frame sent from or destined to the specified address. The optional **discard** keyword causes frames sent from or destined to the specified address to be discarded without further processing. The optional argument *interface* specifies an interface after the **discard** or **forward** keyword to indicate the interface on which that address can be reached. Use the **no** for of the command followed by the Ethernet address to remove an address from the forwarding database.

[**no**] **bridge** *group* **domain** *domain-number*

Enables/disables multiple domain spanning trees. Any number of router/bridges can be placed within the domain. The devices in the domain, and only those devices, will then share spanning-tree information. The argument *group* is the bridge group number, and must be a number between 0 and 10, as specified in the **bridge** *group* **protocol ieee** command. The keyword **domain** is required; the argument *domain-number* is a domain number you choose. The default domain number is zero, and this is the required domain number when communicating to IEEE bridges that do not support this domain extension.

*Note:* This command works only when the bridge group is running the IEEE spanning-tree protocol. Non-Cisco bridges may not work correctly on networks containing Cisco bridges with nonzero domain numbers.

[**no**] **bridge** *group* **forward-time** *seconds*

Sets or returns to the default the forward delay interval, or the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is any value between ten and 200 seconds. The default value is 30 seconds.

[**no**] **bridge** *group* **hello-time** *seconds*

Specifies or returns to the default the interval between HELLO Bridge Protocol Data Units (BPDUs). The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is any value between one and ten seconds. The default value is one second.

[**no**] **bridge** *group* **lat-service-filtering**

Enables or disables LAT service filtering. The argument *group* specifies the bridge group. The default is **no** LAT service filtering**.**

[**no**] **bridge** *group* **max-age** *seconds*

Specifies or removes the interval in which the spanning-tree topology is recomputed when a bridge does not hear BPDUs from the root bridge. The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is the interval the bridge will wait to hear BPDUs from the root bridge. The default interval is 15 seconds.

[**no**] **bridge** *group* **multicast-source**

Allows or disallows the forwarding, but not the learning, of multicast source addresses. The argument *group* is the group number assigned to the spanning tree.

[**no**] **bridge** *group* **priority** *number*

Sets the priority of an individual bridge for selection as the root bridge. The argument *group* is the group number assigned to the spanning tree. The argument *number* can range from 1 to 65,000. The default priority value is 128. A lower number increases the likelihood for selection as the root bridge.

[**no**] **bridge** *group* **protocol** {**dec**|**ieee**}

Defines or removes a spanning-tree protocol and spanning-tree group. The argument *group is* a number between one and nine that refers to a particular spanning tree. The keyword **protocol** specifies the protocol to use, either **ieee** or **dec.**

## *Transparent Bridging Interface Subcommand Summary*

This section provides an alphabetical summary of the bridging-specific interface subcommands.

[**no**] **bridge-group** *group*

Assigns or removes the network interface to or from the spanning-tree group. The argument *group* is the group number assigned to the spanning tree.

[**no**] **bridge-group** *group* **circuit** *number*

Establishes or removes load balancing. The command marks a serial interface as belonging to circuit group number. The argument *group* is the spanning-tree group number.

The argument *number* defines the circuit group number and is a small integer less than ten. Parallel serial interfaces on both bridges must all be flagged as being members of the same circuit group.

[**no**] **bridge-group** *group* **input-address-list** *list*

Assigns or removes an access list to a particular interface for filtering the Ethernet or IEEE 802.3 source addresses. The argument *group* is the spanning-tree group number. The argument *list* is an access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[**no**] **bridge-group** *group* **output-address-list** *list*

Assigns or removes an access list to a particular interface for filtering the Ethernet or IEEE 802.3 destination addresses. The argument *group* is the spanning-tree group number. The argument *list* is an access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[**no**] **bridge-group** *number* **input-lat-service-deny** *grouplist*

When enabled, causes the system to not bridge any LAT service advertisement which match the group list specified on input. The argument *grouplist* lists the LAT groups. The argument *number* is the previously chosen spanning-tree group number. Default is

no filtering.

[**no**] **bridge-group** *number* **input-lat-service-permit** *grouplist*

When enabled, causes the system to bridge only those LAT service advertisements which match the group list specified on input. The argument *grouplist* lists the LAT group codes. The argument *number* is the previously chosen spanning-tree group number. Default is no filtering.

[**no**] **bridge-group** *number* **output-lat-service-deny** *grouplist*

When enabled, causes the system to not bridge onto this output interface any LAT service advertisements that match any group in the argument *grouplist*. The argument *number* is the previously chosen spanning-tree group number. Default is no filtering.

[**no**] **bridge-group** *number* **output-lat-service-permit** *grouplist*

When enabled, causes the system to bridge onto this output interface only those service advertisements that match any group in the argument *grouplist*. The argument *number* is the previously chosen spanning-tree group number. If a message matches both a deny and a permit, the message will not be bridged. The EXEC **show span** command reports the group code filtering in effect. Default is no filtering.

[**no**] **bridge-group** *group* **input-lsap-list** *list*

Adds or removes a filter for IEEE 802.3-encapsulated packets on input. This access list is applied to all IEEE 802.3 frames received on that interface prior to the bridge-learning process. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[**no**] **bridge-group** *group* **output-lsap-list** *list*

Adds or removes a filter for IEEE 802.3-encapsulated packets on output. This access list is applied just before sending out a frame to an interface. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 200, which you assigned with the bridge **access-list** command.

[**no**] **bridge-group** *group* **input-type-list** *list*

Adds or removes a filter for Ethernet- and SNAP-encapsulated packets on input. The access list is then applied to all Ethernet frames received on that interface prior to the bridge learning process. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[**no**] **bridge-group** *group* **output-type-list** *list*

Adds or removes a filter for Ethernet- and SNAP-encapsulated packets on output. This access list is then applied just before sending out a frame to an interface. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299, which you assigned with the bridge **access-list** command.

[**no**] **bridge-group** *group* **lat-compression**

Reduces the amount of bandwidth LAT traffic consumes on serial interfaces. The argument *group* is the spanning-tree group number. Compression is applied to LAT frames being sent out the router/bridge through the interface in question. LAT compression may be specified only for serial interfaces. For the most common LAT operations (user keystrokes and acknowledgment packets), LAT compression reduces LAT's bandwidth requirements by nearly a factor of two.

[**no**] **bridge-group** *group* **path-cost** *cost*

Sets or removes a different path cost. The path cost can range from 0 to 65,535, with higher values indicating higher costs. The argument *group* is the spanning-tree group number. The argument *cost* is the path cost. The default path cost is 100.

Assigns a priority to an interface. This priority is used in tie-breaking when computing a network topology. The argument *group* is the spanning-tree group number. The argument *number* can range from 0 to 255. The default value is zero, and the lower the number, the more likely it is that the bridge on this interface will be chosen as the root.