Chapter 22
Configuring Source-Route Bridging

22

This chapter describes routing and bridging in Token Ring environments. This chapter discusses the following topics:

- The fundamental concepts of local and remote source routing, and how source-route bridges interact with Level 3 routers
- The IEEE 802.5 Token Ring frame format and the routing information field (RIF)
- Enabling use of the RIF for source-route bridging
- Support for Local Acknowledgment of Token Ring-based LLC2 sessions
- Support for bridging between source-route bridge groups and transparent bridge groups with Source-Route Translational Bridging (SR/TLB)
- Filtering datagrams by protocol type, vendor code, and configuring NetBIOS access control filters
- Using Access Expressions to control filtering through programming
- Managing source-route bridges with LAN Network Manager

This chapter also provides source-route bridging configuration examples, and describes how to maintain, monitor, and debug your source-route bridges. Command summaries are included at the end of the chapter.

Source-Route Bridging Overview

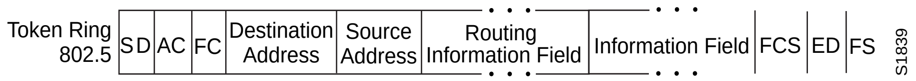
Cisco bridging software includes source-route bridging capability. This ability allows the Cisco router/bridge to simultaneously act as a Level 3 router and a Level 2 source-route bridge. This allows protocols such as Novell or XNS to be routed on Token Rings, while other protocols such as SNA or NetBIOS are source-route bridged.

Source-route bridging technology is a combination of bridging and routing functions. A source-route bridge is allowed to make routing decisions based upon the contents of the Medium Access Control (MAC) frame header. Keeping the routing function at the MAC or Level 2 layer allows the higher-layer protocols to execute their tasks more efficiently, and also allows the LAN to be expanded without the knowledge of the higher-layer protocols.

As designed by IBM and the IEEE 802.5 committee, source-route bridges connect extended Token Ring LANs. A source-route bridge uses the Routing Information Field (RIF) in the IEEE 802.5 MAC header of a datagram (see Figure 1-1) to determine which rings, or Token Ring network segments, the packet must transit. The source station inserts the RIF into the MAC header immediately following the source address field in every frame, giving this style of bridging its name. The destination station reverses the routing field to reach the originating station.

The information in a RIF is derived from explorer packets generated by the source node. These explorer packets traverse the entire source-route bridge network, gathering information on the possible paths the source node might use.

Figure 1-1 IEEE 802.5 Token Ring Frame Format



Unlike transparent spanning-tree bridging, which requires time to recompute topology in the event of failures, source-route bridging allows multiple, active paths through the network, which provides for more timely switches to alternate routes in the event of failure. Most importantly, source-route bridging places the burden of transmitting frames with the end stations by allowing them to determine the routes the frames take.

Cisco's Implementation of Source-Route Bridging

Cisco's source-route bridging software implementation provides these features:

- Provides configurable fast-switching software for source-route bridging.
- Provides for a local source-route bridge that connects two or more Token Ring networks.
- Provides *ring groups* to configure a source-route bridge with more than two network interfaces. A ring group is a collection of Token Ring interfaces in one or more Cisco routers that are collectively treated as a *virtual ring*.
- Provides explorer packets to collect RIF information. An *all-rings* explorer packet follows all possible paths to a destination ring. *Spanning* explorer packets follow a statically configured spanning tree when looking for paths.
- Provides for remote source-route bridges involving multiple router/bridges separated by non-Token Ring segments; one way is to encapsulate the Token Ring traffic inside IP datagrams passed over a TCP connection between two Cisco router/bridges; another way is to use MAC layer encapsulations over a single serial line, Ethernet, Token Ring, or FDDI ring connected between two routers attached to Token Ring networks.
- Provides for configurable limits to the size of the TCP backup queue.

- Provides a dynamically determined RIF cache based on the protocol. It also allows you to manually add entries to the RIF cache.
- Provides for filtering of NetBIOS frames. The frames can be filtered by station name or by a packet byte offset.
- Provides for filtering by MAC address, LSAP header, and protocol type.
- Provides for translation into transparently bridged frames to allow source-route stations to communicate with non-source-route stations (typically on Ethernet).

Configuring Source-Route Bridging

To configure source-route bridging on your Cisco router/bridge, follow these steps:

Step 1: Configure the Token Ring interface for source-route bridging. The Cisco router/bridge supports both local and remote source-route bridging.

To determine if your Token Ring interface has the proper hardware and firmware support for source-route bridging, examine the output of the EXEC **show interface** command for that interface. If the output has a line that reads “Source Route Bridge capable” or “Source Route Transparent Bridge capable,” then you may proceed with configuration. Otherwise, you need to contact Cisco Systems for a hardware and firmware field upgrade.

Step 2: Define the types of explorer packets to use: either spanning tree or the default all-rings explorer packets.

Step 3: Choose the remote peers with which you want to establish locally terminated LLC2 sessions using the Local Acknowledgment functionality.

Step 4: Choose the transparent bridge group, if any, to tie into your source-route bridged network using the Source-Route Translational Bridging (SR/TLB) functionality.

Step 5: Choose access controls and filters as needed to administratively control the traffic traversing the Cisco source-route bridge. Use access expressions, if needed, to provide greater control over the use of the filters.

The source-route bridging software supports filtering of frames. Filtering can be done by protocol type or by vendor code. You can also configure access control filters for packets transmitted across a Token Ring bridge using the NetBIOS interface. Additionally, EX-EC-level commands for monitoring and debugging the bridge are also available. These tasks and commands are described in the following sections.

Enabling and Disabling the Source-Route Fast-Switching Cache

By default, fast-switching software is enabled in the source-route bridging software. Fast switching allows for faster implementations of local source-route bridging between 4/16 Megabit Token Ring cards in the same Cisco router/bridge. This feature also allows for faster implementations of local source-route bridging between two Cisco router/bridges using the 4/16 Megabit Token Ring cards and the direct interface encapsulation. To disable fast switching, use the **no source-bridge route-cache** interface subcommand.

Note: Using either NetBIOS Byte Offset access lists or the access-expression capability to logically combine the access filters disables the fast switching of SRB frames.

The full syntax of this command follows:

source-bridge route-cache
no source-bridge route-cache

Example:

These commands disable use of fast switching between two Token Ring interfaces.

```
interface token 0
source-bridge 1 1 2
no source-bridge route-cache
!
interface token 1
source-bridge 2 1 1
no source-bridge route-cache
```

Determining the Source-Route Information Field

This section explains how to build routing information fields (RIFs). A RIF is built up of ring and bridge numbers. A *ring* is a single Token Ring network segment. Each ring in the extended Token Ring network is designated by a unique 12-bit ring number. Each bridge between two Token Rings is designated by a unique 4-bit bridge number. Bridge numbers must be unique only between bridges that connect the same two Token Rings.

Figure 1-2 illustrates the basic format for the Routing Information Field.

Figure 1-2 Basic RIF Format

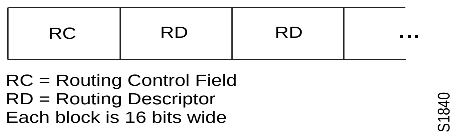
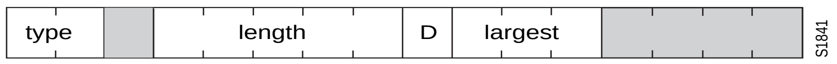


Figure 1-3 illustrates the routing control format for the RIF. Descriptions of each field follow.

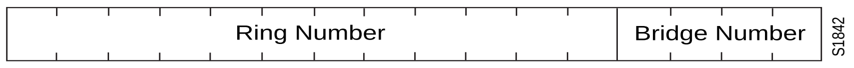
Figure 1-3 RIF Routing Control Format



- Shaded fields are reserved.
- type—RIF type, as follows:
  - 00: Specific route
  - 10: All rings, all routes
  - 11: All rings, spanning routes (limited broadcast)
- length—Total length in bytes of the RIF.
- D—Direction, indicated as follows:
  - 0: Interpret route left to right (forward)
  - 1: Interpret route right to left (reverse)
- largest—Largest frame that can be handled by this route, as follows:
  - 000: 516 bytes (DDN 1822)
  - 001: 1500 bytes (Ethernet)
  - 010: 2052 bytes
  - 011: 4472 bytes (Token Ring at 4 Mb speed)
  - 100: 8144 bytes (Token Bus and Token Ring at 16 Mb maximum)
  - 101: 11407 bytes
  - 110: 17800 bytes
  - 111: 65535 bytes (initial values)

Figure 1-4 describes the routing descriptor format of the RIF string. Definitions of each field follow the figure.

Figure 1-4 Routing Descriptor Format



- Ring Number—Unique hexadecimal ring number within the bridged network.

- Bridge Number—Unique hexadecimal bridge number between any bridges connecting the same two rings.

The following sections describe how to enable and configure static RIF entries.

Enabling Use of the RIF

Level 3 routers that use protocol-specific information (for example, Novell IPX or XNS headers) rather than MAC information to route datagrams must also be able to collect and use RIF information to ensure that the Level 3 routers can transmit datagrams across a source-route bridge. The Cisco software default is to not collect and use RIF information for routed protocols. This allows operation with software that does not understand or properly use RIF information, such as versions of Novell NetWare prior to Version 2.15c.

To enable collection and use of RIF information, use the **multiring** interface subcommand. The full command syntax follows:

multiring {*protocol-keyword* | **all** | **other**}
no multiring {*protocol-keyword* | **all** | **other**}

The current software allows you to specify a protocol. This is specified by the argument *protocol-keyword*. The protocols supported and the keyword you enter follow:

- **apollo**—Apollo Domain
- **appletalk**—AppleTalk Phase 1 and 2
- **clns**—ISO CLNS
- **decnet**—DECnet Phase IV
- **ip**—IP
- **novell**—Novell IPX
- **vines**—Banyan VINES
- **xns**—XNS

There are also two special keywords with the **multiring** command. The keyword **all** enables the multiring for *all* frames. The keyword **other** enables the multiring for *any* routed frame not included in the previous list of supported protocols.

The **multiring** command was extended in software release 8.3 to allow for per-protocol specification of the interface's ability to append RIFs to routed protocols. When it is enabled for a protocol, the router will source packets that include information used by source-route bridges. This allows a Cisco router with Token Ring interfaces, for the protocol or protocols specified, to connect to a source-bridged Token Ring network. If a protocol is not specified for multiring, the Cisco router can only route packets to nodes directly connected to its local Token Ring.

Note: Previous to Software Release 8.3, the **multiring** command enabled multiring protocols, in particular, the use of explorers and RIFs, for *all* routable protocols. This sometimes caused problems when multiring-capable devices speaking one particular protocol were attached to the same ring as a nonmultiring-capable device speaking a different network protocol. If the earlier **multiring** command (pre-8.3 release) was not specified, nodes speaking one particular protocol would be able to communicate through the Cisco router, but nodes speaking other protocols could not. The reverse was true when the multiring capability was specified on the interface.

Note: In 8.3 or later releases of the software, the command **multiring all** is equivalent to the 8.2 and earlier versions of the **multiring** command.

The **no multiring** subcommand with the appropriate keyword disables the use of RIF information for the protocol specified.

Example:

These commands enable a Token Ring interface for the IP and Novell IPX protocols. RIFs will be generated for IP frames, but not for the Novell IPX frames.

```
interface tokenring 0
multiring ip
ip address 131.108.183.37  255.255.255.0
novell network 33
```

Determining the RIF Timeout Interval

RIF information is maintained in a cache whose entries are aged. The global configuration command **rif timeout** determines the number of minutes an inactive RIF entry is kept. The full command syntax follows:

rif timeout *minutes*
no rif timeout

The default interval is 15 minutes. The minimum value is one minute. Assign a new interval value using the *minutes* argument.

The **no rif timeout** command restores the default.

The EXEC command **show rif** displays the contents of the RIF cache. The EXEC command **clear rif-cache** clears the contents of RIF cache. See the sections “Maintaining the Source-Route Bridge” and “Monitoring the Source-Route Bridge” later in this chapter for more information about these commands.

Example:

This command changes the timeout period to five minutes.

```
rif timeout 5
```

Configuring a Static RIF Entry

If a Token Ring host does not support the use of IEEE 802.2 TEST or XID datagrams as explorer packets, you may need to add static information to the RIF cache of the router/bridge.

To enter static source-route information into the RIF cache, use the following variation of the **rif** global configuration command:

> **rif** *MAC-address* [*RIF-string*] [*interface-name* | **ring-group** *ring*]
> **no rif** *MAC-address* [*interface-name* | **ring-group** *ring*]

The argument *MAC-address* is a 12-digit hexadecimal string written as a dotted triple, for example 0010.0a00.20a6.

Using the command **rif** *MAC-address* without any of the optional arguments puts an entry into the RIF cache indicating that packets for this MAC address should not have RIF information.

The command **no rif** *MAC-address* removes an entry from the cache.

The optional argument *RIF-string* is a series of 4-digit hexadecimal numbers separated by a dot (.). This RIF string is inserted into the packets sent to the specified MAC address.

An interface name (for example, tokenring0) can be specified with the optional *interface-name* argument, to indicate the origin of the RIF.

A ring group number (specified with the **source-bridge ring-group** global configuration command) may also be specified with the **ring-group** keyword and *ring* argument, to indicate the origin of the RIF. Ring groups are explained in the section "Configuring Ring Groups and Multiport Source-Route Bridges."

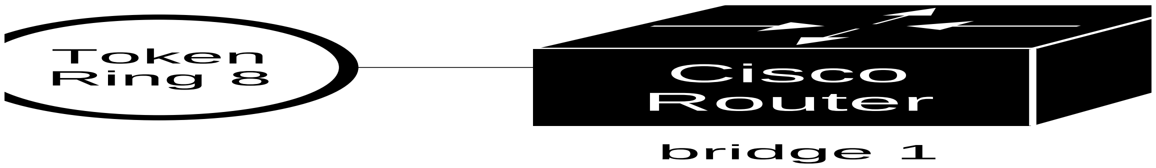
Do not configure a static RIF with any of the *all rings* type codes. Doing so causes traffic for the configured host to appear on more than one ring and leads to unnecessary congestion. The format of a RIF string is illustrated in Figure 1-2, Figure 1-3, and Figure 1-4.

Note: Input to the **source-bridge** configuration subcommand is in decimal format. RIF displays and input are in hexadecimal format, and IBM source-route bridges use hexadecimal for input. It is essential that bridge and ring numbers are consistent for proper network operation. This means you must explicitly declare the numbers to be hexadecimal by preceding the number with 0x, or you must convert IBM hexadecimal numbers to a decimal equivalent when entering these numbers. As an example, IBM hexadecimal bridge number 10 would be entered as hexadecimal number 0x10 or decimal number 16 in the Cisco configuration commands. In the displays, these commands will always be in decimal.

Example:

In this example configuration, the path between rings 8 and 9 connected via source-route bridge 1 is described by the route descriptor 0081.0090. A full RIF, including the route control field, would be 0630.0081.0090. The static RIF entry would be submitted to the leftmost router as shown in Figure 1-5.

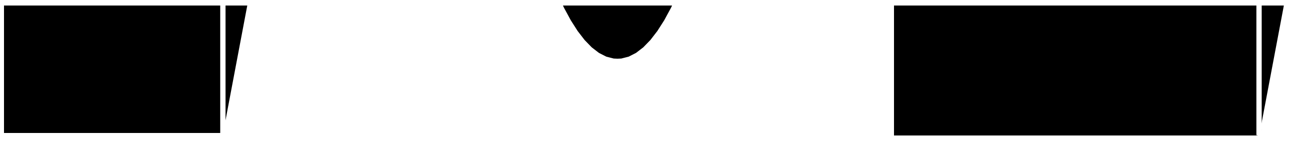
Figure 1-5 Assigning a RIF to a Source-Route Bridge



```
    rif 1000.5A12.3456 0630.0081.0090
```

As another example, assume a datagram was sent from a Cisco router/bridge on ring 21 (15 hexadecimal), across bridge 5 to ring 256 (100 hexadecimal), and then across bridge 10 (A hexadecimal) to ring 1365 (555 hexadecimal) for delivery to a destination host on that ring. See Figure 1-6.

Figure 1-6 Assigning a RIF to a Two-Hop Path



The RIF in the leftmost router describing this two-hop path is 0830.0155.100a.5550, and is entered as follows:

```
    rif 1000.5A01.0203 0830.0155.100a.5550
```

Configuring Local Source-Route Bridging

This section describes how to configure the Cisco router/bridge as a local source-route bridge. A local source-route bridge directly connects two or more Token Ring networks. Bridged traffic does not pass across non-Token Ring media.

When acting as a source-route bridge, only those protocols that are not being routed are source-route bridged. For example, if Novell routing is enabled on the Cisco router/bridge, Novell datagrams will not be source-bridged. Datagrams for other nonrouted protocols will be source-bridged, however.

Enabling Local Source-Route Bridging

To configure an interface for local source-route bridging, use the **source-bridge** interface subcommand as follows:

> **source-bridge** *local-ring bridge-number target-ring*
> **no source-bridge**

The argument *local-ring* is the ring number for this interface's Token Ring. A ring number is a decimal number between 1 and 4095 that uniquely identifies a network segment or ring within the bridged Token Ring network.

The argument *bridge-number* is a decimal number between 1 and 15 that uniquely identifies a bridge connecting the two rings.

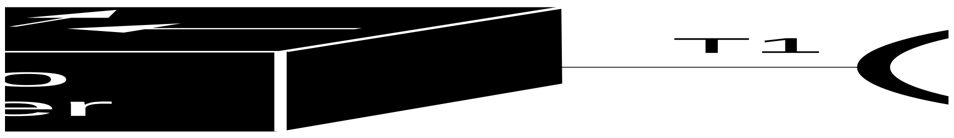
The argument *target-ring* is the decimal ring number of the destination ring on this router/bridge. It must also be unique within the bridged Token Ring network.

The **no source-bridge** command disables source bridging on a particular interface.

Example:

Figure 1-7 illustrates a simple two-port bridge configuration. The sample text following this figure provides the configuration commands.

Figure 1-7 Dual Port Source-Route Bridge Configuration



```
interface token ring 0
source-bridge 129 1 130
!
interface token ring 1
source-bridge 130 1 129
```

Token Rings 129 and 130 are connected via the Cisco router/bridge.

Configuring Explorer Packets

There are two types of explorer packets used to collect RIF information:

- All-rings, all-routes explorer packets follow all possible paths to a destination ring. In a worst case scenario, the number of all-rings explorers generated may be exponentially large.
- Spanning or limited-route explorer packets follow a spanning tree when looking for paths, greatly reducing the number of explorer packets required. There is currently no dynamic spanning-tree algorithm to establish that spanning tree; it must be manually configured.

Enabling Spanning Explorers

Use the **source-bridge spanning** interface subcommand to enable use of spanning explorers. The full command syntax follows:

> **source-bridge spanning**
> **no source-bridge spanning**

The command puts the interface into a forwarding or active state with respect to the spanning tree.

The **no source-bridge spanning** command disables use of spanning explorers. Only spanning explorers will be blocked; everything else will be forwarded. Use of the **source-bridge spanning** command is recommended.

Limiting the Number of Source-Route Bridge Hops

If you wish to limit the maximum number of source-route bridge hops of your network, use the **source-bridge max-hops** interface subcommand. The full command syntax follows:

> **source-bridge max-hops** *count*
> **no source-bridge max-hops**

The argument *count* determines the number of bridges an explorer packet may traverse. Typically, the maximum number of bridges for interoperability with IBM equipment is seven (eight rings and seven bridges).

The command **no source-bridge max-hops** resets the count back to the maximum value.

Example:

The following example builds on the dual-port, source-route bridge configuration seen in Figure 1-7. The example routes IP and source-route bridges all other protocols. Spanning explorers are used.

```
interface tokenring 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface tokenring 1
ip address 131.108.130.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
multiring all
```

The **multiring** subcommand causes the IP routing software to use RIFs, as necessary.

Configuring Ring Groups and Multiport Source-Route Bridges

To configure a source-route bridge with more than two network interfaces, Cisco uses the concept of a *ring group*. A ring group is a collection of Token Ring interfaces in one or more Cisco routers that are collectively treated as a virtual ring. The ring group is denoted by a ring number that must be unique for the network. The ring group's number is used just like a physical ring number, showing up in any route descriptors contained in packets being bridged.

A ring group is defined or removed with the **source-bridge ring-group** global configuration command. The full command syntax follows:

source-bridge ring-group *ring-number*
no source-bridge ring-group *ring-number*

To configure a specific interface as part of a ring group, its target ring number parameter is set to the ring group number specified in this command. You should not use the number 0, as this is reserved to represent the local ring.

Example:

Figure 1-8 shows an example configuration of a four-port Token Ring source-route bridge.

Figure 1-8 Four-Port Source-Route Bridge

Rings 1000, 1001, 1002, and 1003 are all source-route bridged to each other across ring group 7.

```
source-bridge ring-group 7
!
interface tokenring 0
source-bridge 1000 1 7
source-bridge spanning
!
interface tokenring 1
source-bridge 1001 1 7
source-bridge spanning
!
interface tokenring 2
```

```
source-bridge 1002 1 7
source-bridge spanning
!
interface tokenring 3
source-bridge 1003 1 7
source-bridge spanning
```

Configuring Remote Source-Route Bridging

The previous section discussed local source-route bridging, or bridging between Token Rings connected by the same router/bridge. This section describes how to configure remote source-route bridges involving multiple router/bridges separated by non-Token Ring network segments. The following sections assume you are familiar with configuring a Cisco local source-route bridge.

There are two ways to set up remote source-route bridging. You can encapsulate the source-route bridged traffic inside IP datagrams passed over a TCP connection between two router/bridges. TCP is used to ensure the reliable and ordered delivery of source-route-bridged traffic. TCP has the following advantages:

- Token Ring networks may be connected across arbitrary media including Ethernets, FDDI, serial interfaces, X.25 networks, and so forth.
- A multiprotocol backbone network may be used. There is no need to dedicate a special wide area network to carrying Token Ring traffic.
- If the IP network is engineered properly, the source-route traffic can take advantage of multiple redundant paths. Cisco multiprotocol routers can load share over the redundant paths. Also, if a path fails, there is no need for hosts to retransmit explorer packets. The IP routing handles the network reconfiguration transparently to the Token Ring hosts.

You can also set up a remote source-route bridge to use only a MAC layer encapsulation to pass frames over a single physical network connection between two routers attached to Token Rings. Use this method when you are running source-route bridge traffic over a slow serial line (56 kpbs or less). You do not have the flexibility of the TCP approach, but you do have better performance as there is less overhead.

Configuring Remote Source-Route Bridging with TCP

To configure a remote source-route bridge to use TCP, follow these steps:

Step 1: Define a ring group. Every Cisco router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. These other router/bridges are referred to as *peers*.

Step 2: List your peers with multiple uses of the **source-bridge remote-peer** command. You must include one of your own IP addresses in that list. Each peer should appear only once in this list, not one time for each Token Ring present. All peers should have the same list of peers. Listing the peer bridges is described in greater detail below.

Step 3: Configure the Token Ring interfaces for source-route bridging. The value of the target ring parameter for the **source-bridge** command should be the ring group number.

Listing the Peer Bridges

The **source-bridge remote-peer** global configuration command has the following syntax when using TCP:

> **source-bridge remote-peer** *ring-group* **tcp** *ip-address* [**lf** *size*][**local-ack**] [**version** *number*]
> **no source-bridge remote-peer** *ring-group* **tcp** *ip-address*

Use this command to identify the IP address of a peer in the ring group with which to exchange source-bridge traffic using TCP.

The keyword **lf** specifies the maximum size frame to be sent to this remote peer. The router negotiates all transit routes down to this size or lower. Use this argument to prevent timeouts in end hosts by reducing the amount of data they have to transmit in a fixed interval. For example, in some networks containing slow links, it would be impossible to transmit an 8K frame and receive a response within a few seconds. These are fairly standard defaults for an application on a 16 Mb Token Ring. If the frame size is lowered to 516 bytes, then only 516 bytes must be transmitted and a response received in 2 seconds. This is a much easier accomplishment in a network with slow links. The legal values for this argument are 516, 1500, 2052, 4472, 8144, 11407, and 17800 bytes.

The keyword **local-ack** specifies that Local Acknowledgment should be used for LLC2 sessions going to this remote peer. Refer to the section “The Cisco Local Acknowledgment Function” in this chapter for a discussion of Local Acknowledgment.

The keyword **version** specifies the forced RSRB protocol version number for the remote peer.

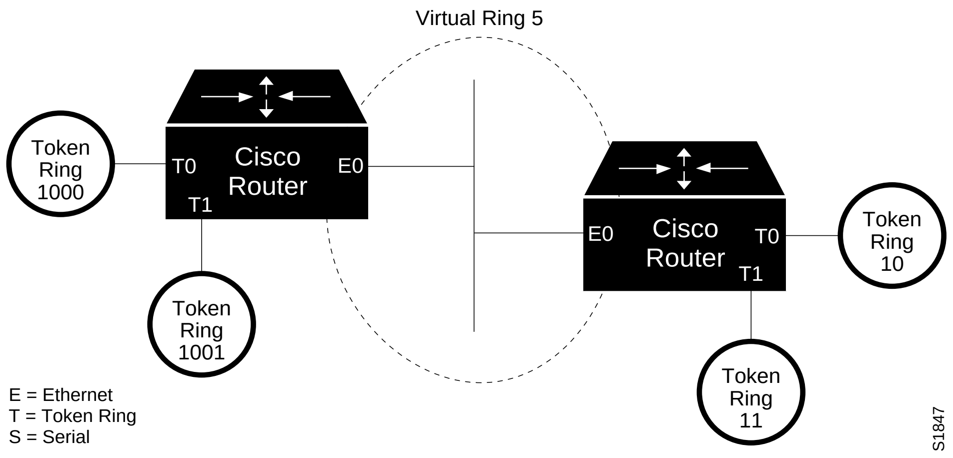
Note: Release 8.2 and earlier implementations of RSRB spoke RSRB Protocol Version 1. Release 8.3 and later implementations of RSRB speak RSRB Protocol Version 2. To provide for backward compatibility, Releases 8.3 and later releases attempt first to form a connection with each remote peer using Version 2 of the RSRB Protocol. If that attempt fails for any reason, then a Version 1 connection is attempted. This regression back to protocol Version 1 can even happen between two peers that are both running Release 8.3 or later if a temporary network problem existed when the Version 2 connection had been attempted.

This behavior is required to provide backward compatibility to Releases 8.2 and earlier. However, beginning with 9.0, certain RSRB features, such as Local Acknowledgment, SDLLC, and Lan Network Manager, require the use of the Version 2 protocol. In these instances, when you know that both peers will be running Release 9.0 and later, it is best to specify that the Version 2 protocol will always be used.

Example:

The following example illustrates a configuration of two router/bridges configured for remote source-route bridging using TCP as a transport. Each router has two Token Rings. They are connected together by an Ethernet segment over which the source-route bridged traffic will pass. The first router configuration is a source-route bridge at address 131.108.2.29. These peers are both running Release 9.0, so the **version** keyword has been specified on the **remote-peer** statements. Figure 22-9 illustrates this network:

Figure 1-9 Remote Source-Route Bridging Using TCP as a Transport



The configuration for the source-route bridge at address 131.108.2.29 as depicted in Figure 1-9 would be as follows:

```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 131.108.2.29 version 2
source-bridge remote-peer 5 tcp 131.108.1.27 version 2
!
interface ethernet 0
ip address 131.108.4.4 255.255.255.0
```

```
!
interface tokenring 0
ip address 131.108.2.29 255.255.255.0
source-bridge 1000 1 5
source-bridge spanning
!
interface tokenring 1
ip address 131.108.128.1 255.255.255.0
source-bridge 1001 1 5
source-bridge spanning
```

The configuration of the source-route bridge at 131.108.1.27 is:

```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 131.108.2.29
source-bridge remote-peer 5 tcp 131.108.1.27
!
interface ethernet 0
ip address 131.108.4.5 255.255.255.0
!
interface tokenring 0
ip address 131.108.1.27 255.255.255.0
source-bridge 10 1 5
source-bridge spanning
!
interface tokenring 1
ip address 131.108.131.1 255.255.255.0
source-bridge 11 1 5
source-bridge spanning
```

Limiting the Size of the Backup Queue

You can limit the size of the backup queue for remote source-route bridging to control the number of packets that can wait for transmission to a remote ring before packets start being thrown away. Use the **source-bridge tcp-queue-max** global command to do this. The full syntax for this command follows.

source-bridge tcp-queue-max *number*
no source-bridge tcp-queue-max

The argument *number* is the number of packets to hold in any single outgoing TCP queue to a remote Cisco router. The default value is 100. Enter the **no source-bridge tcp-queue-max** command to return to the default value.

Example:

If, for example, your network experiences temporary bursts of traffic using the default packet queue length, the following command raises the limit from 100 to 150 packets.

```
source-bridge tcp-queue-max 150
```

Configuring Remote Source-Route Bridging with Direct Encapsulation

To configure a remote source-route bridge to use a point-to-point serial line or a single Ethernet, FDDI, or Token Ring hop, follow these steps:

Step 1: Define a ring group. Every Cisco router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. These other router/bridges are referred to as peers.

Step 2: List the interfaces over which you will be sending source-route bridged traffic with the **source-bridge remote-peer** command. The interfaces must be serial, Ethernet, FDDI, or Token Ring. On a serial interface, you must use the HDLC encapsulation.

Step 3: Configure the Token Ring interfaces for source-route bridging. The value of the target ring parameter for the **source-bridge** commands should be the ring group number.

The **source-bridge remote-peer** global configuration command has the following syntax when used to specify a point-to-point direct encapsulation connection:

source-bridge remote-peer *ring-group* **interface** *interface-name* [*mac-address*] [**lf** *size*] [**version** *number*]
no source-bridge remote-peer *ring-group* **interface** *interface-name*

Use this command to identify the interface over which to send source-route bridged traffic to another Cisco router/bridge in the ring group. A serial interface does not require that you include a MAC-level address; all other types of interfaces do require that you supply a MAC address. To find the MAC address, type the command **show interface** on the remote router, as in the following example output. The MAC address in this example is 5500.2000.dc27:

```
router#show interface
TokenRing 0 is up, line protocol is up
  Hardware is 16/4 Token Ring, address is 5500.2000.dc27 (bia
0000.3000.072b)
  Internet address is 150.136.230.203, subnet mask is 255.255.255.0
  MTU 8136 bytes, BW 16000 Kbit, DLY 630 usec, rely 255/255, load 1/255
  Encapsulation SNAP, loopback not set, keepalive set (10 sec)
  ARP type: SNAP, ARP Timeout 4:00:00
  Ring speed: 16 Mbps
  Single ring node, Source Route Bridge capable
  Group Address: 0x00000000, Functional Address: 0x60840000
  Last input 0:00:01, output 0:00:01, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     16339 packets input, 1496515 bytes, 0 no buffer
     Received 9895 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     32648 packets output, 9738303 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets, 0 restarts
     5 transitions
```

The keyword **lf** specifies the maximum size frame to be sent to this remote peer. The router negotiates all transit routes down to this size or lower. This argument is useful in preventing timeouts in end hosts, by reducing the amount of data they have to transmit in a fixed interval. For example, in some networks containing slow links, it would be impossible to transmit an 8K frame and receive a response within a few seconds. These are fairly standard defaults for an application on a 16 Mb Token Ring. If the frame size is lowered to 516 bytes, then only 516 bytes must be transmitted and a response received in 2 seconds. This is a much easier accomplishment in a network with slow links. The legal values for this argument are 516, 1500, 2052, 4472, 8144, 11407, and 17800 bytes.

The keyword **version** specifies the forced RSRB protocol version number for the remote peer.

Note: Release 8.2 and earlier implementations of RSRB spoke RSRB Protocol Version 1. Release 8.3 and later implementations of RSRB speak RSRB Protocol Version 2. To provide for backward compatibility, Releases 8.3 and later releases attempt first to form a connection with each remote peer using Version 2 of the RSRB Protocol. If that attempt fails for any reason, then a Version 1 connection is attempted. This regression back to protocol Version 1 can even happen between two peers that are both running Release 8.3 or later if a temporary network problem existed when the Version 2 connection had been attempted.

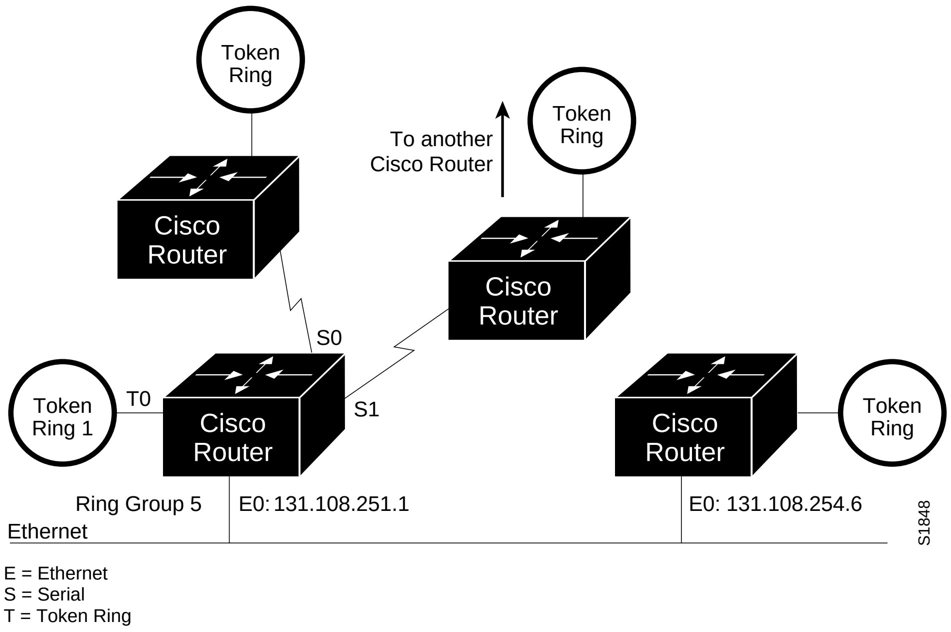
This behavior is required to provide backward compatibility to Releases 8.2 and earlier. However, beginning with 9.0, certain RSRB features, such as Local Acknowledgment, SDLLC, and Lan Network Manager, require the use of the Version 2 protocol. In these instances, when you know that both peers will be running Release 9.0 and later, it is best to specify that the Version 2 protocol will always be used.

Note: It is possible to mix both direct and TCP transport methods within the same ring group.

Example:

Figure 1-10 shows an example configuration of a router/bridge configured for remote source-route bridging using both the direct and TCP transport methods.

Figure 1-10 Remote Source-Route Bridge Using Both Direct and TCP Transport Methods



The configuration for the network as depicted in Figure 1-10 would be as follows:

```
source-bridge ring-group 5
source-bridge remote-peer 5 interface serial0
source-bridge remote-peer 5 interface Ethernet0 0000.0c00.1234
source-bridge remote-peer 5 tcp 131.108.254.6
source-bridge remote-peer 5 tcp 131.108.251.1
!
interface tokenring 0
source-bridge 1 1 5
source-bridge spanning
!
interface ethernet 0
ip address 131.108.251.1 255.255.255.0
```

Note: The two peers using the serial-transport method will only function correctly if there are Cisco router/bridges at the other end of the serial line that have been configured to use the serial transport. The peers must also belong to the same ring group.

Configuring the Largest Frame

Use the **source-bridge largest-frame** global configuration command to configure the largest frame size that is used to communicate with any peers in the ring group. The full syntax of the command follows:

source-bridge largest-frame *ring-group size*
no source-bridge largest-frame *ring-group*

The argument *ring-group* is the ring group number; the argument *size* is the maximum frame size.

The router negotiates all transit routes down to the specified size or lower. This argument is useful in preventing timeouts in end hosts by reducing the amount of data they have to transmit in a fixed interval. For example, in some networks containing slow links, it would be impossible to transmit an 8K frame and receive a response within a few seconds. These are fairly standard defaults for an application on a 16 Mb Token Ring. If the frame size is lowered to 516 bytes, then only 516 bytes must be transmitted and a response received in 2 seconds. This feature is most effective in a network with slow links. The legal values for this argument are 516, 1500, 2052, 4472, 8144, 11407, and 17800 bytes.

Configuring a Proxy Explorer

Cisco has implemented the *proxy explorer* function to allow the Cisco source-route bridge to respond to a particular destination node. Use proxy explorers to limit the amount of explorer traffic propagating through the source-bridge network, especially across low bandwidth serial lines. The proxy explorer is most useful for multiple connections to a single node.

The following conditions must be met in order for a proxy response to occur:

- The destination node must be in the RIF cache.
- The destination node must not be on the same ring as the explorer.
- The explorer packet must be an IEEE 802.2 XID or TEST packet.
- The packet cannot be from the IBM Token Ring LAN Network Manager source SAP.

If all of the above conditions are met, the Cisco source-route bridge will turn the packet around, append the appropriate RIF, and reply to the source node.

Use the **source-bridge proxy-explorer** interface subcommand to configure the interface to respond to any explorer packets that meet the conditions described above. The command has this syntax:

source-bridge proxy-explorer
no source-bridge proxy-explorer

The default is to not respond with proxy explorer packets.

Example:

These commands configure the Cisco router/bridge to use proxy explorers on interface Token Ring 0.

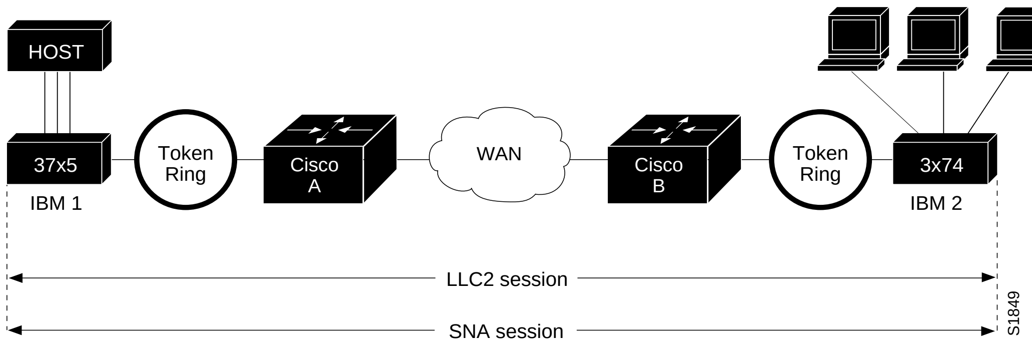
```
interface tokenring 0
source-bridge proxy-explorer
```

The Cisco Local Acknowledgment Function

LLC2, or Logical Link Control–Type 2, an ISO standard data link level protocol used in Token Ring networks, was designed to ensure reliable transmission of data across LAN media having minimal or predictable time delays. With the advent of remote source-route bridging (RSRB) and wide-area network (WAN) backbones, LANs are now separated by wide, geographic distances spanning countries and continents. As a result, these LANs have time delays that are longer than LLC2 allows for bidirectional communication between hosts. The Local Acknowledgment capability in router/bridges supporting RSRB addresses the problem of unpredictable time delays, multiple retransmissions, or loss of user sessions.

Figure 1-11 illustrates an LLC2 session. IBM 1 (for example, a 37x5) on a LAN segment can communicate with IBM 2 (for example, a 3x74) on a different LAN segment separated via a wide area backbone network. Frames are transported between Cisco A and Cisco B using RSRB. However, the LLC2 session between IBM 1 and IBM 2 is still end-to-end; that is, every frame generated by IBM 1 traverses the backbone network to IBM 2, and IBM 2, on receipt of the frame, acknowledges it.

Figure 1-11 LLC2 Session Without Local Acknowledgment

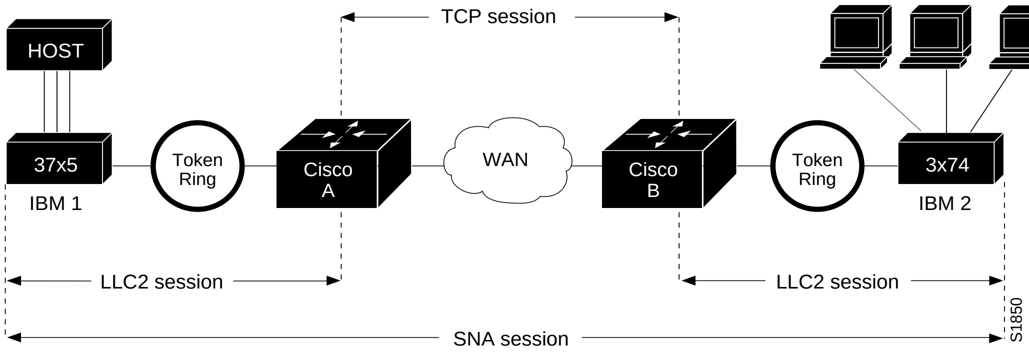


In an LLC2 session, when host A sends a frame to host B, host A expects host B to respond positively or negatively in a certain amount of predefined time commonly called the *T1 time.* If host A does not receive an acknowledgment of the frame it sent to host B within the T1 time, it will retry a few number of times (normally 8 to 10). If there is still no response from host B, host A will drop the session.

On backbone networks consisting of slow serial links, the T1 timer on end hosts could expire before the frames have a chance to reach the remote hosts, causing the end host to retransmit. This results in duplicate frames reaching the remote host at the same time as the first frame also reached the remote host, albeit slowly. These frame duplications break the LLC2 protocol, resulting in the loss of sessions between the two IBM machines.

One way to solve this problem is to increase the timeout value on the end hosts to account for the maximum transit time between the two end machines. However, in networks consisting of hundreds or even thousands of nodes, every machine would need to be reconfigured with new values.With Local Acknowledgment turned on, the LLC2 session between the two IBM end nodes is not end-to-end, as shown in the previous example, but instead terminates at the two local Cisco routers, as shown in Figure 1-12. The LLC2 session with IBM 1 ends at Cisco A and the LLC2 session with IBM 2 ends at Cisco B. Both Cisco A and Cisco B execute the full LLC2 protocol as part of Local Acknowledgment.

Figure 1-12 LLC2 Session with Local Acknowledgment



With Local Acknowledgment enabled in both routers, Cisco A acknowledges frames received from IBM 1. The node IBM 1 still thinks that the acknowledgments it receives are from IBM 2. Cisco A looks like IBM 2 to IBM 1. Similarly, Cisco B acknowledges frames received from IBM 2. The node IBM 2 thinks that the acknowledgments it receives are from IBM 1. Cisco B looks like IBM 1 to IBM 2. As the frames no longer have to travel the WAN backbone networks to be acknowledged, but instead are locally acknowledged by Cisco routers, the end machines do not time out, resulting in no loss of sessions.

The advantages of enabling Local Acknowledgment are:

- Local Acknowledgment solves the T1 timer problem without having to change any configuration on the end nodes.

  The end nodes are unaware that the sessions are being locally acknowledged. In networks consisting of hundreds or even thousands of machines, this is a definite advantage. All the frames acknowledged by the Cisco router appear to the end hosts to be coming from the remote IBM machine. In fact, by looking at a trace from a protocol analyzer, one cannot say if a frame was acknowledged by the local Cisco router or by a remote IBM machine. The MAC addresses and the RIFs generated by the Cisco routers are identical to that generated by the remote IBM machine. The only way to find out if a session is locally acknowledged is to use either a **show local-ack** command or a **show source-route** command on the Cisco router.

- All the supervisory frames (RR, RNR, REJ) frames that are locally acknowledged go no further than the Cisco router.

Without Local Acknowledgment, *every* frame traverses the backbone. With Local Acknowledgment, only data (I-frames) traverse the backbone, resulting in less traffic on the backbone network. For installations in which customers pay for the amount of traffic passing through the backbone, this could be a definite cost-saving measure. A simple protocol exists between the two Cisco *peers* to bring up or down a TCP session.

Notes on the Use of Local Acknowledgment

The Cisco routers at each end of the LLC2 session execute the full LLC2 protocol, which could result in some overhead. The decision to turn on Local Acknowledgment should be based on the speed of the backbone network vis-a-vis the Token Ring speed. For LAN segments separated by slow speed serial links (for example, 56 kbps) the T1 timer problem could occur more frequently. In such cases, it may be wise to turn on Local Acknowledgment. For LAN segments separated by a FDDI backbone, backbone delays will be minimal; in such cases, Local Acknowledgment should not be turned on. Speed mismatch between the LAN segments and the backbone network is one criterion to be used in the decision to use Local Acknowledgment.

There are some situations (such as host B dying between the time A sends data and B would get it), when host A would believe, *at the LLC2 layer*, that data was received that actually was not, because the Cisco router acknowledges that it received data from host A before it knows that host B can actually receive the data. But because both NetBIOS and SNA have error recovery in situations where an end device goes down, these higher-level protocols will resend any missing or lost data. These transaction request/confirmation protocols exist above LLC2, so they are not affected by tight timers, as is LLC2. They also are transparent to Local Acknowledgment.

If you are using NetBIOS applications, note that there are two NetBIOS timers—one at the link level and one at the next higher level. Local Acknowledgment is designed to solve session timeouts at the link level only. If you are experiencing NetBIOS session timeouts, you have two options:

- Experiment with increasing your NetBIOS timers.
- Avoid using NetBIOS applications on slow serial lines.

Configuring Local Acknowledgment for Token Ring Interfaces

Local acknowledgment can only be enabled with TCP remote peers (as opposed to LAN or direct serial-interface remote peers) because the routers need the reliable transmission of TCP to provide the same reliability of the pre-Local Acknowledgment end-to-end connection. Therefore, the direct media encapsulation options for the **source-bridge remote-peer** command cannot be used.

If the LLC2 session between the local host and the router terminates in either router, the other will be informed to terminate its connection to its local host.

If the TCP queue length of the connection between the two Cisco routers reaches 90% of its limit, the Cisco routers will send Receiver-not-ready (RNR) messages to the local hosts until the queue limit is reduced to below this limit.

The configuration of the LLC2 parameters for the local Token Ring interfaces can affect overall performance. Please refer to the chapter “LLC2 and SDLC Link-Level Support” for more details about fine-tuning your network through the LLC2 parameters.

Note: As previously stated, Local Acknowledgment is meant only for extreme cases where communication is otherwise not possible. As the router must maintain a full LLC2 session, the number of simultaneous sessions the router can support before performance degrades depends on the mix of other protocols and their loads also running in the router.

Setting Up Local Acknowledgment

Set up Local Acknowledgment by specifying the **local-ack** keyword of the global **source-bridge remote-peer** command.

source-bridge remote-peer *ring-group* **tcp** *ip-address* [**lf** *size*][**local-ack**] [**version** *number*]

Note: All Local Acknowledgment commands are global configuration commands.

Setting Up on a Per-Remote-Peer Basis

The **source-bridge remote-peer** subcommand is given a new option **local-ack** that indicates that LLC2 sessions destined to a specific remote peer are to be locally administered and acknowledged. The following example configuration demonstrates setting up one remote peer for Local Acknowledgment:

```
.
.
.
 source-bridge ring-group 100
 source-bridge remote-peer 100 tcp 1.1.1.1
 source-bridge remote-peer 100 tcp 1.1.1.2 local-ack
.
.
.
 interface tokenring 0
 source-bridge 1 1 100
.
.
.
```

This configuration allows the following: sessions between a device on Token Ring 0 that must go through remote peer 1.1.1.2 *will* use Local Acknowledgment, but sessions that must go through remote peer 1.1.1.1 *will not* use Local Acknowledgment (that is, they will "pass through").

Note: Both peers need to be configured for Local Acknowledgment. If only one is so configured, unpredictable (and undesirable) results will occur.

Setting Up on a Per-Ring Basis

The **local-ack** parameter of the **source-bridge remote-peer** command locally terminates every new LLC2 session. When you want only some sessions on a few rings to be locally acknowledged and the remaining to pass through, use t.he following command:

source-bridge passthrough *ring-number*
no source-bridge passthrough *ring-number*

The global command **source-bridge passthrough** indicates that if a machine on a Token Ring attempts to start an LLC2 session to an end host that exists on the *ring-number*, the session will “pass through” and not use Local Acknowledgment. More than one **source-bridge passthrough** command may be given. The *ring-number* is either the start ring or the destination ring for the two IBM end machines. The **no** version of the command disables all the rings and allows the session to locally acknowledge.

For example, the following list of commands indicates that rings 4 and 7 are *never* to have Local Acknowledgment applied to them, even if they are accessed through remote peers for which **local-ack** has been specified in the **source-bridge remote-peer** command:

```
source-bridge passthrough 4
source-bridge passthrough 7
```

Performance Tuning for LLC2

In some situations, you may discover that default settings for LLC2 configurations may not be acceptable. In such a case, you may configure LLC2 for optimal use. the chapter “LLC2 and SDLC Link-Level Support” describes the LLC2 commands and how you can use them to optimize your network performance. Significant improvements in network performance can be had through judicious use of these commands.

Displaying Local Acknowledgment Statistics

The command **show local-ack** shows the current state of any current Local Acknowledgment connections, as well as any configured passthrough rings.

show local-ack

Example:

```
router#show local-ack
local 1000.5a59.04f9, lsap 04, remote 4000.2222.4444, dsap 04
llc2 = 1798136, local ack state = connected
Passthrough Rings: 4 7
```

The first two lines of the **show local-ack** command state that there is a Local Acknowledgment session between two Token Ring devices. The device on the local ring has a MAC address of 1000.5a59.04f9 with a sap of 04. The remote device has a MAC address of 4000.2222.4444 with a sap of 04. The state of the Local Acknowledgment session is connected.

The Passthrough Rings display is independent of the rest of the **show local-ack** command. The Passthrough Rings display indicates that there are two rings, 4 and 7, configured for Passthrough. This means that stations on these rings will not have their sessions locally acknowledged but will instead have their acknowledgments end-to-end.

Table 1-1 describes the fields shown in the above example.

Table 1-1 Show Local-Ack Field Descriptions

Field	Description
Local	Indicates the MAC address of the local Token Ring station with which the router has the LLC2 session.
Lsap	The local SAP (service access point) value of the Token Ring station with which the router has the LLC2 session.
Remote	The MAC address of the remote Token Ring station on whose behalf the router is providing acknowledgments. The remote Token Ring station is separated from the router via the TCP backbone.
Dsap	The destination SAP value of the remote Token Ring station on whose behalf the router is providing acknowledgments.
LLC2	A pointer to an internal data structure used by Cisco staff for debugging.
Local ack state:	Any of:
disconnected	No session between the two end hosts.
connected	Full data transfer possible between the two.
awaiting connect	This router is waiting for the other end to confirm a session establishment with the remote host.

Debugging Local Acknowledgment

There are two **debug** commands for Local Acknowledgment: **debug local-ack** and **debug local-ack-state**. For each of the following **debug** commands there is a corresponding **undebug** command that stops the output.

debug local-ack

This command displays Local Acknowledgment debugging information useful for Cisco staff.

debug local-ack-state

This command prints out the new and the old state conditions whenever there is a state change in the Local Acknowledgment state machine. This command is used by Cisco staff for debugging purposes.

Bridging Between Source-Route Bridge Groups and Transparent Bridge Groups (SR/TLB)

This section describes the support for bridging between source-route bridge groups and transparent bridge groups using the Cisco router's Source-Route Translational Bridging (SR/TLB) functionality. This section assumes familiarity with Cisco's implementation of both source-route and transparent bridging. SR/TLB allows you to combine source-route and transparent bridged networks without the need to convert all of your existing source-route bridges to transparent bridge-capable (SRT) nodes. As such, it provides a cost-effective connectivity path between, for example, Ethernets and Token Rings.

Overview of SR/TLB

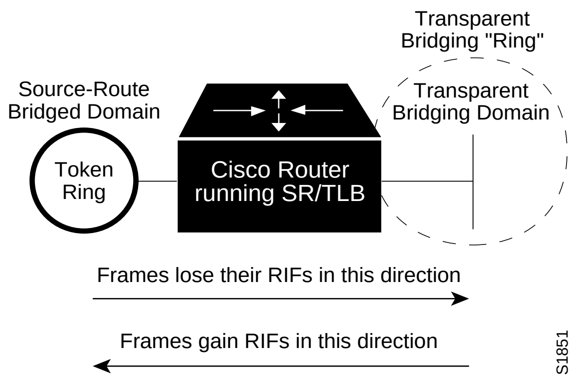
You may bridge packets between a source-route bridging domain and a transparent bridging domain. Using this feature, a software "bridge" is created between a specified virtual ring group and a transparent bridge group. To the source-route station, this bridge looks like a standard source-route bridge. There is a ring number and a bridge number associated with a "ring" that actually represents the entire transparent bridging domain. To the transparent bridging station, the bridge represents just another port in the bridge group.

When bridging from the source-route (typically, Token Ring) domain to the transparent bridging (typically, Ethernet) domain, the source-route fields of the frames are removed. This RIF is cached for use by subsequent return traffic.

When bridging from the transparent bridging domain to the source-route bridged domain, the packet is checked to see if it has a multicast or broadcast destination, or a unicast (single host) destination. If it has a multicast, the packet is sent as a spanning-tree explorer. If it has a unicast destination, a RIF path is looked up to the destination in the RIF cache. If a path is found, it will be used, otherwise the packet will be sent as a spanning-tree explorer.

An example of a simple topology is shown in Figure 1-13.

Figure 1-13 Example of a Simple SR/TLB Topology


Transparent Bridging "Ring"
Source-Route Bridged Domain
Transparent Bridging Domain
Token Ring
Cisco Router running SR/TLB
Frames lose their RIFs in this direction
Frames gain RIFs in this direction
S1851

Note: The spanning-tree protocol messages, used to prevent loops in the transparent bridging domain, are *not* passed between the source-route domain and the transparent bridging domain. Therefore, you must not set up multiple paths between the source-route and transparent bridging domains.

Configuring SR/TLB

A new subcommand of the global **source-bridge** configuration command is used to establish bridging between transparent bridging and source-route bridging.

Note: Before using this command, you must have previously completely configured your router using multiport source-bridge and transparent bridging.

The command syntax is:

> **source-bridge transparent** *ring-group pseudo-ring bridge-num tb-group*
> **no source-bridge transparent** *ring-group pseudo-ring bridge-num tb-group*

The *ring-group* argument is a virtual ring group created by the **source-bridge ring-group** command. This is the source-bridge virtual ring to associate with the transparent bridge group.

The *pseudo-ring* argument is the ring number used to represent the transparent bridging domain to the source-route bridged domain. This number must be a unique number, not used by any other ring in your source-route bridged network.

The *bridge-num* argument is the bridge number of the bridge which leads to the transparent bridging domain.

The *tb-group* argument is the number of the transparent bridge group that you wish to tie into your source-route bridged domain. The **no** version of the command disables this feature.

The steps involved in configuring SR/TLB are to:

Step 1: Configure source-route bridging and transparent bridging.

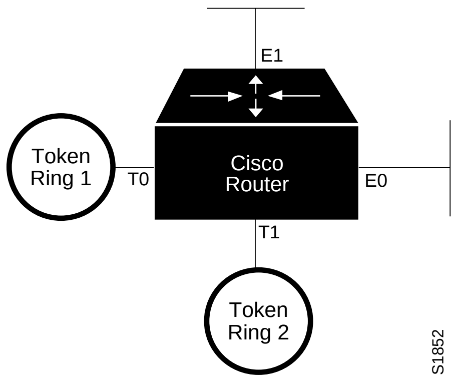
Step 2: Choose a pseudo-ring number for the transparent bridging domain.

Step 3: Choose a bridge-number for the path to the transparent bridging domain.

Step 4: Enter the **source-bridge transparent** command in the configuration.

In the simple example illustrated in Figure 1-14, a four-port router with two Ethernets and two Token Rings is used to connect transparent bridging on the Ethernets to source-route bridging on the Token Rings.

Figure 1-14 Example of a Simple SR/TLB Configuration



Assume the following configuration for source-route bridging and transparent bridging existed before the desire to enable SR/TLB:

```
!
interface tokenring 0
source-bridge 1 1 2
!
interface tokenring 1
source-bridge 2 1 1
!
interface Ethernet 0
bridge-group 1
!
interface Ethernet 0
bridge-group 1
!
bridge 1 protocol dec
```

One aspect of this configuration must change immediately: the two Token Ring interfaces are communicating with two-port local source-route bridging. The addition of SR/TLB creates a third ring, so these two interfaces must be reconfigured to communicate through a virtual ring, as shown:

```
!
source-bridge ring-group 10
!
interface tokenring 0
source-bridge 1 1 10
!
interface tokenring 1
source-bridge 2 1 10
!
interface ethernet 0
bridge-group 1
!
interface ethernet 1
bridge-group 1
!
bridge 1 protocol dec
```

At this point, the configuration is ready for the addition of the **source-bridge transparent** bridge command. You must determine two things:

- A ring number for the pseudo-ring. It must be unique throughout the source-route bridged network. For the example configuration above, use a 3.
- A bridge number for the path to the pseudo-ring. For the example configuration above, use a 1.

Once you have determined the ring number and the bridge number, the command can then be added, giving this final configuration:

```
source-bridge ring-group 10
source-bridge transparent 10 3 1 1
!
interface tokenring 0
source-bridge 1 1 10
!
interface tokenring 1
source-bridge 2 1 10
!
interface ethernet 0
bridge-group 1
!
interface ethernet 1
bridge-group 1
!
bridge 1 protocol dec
```

The meaning of each field in the **source-bridge transparent** command from the above configuration is described in Table 1-2:

Table 1-2 Source-Bridge Command Field Descriptions

Field	Description
10	The ring-group to which the command applies; corresponds to the ring-group number given in the **source-bridge** commands.
3	The ring number assigned to the pseudo-ring.
1 (first)	The bridge number leading to the pseudo-ring.
1 (second)	The transparent bridge-group tied into this source-route bridged ring group; corresponds to the bridge number given in the transparent bridging commands.

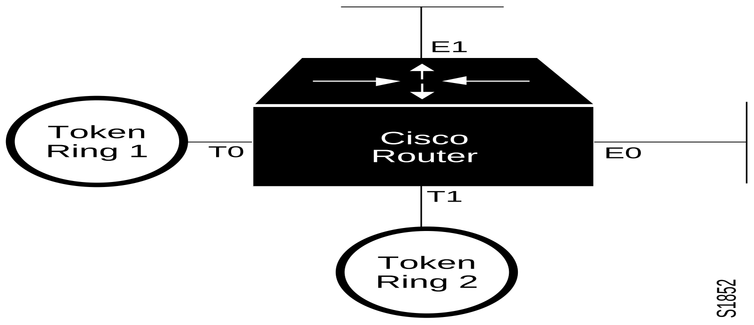
Further Notes on the Use of SR/TLB

The following notes and caveats apply to all uses of SR/TLB:

- There cannot exist multiple paths between the source-route bridged domain and the transparent bridged domain. Such paths can lead to data loops in the network, as the spanning-tree packets used to avoid these loops in transparent bridged networks do not traverse the source-route bridged network.
- Many higher-level protocols, such as IP with its ARP frames, and XNS and Novell IPX with their addressing structure, violate protocol boundaries by putting MAC-level addresses into the data portion of the frame. SR/TLB, in translating between media, cannot hope to translate these addresses in the data portion of the frame. This will lead to confusing situations on end hosts. Therefore, while routing of these frames between Token Ring and Ethernet will work, bridging these frames will not. In general, it is best to assume that no normally routable protocol can be bridged with SR/TLB.
- It has been discovered that some devices, notably PS/2s under certain configurations running OS/2 Extended Edition Version 1.3, do not correctly implement the "largest frame" processing on RIFs received from remote source-route bridged hosts. The maximum Ethernet frame size is smaller than that allowed for Token Ring. As such, bridges allowing for communication between Ethernet and Token Ring will tell the Token Ring hosts, through the RIF on frames destined to the Token Ring, that hosts on the Ethernet cannot receive frames larger than a specified maximum, typically 1472 bytes. Some machines ignore this run-time limit specification and send frames larger than the Ethernet can accept. The Cisco router and any other Token Ring/Ethernet bridge has no choice but to drop these frames. To allow such hosts to successfully communicate across or to an Ethernet, you must manually configure their maximum frame size. For the PS/2, this can be done through Communications Manager.
- Any access filters applied on any frames apply to the frames as they appear on the media to which the interface with the access control applies. This is important because in the far most common use of SR/TLB, Ethernet and Token Ring connectivity, the bit ordering of the MAC addresses in the frame is swapped.

Assume you have the following setup, where you want to connect only a single machine, HostE, on an Ethernet to a single machine, HostR, on the Token Ring, as shown in Figure 1-15:

Figure 1-15 Example of a Bit-Swapped Address



You want to allow only these two machines to communicate across the router. Therefore, you might create the following configuration to restrict the access. This configuration will not work, for the last reason stated in the notes above.

Note: For the sake of readability, the commands to control the bridging are not shown here, just the commands to control the filtering.

```
interface tokenring 0
access-expression output smac(701)
!
interface ethernet 0
bridge-group 1 input-address-list 701
!
access-list 701 permit 0110.2222.3333
```

The Token Ring interface command states to apply the access-list 701 on the Source Address of frames going out to the Token Ring, and the Ethernet interface command says to apply the access list on the source address frames entering the interface from Ethernet. This would work fine if both interfaces used the same bit ordering, but Token Rings and Ethernets use opposite (swapped) bit orderings in their addresses in relationship to each other. Therefore, the address of HostE on the Token Ring is not 0110.2222.3333, but rather 8008.4444.cccc. The following configuration is better. In general, the best way to keep this straight is to keep your access lists for Token Ring and Ethernet completely separate from one another.

```
interface tokenring 0
source-bridge input-address-list 702
!
interface ethernet 0
bridge-group 1 input-address-list 701
!
access-list 701 permit 0110.2222.3333
!
access-list 702 permit 0110.1234.5678
```

Enabling Translation Compatibility with IBM 8209 Bridges

To transfer data between IBM 8209 Ethernet/Token Ring bridges and Cisco routers running the SR/TLB software, (to create a Token Ring backbone to connect Ethernets), issue the following global command on each Cisco router:

bridge old-oui
no bridge old-oui

This command, when given, causes the OUI code in Token Ring frames translated to and from Ethernet Type II to be 0x000000. The default OUI value, used when this command is not given, is 0x0000F8. This default value is compatible with the draft value used in various SRT protocol documents.

The **no** version of the command restores the default value.

Even with the **source-bridge old-oui** command specified to match the OUIs, there is one caveat in that Ethernet SNAP frames traversing a Token Ring backbone made up of both Cisco SR/TLB routers and IBM 8209 bridges will be converted into Ethernet Type II frames on output to the Ethernet attached to the Cisco router. This restriction does not apply to networks built entirely of Cisco SR/TLB routers.The following paragraphs describe the differences in design philosophies between Cisco routers and IBM 8209 bridges that result in this caveat when you use both products.

The Cisco router, acting as a SR/TLB, can distinguish immediately on input on Token Ring interfaces between frames that started on an Ethernet as an Ethernet Type II frame, and those that started on an Ethernet as a SNAP-encapsulated frame. This distinction is possible because the Cisco router uses the 0x0000F8 OUI when converting Ethernet Type II frames into Token Ring SNAP frames, and leaves the OUI as 0x000000 for Ethernet SNAP frames going to a Token Ring. This distinction in OUIs leads to efficiencies in the design and execution of the Cisco SR/TLB product, as no tables need be kept to know which Ethernet hosts use SNAP encapsulations and which hosts use Ethernet Type II.

The IBM 8209 bridge, however, by using the 0x000000 OUI for all frames entering the Token Ring, must take extra measures to perform the translation. For every station on each Ethernet, the 8209 bridges attempt to remember the frame format used by each station, and assume that once a station sends out a frame using Ethernet Type II or 802.3, it will always continue to do so. It must do this because in using 0x000000 as an OUI, there is no way to distinguish between SNAP and Type II frame types. As the Cisco SR/TLB router does not need to keep this database, when 8209 compatibility is enabled with the **source-bridge old-oui** command, it chooses to translate all Token Ring SNAP frames into Ethernet Type II frames. As virtually every nonroutable protocol on Ethernet uses either non-SNAP 802.3 (which traverses fully across a mixed IBM 8209/Cisco router Token Ring backbone) or Ethernet Type II, this results in correct interconnectivity for virtually all applications.

Using SR/TLB in Many IBM LLC2 Environments (0x80d5 Processing)

Many Ethernet-attached IBM devices, including PS/2s running OS/2 Extended Edition and RT-PCs, use a nonstandard encapsulation of LLC2 on Ethernet to allow for communication on Ethernets that support only Ethernet Type 2 communication, and not Ethernet 802.3 frame formats. Such an IBM device does not place its LLC2 data inside an 802.3 format frame, but rather places it into an Ethernet Type II frame whose type is specified as *0x80d5*. Cisco routers support both. The format of these frames is given in the appendix "Frame Conversions."

Note: As these frames were first widely seen on the RT-PC, this format is sometimes called *RT-PC Ethernet format*.

The Cisco router supports the conversion between Token Ring LLC2 data and the Ethernet 0x80d5 format. In so doing, there are now two ways in the router to specify a translation between Ethernet and Token Ring for LLC2 frames.

By default, *0x80d5 processing* is disabled. As such, Token Ring LLC2 frames are translated in a straightforward manner into Ethernet 802.3 LLC2 frames. This works for most non-IBM hosts, however, the hosts using the special Ethernet Type 2 0x80d5 format cannot read these frames.

source-bridge enable-80d5
no source-bridge enable-80d5

This command changes the Cisco router's Ethernet/Token Ring translation behavior to translate Token Ring LLC2 frames into Ethernet *0x80d5 format* frames. The **no** version of the command restores the default behavior.

However, after specifying this command, you may still need to allow some other, non-IBM hosts to use the other form of LLC2 translation. As such, the translation method can be specified on a per-DSAP basis with the following command:

source-bridge sap-80d5 *sap*
no source-bridge sap-80d5 *sap*

These commands enable, or disable with the **no** version of the command, the use of the Ethernet *0x80d5 format* for Ethernet/Token Ring translation of LLC2 frames. By default, the following DSAPS are enabled for 0x80d5 translation simply by specifying the **source-bridge enable-80d5** command:

- 04, 08, 0C—For SNA
- F0—For NetBIOS

Any of these may be disabled with the **no** version of this command.

The parameters specifying the current parameters for the processing of 0x80d5 frames are given at the end of the output of **show span**.

The following commands enable 0x80d5 processing, remove the translation for SAP 08, and add the translation for SAP 1c:

```
source-bridge enable-80d5
no source-bridge sap-80d5 08
source-bridge sap-80d5 1c
```

Note: The 80d5 frame processing option is available only with SR/TLB. It is not available when Source-Route Transparent (SRT) bridging is used.

A partial output of **show span** given the above configuration follows:

```
router#show span

...extra output deleted...

Translation between LLC2 and Ethernet Type II 80d5 is enabled
Translation is enabled for the following DSAPs:
 04 0C 1C F0
```

Frame Conversions Between Source-Route and Transparent Bridging

The frame conversions between the two bridging protocols are given in the appendix "Frame Conversions."

Configuring Administrative Filtering of Source-Routed Frames

Source-route bridges normally filter frames according to the routing information contained in the frame. That is, a bridge will not forward a frame back to its originating network segment or any other network segment that the frame has already traversed. Further types of filtering (administrative filtering) can be specified by the network manager.

Administrative filtering can be done by:

- Token Ring address
- Protocol type—IEEE 802 or SNAP
- Token Ring Vendor code
- NetBIOS host names
- Arbitrary bytes within a NetBIOS frame

Filtering by Token Ring address or vendor code will cause no significant performance penalty. However, performance will be significantly affected when filtering by protocol type. A list of SNAP (Ethernet) type codes is provided in the appendix "Ethernet Type Codes."

Administrative Filtering by Protocol Type

The access list mechanism permits filtering by protocol type. Use the bridge **access-list** command to specify an element in an access list. The order in which **access-list** commands are entered into the system 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, then a deny decision is reached.

Note: If a *single condition* is to be denied, then there must be an **access-list** command that permits *everything* as well, or *all* access will be denied.

Configuring Protocol Type Access Lists

Use the bridge **access-list** global configuration command to configure the access list mechanism for filtering frames by protocol type. The command has this syntax:

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

The argument *list* is a user-selectable number in the range 200 – 299, inclusive, 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. Specify either a Link Service Access Point (LSAP) type code for 802-encapsulated packets, or a SNAP type code for SNAP-encapsulated packets. (LSAP, sometimes called SAP, refers to the type codes found in the DSAP and SSAP fields of the 802 header.)

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 0x0101. This is because these two bits are used for purposes other than identifying the SAP code. The **no** version removes the single specified entry from the access list.

Example:

In this example, the access list permits only Novell frames (LSAP 0xE0E0) and filters out all other frame types. This set of access lists would be applied to an interface via the **source-bridge input-lsap list** or **source-bridge input-lsap list** command (described in following sections).

```
access-list 201 permit 0xE0E0 0x0101
access-list 201 deny  0x0000 0xFFFF
```

Combine the DSAP/LSAP fields into one number to do LSAP filtering, for example, 0x E0E0—not 0xE0. Note that the deny condition specified in the above example is not required; access lists have an implicit deny as the last statement. Adding this statement can serve as a useful reminder, however.

The following access list filters out only SNAP type codes assigned to DEC (0x6000 through 0x6007) and lets all other types pass. This set of access lists would be applied to an interface via the **source-bridge input-type list** or **source-bridge output-type-list** command which are described in the following sections.

```
access-list 202 deny     0x6000 0x0007
access-list 202 permit   0x0000 0xFFFF
```

Note: Use the last item of an access list to specify a default action, for example, to permit everything else or to deny everything else. If nothing else in the access list matches, then the default action is to deny access; that is, filter out all other type codes.

Type code access lists will negatively affect system performance by greater than 30 percent. Therefore, Cisco Systems recommends keeping the lists as short as possible and using wild card bit masks whenever possible.

Filtering IEEE 802 Frames on Input

You can filter IEEE 802-encapsulated packets on input. The access list specifying the type codes to be filtered is given by this variation of the **source-bridge** interface subcommand:

source-bridge input-lsap-list *list*

The argument *list* is the access list number. This access list is applied to all IEEE 802 frames received on that interface prior to the source-routing process. Specifying a zero disables the filter.

Example:

This command specifies access list 203.

```
source-bridge input-lsap-list 203
```

Filtering IEEE 802 Frames on Output

The software allows you to filter IEEE 802-encapsulated packets on output. The access list specifying the type codes to be filtered is given by this variation of the **source-bridge** interface subcommand:

source-bridge output-lsap-list *list*

The argument *list* is the access list number. This access list is applied just before sending out a frame to an interface. Specifying a zero disables the filter.

Example:

This command specifies access list 251.

```
source-bridge output-lsap-list 251
```

Filtering SNAP Frames on Input

To filter SNAP-encapsulated packets on input, use the access list specifying the type codes to be filtered with this variation of the **source-bridge** interface subcommand:

source-bridge input-type-list *list*

The argument list is the access list number. This access list is then applied to all SNAP frames received on that interface prior to the source-routing process. Specify a zero for *list* to disable the application of the access list on the bridge group.

Example:

This command specifies access list 202.

```
source-bridge input-type-list 202
```

Filtering SNAP Frames on Output

To filter SNAP-encapsulated frames on output, use the access list specifying the type codes to be filtered. This is entered with this variation of the **source-bridge** interface subcommand:

> **source-bridge output-type-list** *list*

The argument *list* is the access list number. This access list is then applied just before sending out a frame to an interface. Specify a zero for *list* to disable the application of the access list on the bridge group.

Note: Input and output type code filtering on the same interface reduces performance and is not recommended.

Access lists for Token Ring- and IEEE 802-encapsulated packets affect only source-route bridging functions. Such access lists do not interfere with protocols that are being routed.

Example:

The following example allows only AppleTalk Phase 2 packets to be source-route bridged between Token Rings 0 and 1, and allows Novell packets only to be source-route bridged between Token Rings 2 and 3.

```
source-bridge ring-group 5
!
interface tokenring 0
ip address 131.108.1.1 255.255.255.0
source-bridge 1000 1 5
source-bridge spanning
source-bridge input-type-list 202
!
interface tokenring 1
ip address 131.108.11.1 255.255.255.0
source-bridge 1001 1 5
source-bridge spanning
source-bridge input-type-list 202
!
interface tokenring 2
ip address 131.108.101.1 255.255.255.0
source-bridge 1002 1 5
source-bridge spanning
source-bridge input-lsap-list 203
!
interface tokenring 3
ip address 131.108.111.1 255.255.255.0
```

```
source-bridge 1003 1 5
source-bridge spanning
source-bridge input-lsap-list 203
!
! SNAP type code filtering
! permit ATp2 data (0x809B)
! permit ATp2 AARP (0x80F3)
access-list 202 permit 0x809B 0x0000
access-list 202 permit 0x80F3 0x0000
access-list 202 deny 0x0000 0xFFFF
!
! LSAP filtering
! permit IPX (0xE0E0)
access-list 203 permit 0xE0E0 0x0101
access-list 203 deny 0x0000 0xFFFF
```

Note that it is not necessary to check for an LSAP of 0xAAAA when filtering SNAP encapsulated AppleTalk packets, because for source-route bridging, the use of type filters implies SNAP encapsulation.

Administrative Filtering by Vendor Code or Address

To configure administrative filtering by vendor code or address, define access lists which look for Token Ring addresses or for particular vendor codes for administrative filtering. No noticeable performance will be lost in using these access lists. The lists can be of indefinite length.

Configuring Vendor Code Access Lists

To configure a vendor code access list, use the global bridge **access-list** command for IEEE 802 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, inclusive, and *address* and *mask* are 48-bit Token Ring addresses written in dotted triplet form. The ones bits in *mask* are the bits to be ignored in *address*. See the section “Filtering Destination Addresses” for an example of command use.

Note that for source address filtering, the mask should always have the high order bit set. This is because the IEEE 802 standard uses this bit to indicate whether a RIF is present, and not as part of the source address.

Filtering Incoming Frames

To configure filtering on IEEE 802 source addresses, assign an access list to a particular interface for filtering the Token Ring or IEEE 802 source addresses. Use this variation of the **source-bridge** interface subcommand to do this:

source-bridge input-address-list *list*
no source-bridge input-address-list *list*

The argument *list* is the access list number. The **no** version of this command removes the application of the access list. See the section "Filtering Destination Addresses" for an example of command use.

Filtering Outgoing Frames

To configure filtering on IEEE 802 source addresses, assign an access list to a particluar interface for filtering the Token Ring or IEEE 802 source address. Use this variation of the **source-bridge** interface subcommand to do this:

source-bridge output-address-list *list*
no source-bridge output-address-list *list*

The argument *list* is the access list number.The **no** version of this command removes the application of the access list.

Note: The **source-bridge output-address-list** command uses the source address for filtering, which differs from other MAC level access lists that use the destination address for output filtering.

Example:

To disallow the bridging of Token Ring packets of all IBM workstations on Token Ring 1, use this sample configuration. Software assumes that all such hosts have Token Ring addresses with the vendor code 1000.5A00.0000. The first line of the access list denies access to all IBM workstations while the second line permits everything else. Then, the access list can be assigned to the input side of Token Ring 1.

```
access-list 700 deny 1000.5A00.0000   8000.00FF.FFF
access-list 700 permit 0000.0000.0000   FFFF.FFFF.FFF
interface token ring 1
source-bridge input-address-list 700
```

Configuring NetBIOS Access Control Filtering

NetBIOS is the interface used by the IBM Token Ring/PC Network Interconnect Program to transmit messages between stations, typically IBM PCs, on a Token Ring network. NetBIOS allows messages to be exchanged between the stations using a name rather than a station address. Each station knows its name and is responsible for knowing the names of other stations on the network.

The Cisco bridging software provides for configuring access control filters for packets transmitted across a Token Ring bridge using the NetBIOS interface. Two types of filters may be configured: one on source and destination station names, and one on arbitrary byte patterns in the packet itself.

Configuring Access Control Using Station Names

To configure access control using station names, follow these steps:

Step 1: Define the access list name and specify the access condition, either permit or deny.

Step 2: Specify the direction of the message to be filtered on the interface. The choices are incoming or outgoing messages.

Configuration Tips and Notes

Keep the following notes in mind as you configure NetBIOS access control:

- The access lists are scanned in the order they are entered.
- There is no way to put a new access list entry in the middle of an access list. All new additions to existing NetBIOS access lists are placed at the end of the existing list.
- The case of the letters you use to enter arguments is important. The software makes a literal translation, so that a lowercase "a" is different from an uppercase "A." (Most nodes are named in uppercase letters.)
- You may have both a host NetBIOS access list and byte NetBIOS access list with the same name. The two lists are identified as unique and bear no relationship to each other.
- The names in the access lists are compared with the source name field for NetBIOS commands 00 and 01 (ADD_GROUP_NAME_QUERY and ADD_NAME_QUERY) and destination name field for NetBIOS commands 08 and 0A (DATAGRAM and NAME_QUERY).
- If a station name is not found in an access list, the default action is to deny the access.

Note: The NetBIOS access filters are implemented to minimize their performance hit by not having them examine all packets. Rather, the filters examine a select few that are required to allow new NetBIOS client/server connections from forming and existing in the long term, thereby effectively stopping new access and load across the router. However, existing sessions will not be stopped immediately with the application of a new access filter. All new sessions will be filtered by the filter, but the existing sessions could stay for some time.

Assigning the Station Access List Name

The NetBIOS station access list contains the station name with which to match, along with a permit or deny condition. Use the **netbios access-list host** global configuration command to assign the name of the access list to a station or set of stations on the network. The full command syntax follows:

> **netbios access-list host** *name* {**permit**|**deny**} *pattern*
> **no netbios access-list host** *name* {**permit**|**deny**} *pattern*

The argument *name* is the name of the access list being defined.

The argument *pattern* is a set of characters. The characters can be the name of the station, or a combination of characters and pattern matching symbols that establish a pattern for a set of NetBIOS station names. This can be especially useful when stations have names with the same characters, such as a prefix. Table 1-3 explains the pattern matching symbols that can be used.

Table 1-3 Station Name Pattern Matching Characters

Character	Field
*	Used at the end of a string to match any character or string of characters.
?	Matches any single character.

The **no netbios access-list host** command removes an entire list, or just a single entry from a list, depending upon the argument given for *pattern*.

Examples:

This command specifies a full station name to match.

```
netbios access-list host marketing permit ABCD
```

This command specifies a prefix where the pattern matches any name beginning with the characters DEFG. Note that the string DEFG itself is included in this condition.

```
netbios access-list host marketing deny DEFG*
```

This command permits any station name with the letter W as the first character and the letter Y as the third character in the name. The second and fourth letters in the name can be any character. This example would allow stations named WXYZ and WAYB; however, stations named WY and WXY would not be included in this statement, as the ? must match some specific character in the name.

```
netbios access-list host marketing permit W?Y?
```

This example illustrates how to combine wildcard characters:

```
netbios access-list host marketing deny AC?*
```

The command specifies that the marketing list deny any name beginning with AC that is at least three characters in length (the ? would match any third character). The string ACBD and ACB would match, but the string AC would not.

This command removes the entire marketing NetBIOS access list.

```
no netbios access-list host marketing
```

To remove single entries from the list, use a command such as the following:

```
no netbios access-list host marketing deny AC?*
```

This example removes only the list that filters station names with the letters AC at the beginning of the name.

Keep in mind that the access lists are scanned in order. In this example the first list denies all entries beginning with the letters ABC, including one named ABCD. This voids the second command because the entry permitting a name with ABCD comes *after* the entry denying it.

```
netbios access-list host marketing deny ABC*
netbios access-list host marketing permit ABCD
```

Specifying a Station Access List Filter on Incoming Messages

To define an access list filter on incoming messages, use the **netbios input-access-filter host** interface subcommand. The full command syntax follows:

netbios input-access-filter host *name*
no netbios input-access-filter host *name*

The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list host** global configuration commands.

Use the **no netbios input-access-filter host** command with the appropriate argument to remove the entire access list.

Example:

These commands filter packets coming into Token Ring unit 1 using the NetBIOS access list named *marketing*.

```
interface token 1
netbios input-access-filter host marketing
```

Specifying a Station Access List Filter on Outgoing Messages

To define an access list filter on outgoing messages, use the **netbios output-access-filter host** interface subcommand. The full command syntax follows.

netbios output-access-filter host *name*
no netbios output-access-filter host *name*

The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list** global configuration commands.

Use the **no netbios output-access-filter host** command to remove the entire access list.

Example:

These commands filter packets leaving Token Ring unit 1 using the NetBIOS access list named *engineering*.

```
interface token 1
netbios output-access-filter host engineering
```

Configuring Access Control Using a Byte Offset

To configure access control using a byte offset, follow these steps:

Step 1: Define the access list name and specify the access condition, either permit or deny.

Step 2: Specify the direction of the message to be filtered on the interface. The choices are incoming or outgoing messages.

Configuration Tips and Notes

Keep the following notes in mind while configuring access control using a byte offset:

- The access lists are scanned in the order in which they are entered.
- There is no way to put a new access list entry in the middle of an access list. All new additions to existing NetBIOS access lists are placed at the end of the existing list.
- When an access list entry has an offset plus the length of the pattern that is larger than the packet's length, the entry will not make a match for that packet.
- You may have both a host NetBIOS access list and byte NetBIOS access list with the same name. The two lists are identified as unique and bear no relationship to each other.
- As these access lists allow arbitrary byte offsets into packets, these access filters can have a significant impact on the amount of packets per second transiting across the bridge. They should be used only when situations absolutely dictate their use.
- If a station name is not found in an access list, the default action is to deny the access.

Assigning the Bytes Access List Name

The NetBIOS byte offset access list contains a series of offsets and hexadecimal patterns with which to match byte offsets in NetBIOS packets. Use the **netbios access-list bytes** global configuration command to define the offset and patterns. The full command syntax follows:

netbios access-list bytes *name* {**permit**|**deny**} *offset pattern*
no netbios access-list bytes *name* {**permit**|**deny**} *offset pattern*

The argument *name* is the name of the access list being defined.

The argument *offset* is a decimal number indicating the number of bytes into the packet where the byte comparison should begin. An offset of zero points to the very beginning of the NetBIOS header. Therefore, the NetBIOS delimiter string (0xffef), for example, begins at offset 2.

The argument *pattern* is a hexadecimal string of digits representing a byte pattern. The argument *pattern* must conform to certain conventions. These conventions follow.

Byte Offset Pattern Matching

The byte pattern must be an even number of hex digits in length. The byte pattern in the following example is legal:

```
netbios access-list bytes marketing permit 3 0xabcd
```

But the byte pattern in this example would not be accepted:

```
netbios access-list bytes marketing permit 3 0xabc
```

The byte pattern must be no more than 16 bytes (32 hexadecimal digits) in length. The byte pattern in this example would *not* be permitted:

```
netbios access-list bytes marketing permit 3 00112233445566778899aabbccddeeff00
```

You can specify a wildcard character in the byte string indicating that the value of that byte does not matter in the comparison. This is done by specifying two asterisks (**) in place of digits for that byte. For example, the following command would match 0xabaacd, 0xab00cd, and so on.

```
netbios access-list bytes marketing permit 3 0xab**cd
```

Examples:

This command deletes the entire marketing NetBIOS access list named *marketing*.

```
no netbios access-list bytes marketing
```

This command removes a single entry from the list:

```
no netbios access-list bytes marketing deny 3 0xab**cd
```

Remember that, as with all Cisco access lists, the NetBIOS access lists are scanned in order. In the following example, the first line serves to deny all packets with a byte pattern starting in offset 3 of 0xab. However, this denial would also include the pattern 0xabcd because the entry permitting the pattern 0xabcd comes *after* the first entry:

```
netbios access-list bytes marketing deny 3 0xab
netbios access-list bytes marketing permit 3 0xabcd
```

Specifying a Bytes Access List Filter on Incoming Messages

To define an access list filter on incoming messages, use the **netbios input-access-filter bytes** interface subcommand. The full command syntax follows:

netbios input-access-filter bytes *name*
no netbios input-access-filter bytes *name*

The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list bytes** global configuration commands.

Use the **no netbios input-access-filter bytes** command with the appropriate name to remove the entire access list.

Example:

These commands illustrate how to specify a filter on packets coming into Token Ring unit 1 of the NetBIOS access list named *marketing*.

```
interface token 1
netbios input-access-filter bytes marketing
```

Specifying a Bytes Access List Filter on Outgoing Messages

To define an access list filter on outgoing messages, use the **netbios output-access-filter bytes** interface subcommand. The full command syntax follows:

netbios output-access-filter bytes *name*
no netbios output-access-filter bytes *name*

The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list bytes** global configuration commands.

Use the **no netbios output-access-filter bytes** command to remove the entire access list.

Example:

These commands filter packets leaving Token Ring unit 1 using the NetBIOS access list named *engineering*.

```
interface token 1
netbios output-access-filter bytes engineering
```

Combining Administrative Filters with Boolean Access Expressions

The boolean access expression functionality allows you to combine access filters in new ways for Token Rings. With these access expressions, you may now indicate complex conditions under which bridged frames may enter or leave an interface. With these expressions, you can achieve levels of control on the forwarding of frames that would otherwise be impossible when using only the simple access expressions.

Access expressions are constructed from elemental access lists that define administrative filtering for the following fields in the packets:

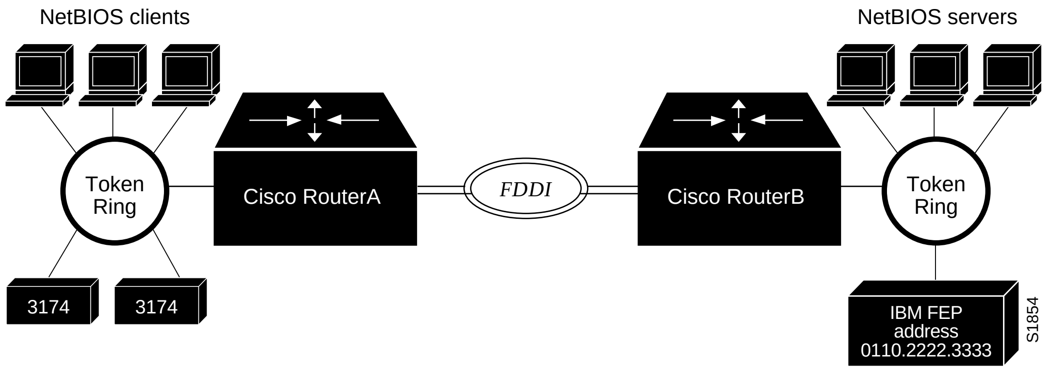
- LSAP and SNAP type codes
- MAC addresses
- NetBIOS station names
- NetBIOS arbitrary byte values

Note: For any given router interface, an access expression is mutually exclusive of the above elemental access lists for a given direction. For example, if an input access list is defined for MAC addresses on an interface, no access expression can be specified for the input side of that interface.

Access expressions take existing configured access lists and combine them in ways to meet complex access control needs.The following is a typical example that shows the usefulness of access expressions.

In this example network, you have a Token Ring interface on a Cisco router connected to an FDDI backbone, which has a second router attached to it in a similar setup, as shown in Figure 1-16:

Figure 1-16 Access Expression Example



On both Token Rings, you have to support both SNA and NetBIOS bridging. On the Token Ring attached to Cisco RouterA, you would like to allow the NetBIOS clients to talk to any of the NetBIOS servers off of RouterB, or any other, unpictured router. However, you would like to limit the 3174s to only talk to the one FEP off of RouterB, at MAC address 0110.2222.3333.

Without the use of access expressions, you could not achieve your desired result.A filter on RouterA which restricted access to only the FEP would have the side result of also restricting access of the NetBIOS clients to the FEP, something that is definitely not desired. What is needed is an access *expression* that would state “If it is a NetBIOS frame, pass through, but if it is an SNA frame, allow only access to address 0110.2222.3333.”

Configuring Access Expressions

Take the following steps to configure an access expression.

Step 1: Design the access expression (on paper).

Step 2: Design and configure in the router the individual access lists needed by the access expression.

Step 3: Configure the access expression itself into the router.

Designing the Access Expression

When designing an access expression, you want to come up with some phrase that indicates, in its entirety, all the frames that will *pass* the access expression. We will design this access expression to apply on frames coming from the Token Ring interface on RouterA.

For this example, you will want the access expression to say:

> "Pass the frame if it is NetBIOS, or if it is an SNA frame destined to address 0110.2222.3333."

This phrase should be written more in the form of a boolean expression, as in:

> "Pass if "NetBIOS or (SNA and destined to 0110.2222.3333).""

Designing and Configuring the Access Lists Used by the Expression

This statement given above requires three access lists to be designed and configured:

- An access list that passes a frame if it is a NetBIOS frame (SAP = 0xF0F0)
- An access list that passes a frame if it is an SNA frame (SAP=0x0404)
- An access list that passes a MAC address of 0110.2222.3333

The following configuration allows for all the conditions stated above:

```
! Access list 201 passes NetBIOS frames
access-list 201 permit 0xf0f0
!
! Access list 202 permits SNA frames
access-list 202 permit 0x0404
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

Configuring the Access Expression into the Router

The format of the per-interface subcommand to define an access expression is:

access-expression {**in**|**out**} *expression*
no access-expression {**in**|**out**} *expression*

One of **in** or **out** is specified to indicate whether the access expression is applied to packets entering or leaving this interface. You may specify both an input and an output access expression for an interface, but only one of each. The argument *expression* is the boolean access list expression, built as shown below. The **no** version of the command removes the access expression from the given interface.

An access expression consists of a list of terms, separated by boolean operators, and optionally grouped in parentheses.

An access expression term specifies a type of access list, followed by its name or number. The result of the term is either true or false, depending on if the access list specified in the term permits or denies the frame. The term itself can be one of the following as specified in Table 1-4:

Table 1-4 Access Expression Terms

Access Expression Term	Definition
lsap(2nn)	Specifies the LSAP access list nnn to be evaluated for this frame.
type(2nn)	Specifies the SNAP type access list to be evaluated for this frame.
smac(7nn)	Specifies an access list to match the source MAC address of the frame.
dmac(7nn)	Specifies an access list to match the destination MAC address of the frame.
netbios-host(name)	Specifies a netbios-host access list to be applied on NetBIOS frames traversing the interface.
netbios-bytes(name)	Specifies a netbios-bytes access list to be applied on NetBIOS frames traversing the interface.

Note: The netbios-host and netbios-bytes access expression terms will always return FALSE for frames that are not NetBIOS frames.

The terms are separated by boolean operators as listed in Table 1-5.

Table 1-5 Boolean Operators for Access Expression Terms

Boolean Operators	Definitions
~ (called "not")	Negates, or reverses, the result of the term or group of terms immediately to the right of the ~. Example: "~lsap(201)" returns FALSE if "lsap(201)" itself were TRUE.
& (called "and")	Returns TRUE if the terms or parenthetical expressions to the left and right of the **&** both return TRUE. Example: "lsap(201) & dmac(701)" returns TRUE if both the lsap(201) and dmac(701) terms return TRUE.
| (called "or")	Returns TRUE if the term or parenthetical expression to the left or right of the **|** either or both return TRUE. Example: "lsap(201) | dmac(701)" returns TRUE if either the lsap(201) or dmac(701) terms return TRUE.

Terms may be grouped in parenthetical expressions. Any of the terms and operators may be placed in parentheses, similar to what is done in arithmetic expressions, to affect order of evaluation.

Note: The incorrect use of parentheses can drastically affect the result of an operation, as the expression is read LEFT to RIGHT.

Example:

In math, you have:

3 * 4 + 2 = 14 but 3 * (4 + 2) = 18

Similarly, the following access expressions would return TRUE if lsap(201) and dmac(701) returned TRUE, or, if smac(702) returned TRUE:

lsap(201) & dmac(701) | smac(702)

However, the following access expression would return TRUE only if lsap(201) returned TRUE, and, either of dmac(701) or smac(702) returned TRUE:

lsap(201) & (dmac(701) | smac(702))

Going back to our example, we had the phrase:

"Pass the frame if it is NetBIOS, or if it is an SNA frame destined to address 0110.2222.3333."

This phrase was converted to the simpler form of:

Pass if “NetBIOS or (SNA and destined to 0110.2222.3333).”

So, for the following configuration:

```
! Access list 201 passes NetBIOS frames
access-list 201 permit 0xf0f0
!
! Access list 202 permits SNA frames
access-list 202 permit 0x0404
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

The resulting access expression would be:

```
access-expression in lsap(201) | (lsap(202) & dmac(701))
```

Therefore, the full configuration example is:

```
interface tokenring 0
access-expression in lsap(201 | (lsap(202) & dmac(701))
!
! Access list 201 passes NetBIOS frames
access-list 201 permit 0xf0f0
!
! Access list 202 permits SNA frames
access-list 202 permit 0x0404
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

Optimizing Access Expressions

It is possible to achieve the same result with more than one access expression. For example, if we wanted to allow not just NetBIOS and SNA to address 0110.2222.3333, but more generally, access for everything, but for the fact that SNA traffic was restricted to one host, the phrase could have been written as:

“Allow access if the frame is not an SNA frame, or if it is going to host 0110.2222.3333.”

More tersely this would be:

“Not SNA or destined to 0110.2222.3333.”

Using the access lists defined above, the resulting configuration would be:

```
interface tokenring 0
access-expression in ~lsap(202) | dmac(701)
!
! Access list 202 permits SNA frames
access-list 202 permit 0x0404
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

This is a better and simpler access list than the first, and will probably result in better run-time execution as a result. Therefore, it is best to work with your access expressions before configuring them into the router. Try to make them as simple as possible.

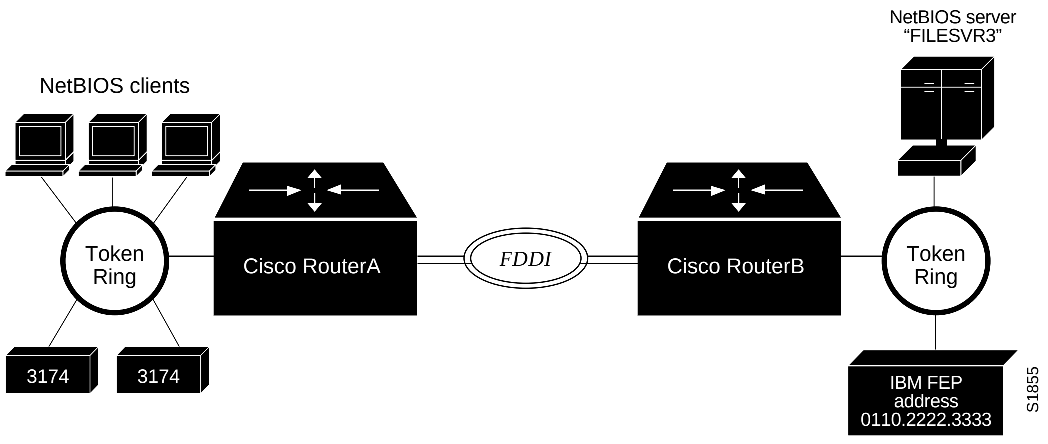
An Example Using NetBIOS Access Filters

Assume you want to permit access for the IBM 3174s to the FEP at address 0110.2222.3333, and also want to permit access by the NetBIOS clients to the NetBIOS server named FILESVR3. The following set of router configuration commands would meet this need:

```
netbios access-list host MIS permit FILESVR3
netbios access-list host MIS deny *
!
access-list 202 permit 0x0404
!
access-list 701 permit 0110.2222.3333
!
interface tokenring 0
access-expression in (lsap(202) & dmac(701)) | netbios-host(MIS)
```

Figure 1-17 illustrates this network configuration.

Figure 1-17 Network Configuration Using NetBIOS Access Filters



Altering Access Lists Used in Access Expressions

As access expressions are composed of access lists, special care must be taken when deleting and adding access lists that are referenced in these access expressions.

If an access list that is referenced in an access expression is deleted, the access expression merely ignores the deleted access list. However, if you want to redefine an access list by first deleting it and then adding a new one with the same name or number, you must let the access expression know about the change by redefining the access expression. This redefinition is necessary in order for the new access list to take effect.

As an example, if you want to redefine the NetBIOS access list named MIS that was used in the preceding example, the following sequence of configuration commands should be entered.

```
!  Replace the NetBIOS access list
!
no netbios access-list host MIS
netbios access-list host MIS permit FILESVR4
netbios access-list host MIS deny *
!
!  Replace the access-expression
!
interface tokenring 0
no access-expression in
access-expression in (lsap(202) & dmac(701)) | netbios-host(MIS)
```

Interoperability Issues

This section describes known interoperability issues between Cisco router/bridges and specific Token Ring implementations.

PC/3270 Emulation Software

The IBM PC/3270 emulation program Version 3.0 does not properly send packets over a Cisco source-route bridge.

Note: This problem exists only when using the older 4 Mb (CSC-R) Token Ring card.

The Cisco implementation confuses the IBM implementation into not looking beyond the local ring for the remote host.

The interface subcommand **source-bridge old-sna** rewrites the RIF headers of explorer packets send by the PC/3270 emulation program to go beyond the local ring. The **no source-bridge old-sna** command disables this compatibility mode.

source-bridge old-sna
no source-bridge old-sna

Examples:

These commands enable RIF rewriting.

```
interface tokenring 0
source-bridge old-sna
```

These commands disable RIF rewriting.

```
interface tokenring 0
no source-bridge old-sna
```

TI MAC Firmware

There is a known defect in earlier versions of the Texas Instruments' (TI) Token Ring MAC firmware. This implementation is used by Proteon, Apollo, and IBM RTs. A host using a MAC address whose first two bytes are zeros (such as a Cisco router/bridge) will not properly communicate with hosts using that version of TI firmware.

Cisco provides two solutions. The first involves installing a static RIF entry for every faulty node with which the router communicates. If there are many such nodes on the ring this may not be practical. The second solution involves setting the MAC address of the Cisco Token Ring to a value that works around the problem.

The interface subcommand **mac-address** sets the MAC layer address, and has this syntax:

mac-address *ieee-address*

The argument *ieee-address* is a 48-bit IEEE MAC address written as a dotted triple of four-digit hexadecimal numbers.

This command forces the use of a different MAC address on the specified interface, thereby avoiding the TI MAC firmware problem. It is up the network administrator to ensure that no other host on the network is using that MAC address.

Example:

This example command sets the MAC layer address where *xx.xxxx* is an appropriate second half of the MAC address to use.

```
interface tokenring 0
mac-address 5000.5axx.xxxx
```

Spurious Frame-Copied Errors

An IBM 3174 controller can be configured to report frame-copied errors to IBM LAN Network Manager software. These errors indicate that another host is responding to the MAC address of the 3174 controller. If a Cisco router/bridge is present on the same ring configured for source-route bridging, however, then the likely problem is the 3174 noticing that the Cisco router/bridge is setting the Address Recognized and Frame Copied bits. There is not a problem, merely a warning. No data is being lost.

Both the 3174 and the IBM LAN Network Manager software can be configure to ignore frame-copied errors.

Note: This problem exists only when using the older 4 Mb (CSC-R) Token Ring card.

Cisco's Implementation of LAN Network Manager

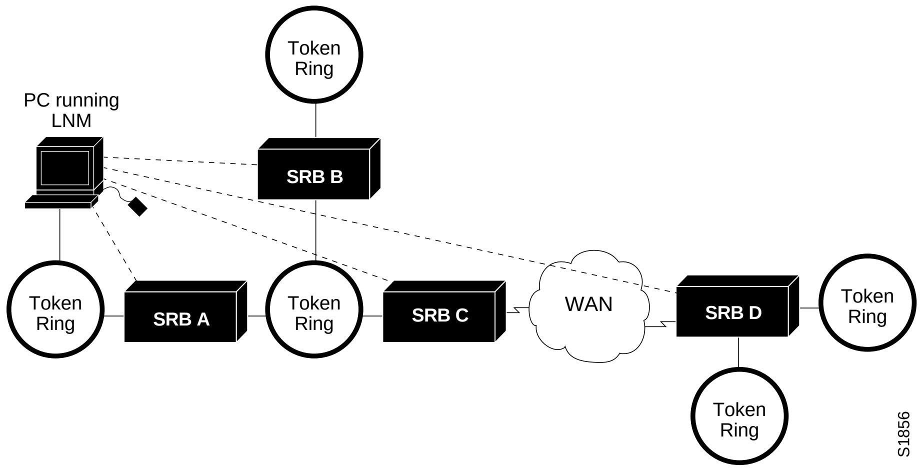
LAN Network Manager (LNM), formerly called LAN Manager, is an IBM product for managing a collection of source-route bridges. A source-route bridge connects multiple physical Token Rings into one logical network segment. LNM provides access to services so that you can monitor the entire source-route bridge environment through the use of a proprietary protocol. LNM gives the ability to manage the configuration of source-route bridges, monitor Token Ring errors, and gather information from Token Ring parameter servers.

Note: LNM is supported on the CSC-R16 or CSC-R16M Token Ring interface cards running at least SBEMON 3.0. LNM support is not provided on CSC-R or CSC-R16 cards with SBEMON 2.0.

LNM is not limited to managing locally attached Token Ring networks; it can also manage all other rings in the source-route bridged network. To accomplish this task, LNM works in conjunction with the IBM Bridge Program to manage all other rings in the bridged network. The IBM Bridge Program gathers data about the local Token Ring network and relays it back to LNM. In this manner, the bridge program becomes a proxy for information about its local Token Ring. Without this ability, you would require direct access to a device on every Token Ring in the network. This process would make managing a source-route bridged environment awkward and cumbersome.

Figure 1-18 shows some rings attached through a cloud and one LNM linking to a source-route bridge on each local ring.

Figure 1-18 LNM Linking to a Source-Route Bridge on Each Local Ring



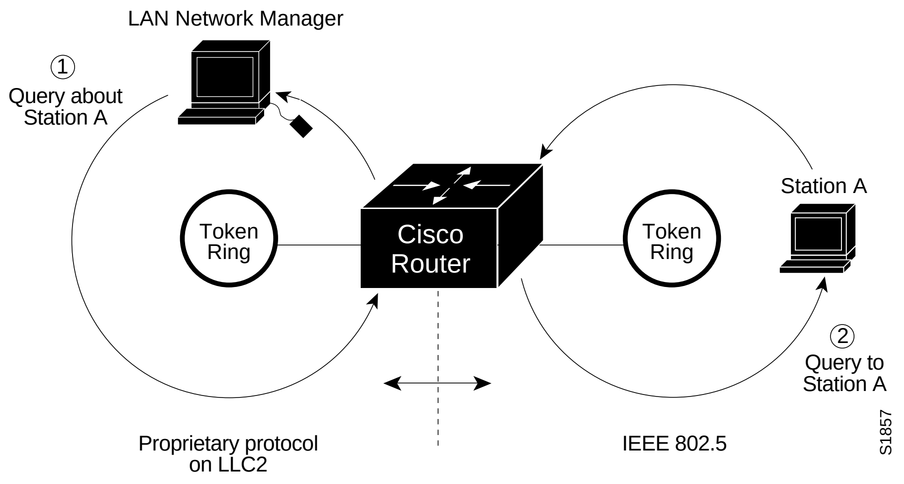
If LNM requires information about a station somewhere on a Token Ring, it sends a query to one of the source-route bridges connected to that ring using an IBM proprietary protocol. If the bridge can provide the requested information, it simply responds directly to LNM. If the bridge does not have the necessary information, it queries the station, using a protocol published in the IEEE 802.5 specification. In either case, the bridge sends a valid response back to LNM, using the proprietary protocol.

A good analogy to this would be a language translator who sits between a French speaking diplomat and a German speaking diplomat. If the French diplomat asks the translator a question in French for the German diplomat and the translator knows the answer, he or she simply responds without translating the original question into German. If the French diplomat asks a question the translator does not know how to answer, he or she must first translate the question to German, wait for the German diplomat to answer, and then translate the answer back to French.

Relating this back to LNM, if LNM asks a question of a source-route bridge in the proprietary protocol and the bridge knows the answer, it responds directly using the same protocol. If the bridge does not know the answer, it must first translate the question to the IEEE 802.5 protocol, ask the question to the station on the ring, and then translate the response back to the proprietary protocol to send to LNM.

Figure 1-19 illustrates requests from the LNM originating in an IBM proprietary protocol, and then being translated into IEEE 802.5 MAC-level frames.

Figure 1-19 LAN Network Manager Monitoring and Translating



Note that the proprietary protocol used by LNM to communicate with the source-route bridge is a nonroutable protocol running over a Logical Link Control (LLC2) connection. Although its protocol cannot be routed, LNM can monitor or manage anything within the source-route bridged (SRB) network.

How Cisco Routers Work with LAN Network Manager

As of Software Release 9.0, Cisco routers using 4/16 Mbps Token Ring interfaces configured for SRB support the proprietary protocol used by LNM and the source-route bridge. These routers can provide all of the functions currently provided by the IBM Bridge Program. Thus LNM can communicate to a router as if it were an IBM source-route bridge, such as the IBM 8209, and can manage or monitor any Token Ring connected to the Cisco router.

Through IBM Bridge Support, LNM provides three basic services for the source-route bridged network:

- The Configuration Report Server (CRS) keeps track of the current logical configuration of a Token Ring, and reports any changes to LNM. CRS also reports various other events, such as the change of an Active Monitor on a Token Ring.
- The Ring Error Monitor (REM) monitors errors reported by any station on the ring. In addition, REM monitors whether the ring is in a functional state or a failure state.
- The Ring Parameter Server (RPS) reports to LNM when any new station joins a Token Ring, and ensures that all stations on a ring are using a consistent set of reporting parameters.

Another capability of the IBM Bridge Support for LNM is asynchronous notification of some events that can occur on a Token Ring. Examples of these events include notification of a new station joining the Token Ring, or of the ring entering failure mode, known as *beaconing*. Support is also provided for LNM to change the operating parameters in the bridge. For a complete description of LAN Network Manager, please refer to the product manual supplied by IBM with the LAN Network Manager program.

Configuring Cisco routers to perform the functions of an IBM Bridge for communication with LNM occurs automatically when source-route bridging is enabled on the router. There are several options for modifying the behavior of the LNM support, but none are required for basic functionality. For example, because users can now modify the operation of the router through the Simple Network Management Protocol (SNMP) as well as through LNM, there is an option to exclude a user from modifying the router configuration through LNM. You can also specify which of the three LNM services (CRS, REM, RPS) the source-route bridge will perform.

Once the LNM support is running, use the commands described in the following sections to observe the configuration of the LNM bridge and its operating parameters. These commands show in detail the status of the LNM support within the Cisco source-route bridge.

LNM support in the Cisco source-route bridges is a powerful tool for you in managing source-route bridged networks. Through the ability to communicate with LNM and to provide the functionality of the IBM Bridge Program, the Cisco device appears as part of the IBM network. You, therefore, gain from the interconnectivity of Cisco products, without having to learn a new management product or interface.

Configuring the Router for LAN Network Manager Support

Configuring a Cisco router/bridge for LNM support is very simple. It happens automatically as a part of configuring the router to act as a source-route bridge. There are several commands available to modify the behavior of the LNM support, but none of them are necessary for it to function.

Because there is now more than one way to remotely change parameters in a Cisco router, some way was needed to prevent them from detrimentally interacting with each other. The global configuration command **lnm snmp-only** will prevent any LNM stations from modifying parameters in the Cisco router. It does not affect the ability of LNM to monitor events, only to change things. The syntax of this global command is:

lnm snmp-only
no lnm snmp-only

LNM has a concept of reporting links, and reporting link numbers. A reporting link is simply a connection (or potential connection) between a LAN Reporting Manager (LRM) and a Bridge. A reporting link number is a unique number used to identify a reporting link. An IBM bridge allows four simultaneous reporting links, numbered 0 through 3. Only the LRM attached on the lowest number connection is allowed to change any parameters, and then only when that connection number falls below a certain configurable number. In the default configuration, the LRM connected through link 0 is the only LRM allowed to change parameters. The interface subcommand to change this parameter is **lnm alternate**. The syntax for this interface subcommand is:

lnm alternate *number*
no lnm alternate

The *number* parameter must be an integer between 0 and 3. Setting *number* to 0 is the same as using the **no** version of the command, and is the default configuration. Setting this parameter on one interface in a source-route bridge will cause it to appear on the other interface of the bridge. This is because the command applies to the bridge itself, and not to either of the interfaces.

Each reporting link has its own password. This is used not only to prevent unauthorized access from an LRM to a bridge, but also to control access to the different reporting links. This is important because of the different abilities associated with the various reporting links. The syntax of this interface subcommand is:

lnm password *number string*
no lnm password *number*

The *number* parameter identifies which reporting link to apply the password command. In order to maintain compatibility with LNM, the parameter *string* should be a six to eight character string containing:

- Only letters
- Only numbers
- The characters @, #, $, or %

The **no** version of this command returns the password to its default value of 00000000.

Passwords are displayed only through use of the privileged-level **write terminal** command.

Note: There are two parameters in an IBM bridge that have no corresponding parameter in the Cisco router. This means that any attempt to modify these parameters from LNM will fail with an error message. The LNM names of these two parameters are the *route active status* and the *single route broadcast mode.*

Configuring Servers

As in an IBM bridge, the Cisco router provides several functions that gather information from a local Token Ring. All of these functions are enabled by default, but may be disabled by a configuration command. The syntax of these interface subcommands is as follows:

lnm crs
no lnm crs

lnm rem
no lnm rem

lnm rps
no lnm rps

The first of these services is the Configuration Report Server (CRS). This service keeps track of the current logical configuration of a Token Ring, and reports any changes to LNM. It also reports on various other activities such as the change of the Active Monitor on a Token Ring.

The second service is the Ring Error Monitor (REM). This services monitors errors reported by any station on the ring. It also monitors whether the ring is in a functional state, or in a failure state.

The third service is the Ring Parameter Server (RPS). This service ensures that all stations on a ring are using a consistent set of reporting parameters, and are reporting to LNM when any new station joins a Token Ring.

Changing Reporting Timers

The router sends a message to all attached LNMs whenever it begins to drop frames. The threshold where this report is generated is a percentage of the number of frames dropped compared to the number of frames forwarded. This threshold is configurable, and defaults to a value of 0.10%. The threshold is entered as a single number expressing the percentage loss rate in hundredths of a percent. The valid range is 0 to 9999. The syntax of this interface subcommand is as follows:

lnm loss-threshold *number*
no lnm loss-threshold

All stations on a Token Ring notify the Ring Error Monitor when they detect errors on the ring. In order to prevent excessive messages, error reports are not sent immediately, but are accumulated for a short period of time and then reported. A station learns this value from a Cisco router (configured as a source-route bridge) when it first enters the ring. This value is entered as the time interval in tens of milliseconds between error messages. The negative form of the command restores the timer value to its default of 200, or 2 seconds. The valid range is 0 to 65535. The syntax of this interface subcommand is as follows:

lnm softerr *number*
no lnm softerr

Configuring LAN Network Manager

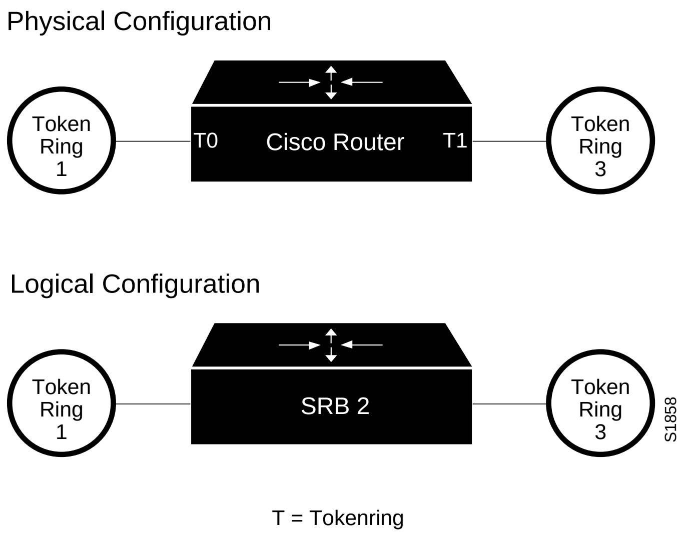
Configuring LNM is a fairly simple task, and is well covered in the LNM documentation. What may not be so simple is figuring out what information to enter into LNM, especially when there is a virtual ring involved. The basic problem extends from the fact that LNM is designed to only understand the concept of a two-port bridge, and the Cisco router with a virtual ring is a *multiport* bridge. The solution is to configure a virtual ring into LNM as a series of two-port bridges. The EXEC command **show lnm config** provides all the information necessary to configure these bridges into LNM.

For example, you have a router with two Token Rings configured as a local source-route bridge. The configuration file is:

```
interface TokenRing 0
source-bridge 1 2 3
!
interface TokenRing 1
source-bridge 3 2 1
```

Figure 1-20 illustrates the physical configuration and the logical configuration.

Figure 1-20 Cisco Router with Two Token Rings Configured as a Local Source-Route Bridge



The **show lnm config** command displays the logical configuration of this bridge, including all the pertinent information for configuring this router into LNM.

Example:

```
Doomgiver#show lnm config

Bridge(s) currently configured:
From     ring 001, address 0000.3000.abc4
Across bridge 002
To       ring 003, address 0000.3000.5735
```

For a more complex example, you have a router with three Token Rings configured as a multiport bridge. The configuration file appears as follows:

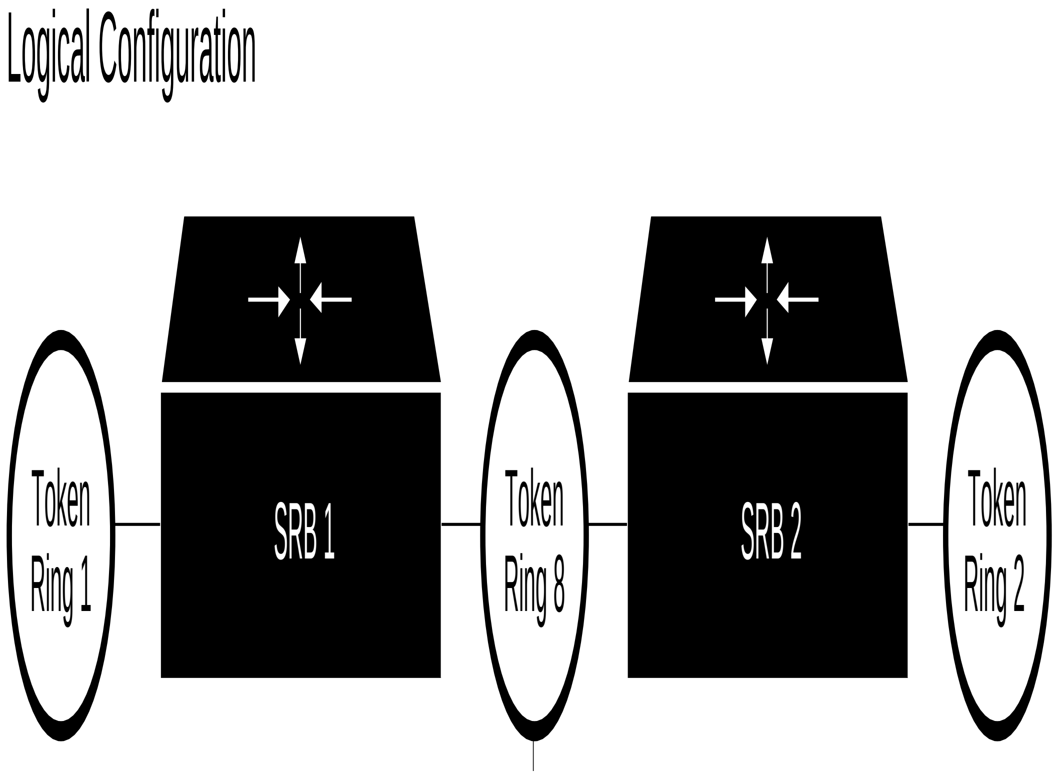
Example:

```
source-bridge ring-group 8
!
interface TokenRing 0
source-bridge 1 1 8
!
interface TokenRing 1
source-bridge 2 2 8
!
interface TokenRing 2
```

```
source-bridge 3 3 8
```

Figure 1-21 illustrates the physical and logical configuration.

Figure 1-21 Cisco Router with Three Token Rings Configured as a Multiport Bridge



The **show lnm config** command displays the logical configuration of this bridge, including all the pertinent information for configuring this router into LNM.

Example:

```
Doomgiver#show lnm config
Bridge(s) currently configured:

            From     ring 001, address 0000.0028.abcd
            Across bridge 001
            To       ring 008, address 4000.0028.abcd

            From     ring 002, address 0000.3000.abc4
            Across bridge 002
            To       ring 008, address 4000.3000.abc4

            From     ring 003, address 0000.3000.5735
            Across bridge 003
            To       ring 008, address 4000.3000.5735
```

Monitoring LNM Support

This section describes how you can monitor the LNM support.

Displaying Bridge Information

The command **show lnm bridge** displays all currently configured bridges, and all parameters that are related to the bridge as a whole, and not to one of its interfaces. Enter this command at the EXEC prompt:

show lnm bridge

The output of this command looks like the following.

```
Bridge 001-2-003, Ports 0000.3000.abc4, 0000.0028.abcd
Active Links:  0000.0000.0000  0000.0000.0000  0000.0000.0000
0000.0000.0000
Notification: 0 min, Threshold 00.10%
```

The above display's fields are described in Table 1-6.

Table 1-6 Show LNM Bridge Field Descriptions

Field	Description
Bridge	Indicates the ring and bridge numbers of this bridge.
Ports	Shows the MAC addresses of the two interfaces of this bridge.
Active Links	Shows any LNM stations that are currently connected to this bridge. An entry preceded by an asterisk is the controlling LNM.
Notification	The current counter notification interval in minutes.
Threshold	The current loss threshold that will trigger a message to LNM.

Displaying LNM Configuration Information

The command **show lnm config** displays the logical configuration of all bridges configured in a router. This information is needed to properly configure LNM for a Cisco router. This is especially important when using a virtual ring. Enter this command at the EXEC prompt:

show lnm config

The output of this command looks like the following.

```
Bridge(s) currently configured:

            From     ring 001, address 0000.3000.abc4
            Across bridge 002
            To       ring 003, address 0000.0028.abcd
```

The above display's fields are described in Table 1-7.

Table 1-7 Show LNM Configuration Field Descriptions

Field	Description
From	Indicates the ring number and MAC address of the first interface in the two-port bridge.
Across	Indicates the bridge number assigned to this bridge.
To	Indicates the ring number and MAC address of the second interface in the two-port bridge.

Displaying Token Ring Information

The command **show lnm interface** displays all LNM relevant information about a specific interface, or about all interfaces. If a specific interface is requested, it also displays a list of all currently active stations on that interface. Enter this command at the EXEC prompt:

show lnm interface *interface*

The output of this command looks like the following:

```
                                                  nonisolating error counts
interface   ring   Active Monitor   SET    dec   lost  cong.   fc  freq.
token

TokenRing1  0001*  1000.5a98.23a0  00200  00001  00000 00000 00000 00000
00002

Notification flags: FE00, Ring Intensive: FFFF, Auto Intensive: FFFF
Active Servers: LRM LBS REM RPS CRS
Last NNIN:   never, from 0000.0000.0000.
Last Claim:  never, from 0000.0000.0000.
Last Purge:  never, from 0000.0000.0000.
Last Beacon: never, 'none' from 0000.0000.0000.
Last MonErr: never, 'none' from 0000.0000.0000.

                                                    isolating error counts
    station     int  ring  loc.   weight   line  inter burst   ac  abort
1000.5a98.23a0   T1  0001  0000   00 - N   00000 00000 00000 00000 00000
1000.5a98.239e   T1  0001  0000   00 - N   00000 00000 00000 00000 00000
1000.5a6f.bc15   T1  0001  0000   00 - N   00000 00000 00000 00000 00000
0000.3000.abc4   T1  0001  0000   00 - N   00000 00000 00000 00000 00000
1000.5a98.239f   T1  0001  0000   00 - N   00000 00000 00000 00000 00000
```

The above display's fields are described in Table 1-8.

Table 1-8 Show LNM Interface Field Descriptions

Field	Description
interface	Interface about which information was requested.
ring	Number assigned to that Token Ring.
an asterisk	Indicates that there are stations with nonzero error counters present on this ring.

Field	Description
Active Monitor	Address of the station that is currently providing "Active Monitor" functions to the ring.
SET	Current soft error reporting time for the ring in units of tens of milliseconds.
nonisolating error counts: dec	Rate at which the various counters of nonisolating error are being decreased. This number is in errors per 30 seconds.
other nonisolating error counts	Current values of the five nonisolating error counters specified in the 802.5 specification. These are Lost Frame errors, Receiver Congestion errors, FC errors, Frequency errors, and Token errors.
Notification Flags	Representation of which types of ring errors are being reported to LNM. The description of this number can be found in the *IBM Architecture Reference Manual*.
Ring Intensive	Representation of which specific ring error messages are being reported to LNM when in the "Ring Intensive" reporting mode. The description of this number can be found in the *IBM Architecture Reference Manual*.
Auto Intensive	Representation of which specific ring error messages are being reported to LNM when in the "Auto Intensive" reporting mode. The description of this number can be found in the *IBM Architecture Reference Manual*.
Active Servers	A list of which services are currently active on this Token Ring. The possible acronyms and their meanings are: CRS—Configuration Report Server LRM—LAN Reporting Manager LBS—LAN Bridge Server REM—Ring Error Monitor RPS—Ring Parameter Server
Last NNIN	Time since the last "Neighbor Notification Incomplete" frame was received, and the station that sent this message.
Last Claim	Time since the last "Claim Token" frame was received, and the station that sent this message.
Last Purge	Time since the last "Purge Ring" frame was received, and the station that sent this message.
Last Beacon	Time since the last "Beacon" frame was received, the type of the last beacon frame, and the station that sent this message.
Last Mon Err	Time since the last "Report Active Monitor Error" frame was received, the type of the last monitor error frame, and the station that sent this message.

* The description of these servers can be found in the *IBM Architecture Reference Manual*.

See the **show lnm station** command for a description of the fields in the bottom half of the command output.

The same exact information can be obtained by specifying a ring number assigned to a Token Ring instead of specifying the interface number. To do this you would enter this command at the EXEC prompt:

show lnm ring *number*

The output of this command is the same as the output of the **show lnm interface** command.

Displaying Station-Specific Information

The command **show lnm station** displays LNM-relevant information about a specific station, or about all known stations on all rings. If a specific station is requested, it also displays a detailed list of that station's current MAC-level parameters. Enter this command at the EXEC prompt:

show lnm station *address*

The output of this command looks like the following.

```
                                               isolating error counts
   station      int  ring  loc.   weight   line  inter burst   ac  abort
1000.5a6f.bc15   T1  0001  0000   00 - N   00000 00000 00000 00000 00000

Unique ID:  0000.0000.0000          NAUN: 0000.3000.abc4
Functional: C000.0000.0000         Group: C000.0000.0000
Physical Location:   00000        Enabled Classes:  0000
Allowed Priority:    00000        Address Modifier: 0000
Product ID:     00000000.00000000.00000000.00000000.0000
Ucode Level:    00000000.00000000.0000
Station Status: 00000000.0000
Last transmit status: 00
```

Table 1-9 Show LNM StationField Descriptions

Field	Description
station	MAC address of the given station on the Token Ring.
int	Interface used to reach the given station.
ring	Number of the Token Ring where the given station is located.
loc	Physical location number of the given station.
weight	Weighted accumulation of the errors of the given station, and of its NAUN. The three possible letters and their meanings are:*
N	Not in a reported error condition.
P	In a “pre-weight” error condition.
W	In a “weight” error condition.
isolating error counts	Current values of the five isolating error counters specified in the 802.5 specification. These are Line errors, Internal errors, Burst errors, AC errors, and Abort errors.
Values below this point will be zero unless LNM has previously requested this information.	
Unique ID	Uniquely assigned value for this station.
NAUN	MAC address of this station’s “upstream” neighbor.
Functional	MAC-level functional address currently in use by this station.
Group	MAC-level group address currently in use by this station.
Physical Location	Number assigned to this station as its “Physical Location” identifier.
Enabled Classes	Functional classes that the station is allowed to transmit.
Allowed Priority	Maximum access priority that the station may use when transmitting onto the Token Ring.
Address Modifier	Reserved field.
Product ID	Encoded 18-byte string used to identify what hardware/software combination is running on this station.
Ucode Level	10-byte EBCDIC string indicating the microcode level of the station.
Station Status	Implementation-dependent vector that is not specified anywhere.
Last transmit status	Contains the strip status of the last “Report Transmit Forward” MAC frame forwarded by this interface.

* The description of these servers can be found in the *IBM Architecture Reference Manual.*

Debugging LNM Support

Use the following debugging commands to debug LNM. For each of the following **debug** commands there is a corresponding **undebug** command that stops the output.

debug lnm-events

The **debug lnm-events** command displays any unusual events that occur on a Token Ring. This includes such things as stations reporting errors, error thresholds being exceeded, and so on.

debug lnm-llc

The **debug lnm-llc** command displays all communication between the Cisco router/bridge and the LNMs that have connections to it. One line is displayed for each message sent or received.

debug lnm-mac

The **debug lnm-mac** command displays all management communication between the Cisco router/bridge and all stations on the local Token Rings. One line is displayed for each message sent or received.

Source-Route Bridging Configuration Examples

The Token Ring system software is written such that a minimum of configuration is the normal case. A Token Ring equipped router is by default a single-ring host. Source bridging is off by default. Following are configuration examples that you may use to make your own configuration files. Refer to previous illustrations in this chapter for a visual orientation of the networks illustrated here.

Basic Token Ring Configuration

This example configures the Cisco router/bridge for IP and Novell IPX routing.

Example:

```
novell routing
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
novell network 32
multiring all
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
```

```
novell network 461
multiring all
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
novell network 95
```

Basic Token Ring Source Bridge Configuration

In this partial configuration, the source bridge is turned on, IP is routed, and all other protocols are bridged.

Example:

```
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
```

Source Bridge Only Configuration

In this partial configuration, all protocols are bridged, including IP. Because IP is being bridged, the system has only one IP address.

Example:

```
no ip routing
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
!
interface TokenRing 1
ip address 131.108.129.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
!
interface Ethernet 0
ip address 131.108.129.2 255.255.255.0
```

Other Protocols Routed at the Same Time

In this configuration, IP, XNS, and Novell are all being routed while all others will be bridged between rings. While not strictly necessary, the Novell and XNS network numbers are set consistently with the IP subnetwork numbers. This makes the network easier to maintain.

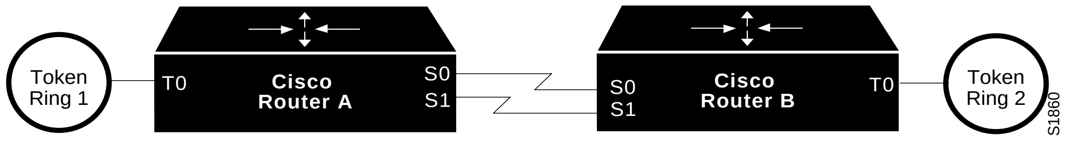
Example:

```
xns routing 0000.0C00.02C3
!
novell routing 0000.0C00.02C3
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
xns network 129
novell network 129
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
xns network 130
novell network 130
source-bridge 130 1 129
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
xns network 2
novell network 2
```

Remote Source-Route Bridge with Simple Reliability

In the following sample, the basic two-port remote source-route bridge configuration is extended to include both reliability and load sharing. The two routers are connected by two serial lines. When both serial lines are up, traffic is split between them, effectively combining the bandwidth of the connections. If either one of the serial lines goes down, all traffic is routed to the remaining line with no disruption. This happens transparently with respect to the end connections, unlike other source-route bridges that would abort those connections. This configuration is shown in Figure 1-22.

Figure 1-22 Remote Source-Route Bridge—Simple Reliability



Configuration for Router/Bridge A:

```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 204.31.7.1
source-bridge remote-peer 5 tcp 204.31.8.1
!
interface TokenRing 0
ip address 204.31.7.1 255.255.255.0
source-bridge 1 1 5
source-bridge spanning
multiring all
!
interface Serial 0
ip address 204.31.9.1 255.255.255.0
!
interface Serial 1
ip address 204.31.10.1 255.255.255.0
!
router igrp 109
network 204.31.7.0
network 204.31.9.0
network 204.31.10.0
!
hostname RouterA
```

Configuration for Router/Bridge B:

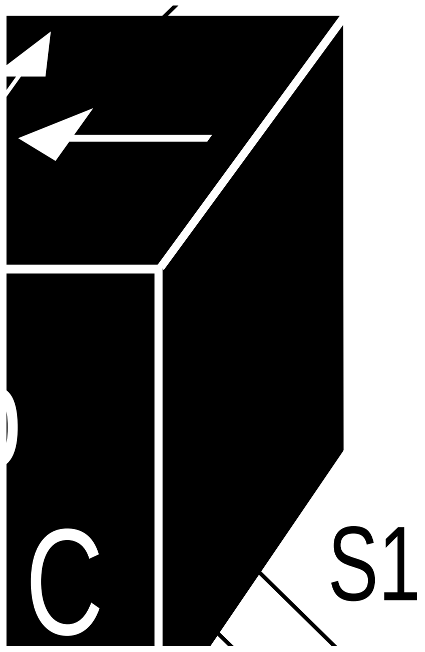
```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 204.31.7.1
source-bridge remote-peer 5 tcp 204.31.8.1
!
interface TokenRing 0
ip address 204.31.8.1 255.255.255.0
source-bridge 2 1 5
source-bridge spanning
multiring all
!
interface Serial 0
ip address 204.31.9.2 255.255.255.0
!
interface Serial 1
ip address 204.31.10.2 255.255.255.0
!
router igrp 109
network 204.31.8.0
network 204.31.9.0
network 204.31.10.0
!
hostname RouterB
```

Remote Source-Route Bridge—Complex Configuration, with Local Acknowledgment

In the following example, a triangular configuration is used to provide the maximum reliability with minimal cost. In addition, one of the links is doubled to gain better bandwidth. In addition to IP and source-route traffic, AppleTalk is also being routed between all the sites. Note that, in this configuration, all the sessions between Router C and Router D are locally acknowledged. All the sessions between Router C and Router E are not locally acknowledged and are configured for normal RSRB. This example shows that not every peer must be locally acknowledged, but rather, that Local Acknowledgment can be turned on or off at the customer's discretion.

This configuration is represented in Figure 1-23.

Figure 1-23 Remote Source-Route Bridge with Local Acknowledgment—Complex Configuration



Configuration for Router/Bridge C:

```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
source-bridge remote-peer 5 tcp 132.21.2.6 local-ack
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.1.1 255.255.255.0
source-bridge 1 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.4.25 255.255.255.0
appletalk address 4.25
appletalk zone Twilight
```

```
!
interface Serial 0
ip address 132.21.16.1 255.255.255.0
appletalk address 16.1
appletalk zone Twilight
!
interface Serial 1
ip address 132.21.17.1 255.255.255.0
appletalk address 17.1
appletalk zone Twilight
!
interface Serial 2
ip address 132.21.18.1 255.255.255.0
appletalk address 18.1
appletalk zone Twilight
!
router igrp 109
network 132.21.0.0
!
hostname RouterC
```

Configuration for Router/Bridge D:

```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1 local-ack
source-bridge remote-peer 5 tcp 132.21.2.6
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.2.6 255.255.255.0
source-bridge 2 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.5.1 255.255.255.0
appletalk address 5.1
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.16.2 255.255.255.0
appletalk address 16.2
appletalk zone Twilight
!
interface Serial 1
ip address 132.21.19.1 255.255.255.0
appletalk address 19.1
appletalk zone Twilight
!
router igrp 109
network 132.21.0.0
!
hostname RouterD
```

Configuration for Router/Bridge E:

```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
source-bridge remote-peer 5 tcp 132.21.2.6
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.10.200 255.255.255.0
source-bridge 10 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.7.1 255.255.255.0
appletalk address 7.1
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.19.2 255.255.255.0
appletalk address 19.2
appletalk zone Twilight
!
interface Serial 1
ip address 132.21.17.2 255.255.255.0
appletalk address 17.2
appletalk zone Twilight
!
interface Serial 2
ip address 132.21.18.2 255.255.255.0
appletalk address 18.2
appletalk zone Twilight
!
router igrp 109
network 132.21.0.0
!
hostname RouterE
```

Maintaining the Source-Route Bridge

Use this EXEC command to maintain the source-route bridge cache.

clear rif-cache

The **clear rif-cache** command clears the entire RIF cache.

Monitoring the Source-Route Bridge

Use the EXEC commands described in this section to obtain displays of activity on the source-route bridge.

Displaying the RIF Cache

The **show rif** EXEC command displays the current contents of the RIF cache. Enter this command at the EXEC prompt:

show rif

The following is a sample display of **show rif**:

```
Codes: * interface, - static, + remote
Hardware Addr  How   Idle (min)  Routing Information Field
5C02.0001.4322 rg5          -   0630.0053.00B0
5A00.0000.2333 TR0          3   08B0.0101.2201.0FF0
5B01.0000.4444 -            -   -
0000.1403.4800 TR1          0   -
0000.2805.4C00 TR0          *   -
0000.2807.4C00 TR1          *   -
0000.28A8.4800 TR0          0   -
0077.2201.0001 rg5         10   0830.0052.2201.0FF0
```

The following is sample output. The field descriptions are found in Table 1-10.

Table 1-10 RIF Cache Display Field Description

Field	Description
Hardware Addr	Lists the MAC-level addresses.
How	Describes how the RIF has been learned. Possible values include a ring group (rg), or interface (TR).
Idle	Indicates how long, in minutes, since the last response was received directly from this node.
Routing Information Field	Lists the RIF.

Entries marked with an asterisk (*) are the router/bridge's interface addresses. Entries marked with a dash (–) are static entries. Entries with a number denote cached entries. If the RIF timeout is set to something other than the default of 15 minutes, the timeout is displayed at the top of the display.

Displaying the Current Bridge Configuration

The **show source-bridge** EXEC command displays the current source bridge configuration and miscellaneous statistics. Enter this command at the EXEC prompt:

show source-bridge

The following is sample output. The field descriptions are found in Table 1-11.

```
wilma#show source-bridge

Local Interfaces:       max      receive            transmit
      srn bn  trn rg px sp hp     cnt:bytes         cnt:bytes      drops
  TR0   5  1  10  *     *  7      39:1002            23:629          23

Ring Group 10:
  This peer: TCP 150.136.92.92
   Maximum output TCP queue length, per peer: 100
  Peers:               state   vl  pkts_rx  pkts_tx  expl_gn    drops  
TCPq
   TCP 150.136.92.92     -     2        0       0        0       0    0
   TCP 150.136.93.93     open   2*     18      18        3       0    0
Rings:
   bn: 1 rn: 5    local  ma: 4000.3080.844b TokenRing0           fwd: 18
   bn: 1 rn: 2   remote  ma: 4000.3080.8473 TCP 150.136.93.93    fwd: 36

Explorers: ------- input -------             ------- output -------
       spanning  all-rings     total      spanning  all-rings     total
  TR0         0          3         3             3          5         8
wilma#
```

Table 1-11 Current Bridge Configuration Field Descriptions

Field	Description
Local Interfaces:	Description of local interfaces.
max	Maximum routing descriptor length.
receive	Packets: bytes received on interface for source bridging.
transmit	Packets: bytes transmitted on interface for source bridging.
srn	Ring number of this Token Ring.
bn	Bridge number of this router, for this ring.
trn	The group in which the interface is configured. (The target ring number, or virtual ring group.)
rg	Indicates ring group, noted by an asterisk (*).
px	Indicates an interface that can respond with proxy explorers, noted by an asterisk (*).
sp	Indicates a spanning explorer enabled on the interface, noted by an asterisk (*).
hp	Indicates hops.
Ring Group:	Describes the ring group.
This peer:	Address and address type of this peer.
Maximum output TCP queue length, per peer:	Maximum number of packets queued up on this peer before router starts dropping packets.
Peers:	Addresses and address types of the ring group peers.
state	Current state of the peer, open or closed. A hyphen indicates this router.

Field	Description
vl	Indicates version of remote source-route bridge. The l indicates Local Acknowledgment, noted by an asterisk (*).
pkts_rx	Lists the number of packets received.
pkts_tx	Lists the number of packets transmitted.
expl_gn	Lists the explorers generated.
drops	Lists the number of dropped packets.
TCP q	Lists the current TCP backup queue length.
Rings:	Describes the ring groups. Information displayed includes the bridge groups, ring groups, whether the group is local or remote, the MAC address, the network address or interface type, and the number of packets forwarded. A type shown as "locvrt" indicates a local virtual ring used by SDLLC or SR/TLB; a type shown as "remvrt" indicates a remote virtual ring used by SDLLC or SR/TLB.
Explorers:	
input	Explorers received by router.
output	Explorers generated by router.
TR0	The interface on which explorers were received.
spanning	Spanning-tree explorers.
all-rings	All-rings explored.
total	Summation of spanning and all-rings.

Displaying Information About the Token Ring Interface

The EXEC command **show controllers token** displays internal state information about the Token Ring interfaces in the system. Enter this command at the EXEC prompt:

show controllers token

The command displays the versions number of the Token Ring firmware, and the source-bridge capability of the interface. These statistics are most useful to Cisco staff for system troubleshooting.

Displaying Token Ring Interface Statistics

The **show interface**s EXEC command display for Token Rings provides high-level statistics about the state of source bridging for a particular interface. Enter this command at the EXEC prompt:

show interfaces

Refer to the section "Token Ring Interface Support" in the chapter "Configuring the Interfaces" for a description of the information this command displays.

Debugging the Source-Route Bridge

Use the privileged-level EXEC **debug** commands described in this section to track activity in the source-route bridge network. Generally, these commands will be executed when working with Cisco staff, to track system problems. For each **debug** command there is a corresponding **undebug** command that stops the output.

In order to display traffic source-routed through an interface, fast switching of SRB frames must first be disabled with the **no source-bridge route-cache** interface subcommand.

debug rif

The **debug rif** command provides informational displays for entries entering and leaving the RIF cache.

Descriptions of the messages that can be generated with **debug rif** follow.

```
RIF: L Sending XID for <address>
```

The router/bridge wanted to send a packet to <address> but did not find it in the RIF cache. It sent an XID explorer packet to determine which RIF it should use. The attempted packet is dropped.

```
RIF: L No buffer for XID to <address>
```

Similar to the previous display, however a buffer in which to build the XID packet could not be obtained.

```
RIF: U chk <address>[<rif>]
```

A packet is being checked to see if its MAC address is already in the RIF cache. <interface> is either the physical interface the packet arrived on, or the static/remote interface. <ring group>, a number, is only valid if this RIF check is for a remote packet. <code> denotes the kind of RIF entry being checked; this is an internal code and is not documented.

```
RIF: U remote rif too small [<rif>]
```

A packet's RIF was too short to be valid.

```
RIF: U rej <address>too big [<rif>]
```

A packet's RIF exceeded the maximum size allowed and was rejected. The maximum size is 18 bytes.

```
RIF: U upd interface <address>
```

The RIF entry for this router/bridge's interface has been updated.

```
RIF: U ign <address>interface update
```

A RIF entry that would have updated an interface corresponding to one of this routers interfaces.

```
RIF: U upd <address>[<rif>]
```

The RIF entry for <address> has been found and updated.

```
RIF: U add <address>[<rif>]
```

The RIF entry for <address> has been added to the RIF cache.

```
RIF: U no memory to add rif for <address>
```

No memory to add a RIF entry for <address>.

```
RIF: removing rif entry for <address>, type <code>
```

The RIF entry for <address> has been forcibly removed.

```
RIF: flushed <address>
```

The RIF entry for <address> has been removed because of a RIF cache flush.

```
RIF: expired <address>
```

The RIF entry for <address> has been aged out of the RIF cache.

```
RIF: rcvd XID response from <address>
```

An XID response from <address> was inserted into the RIF cache.

```
RIF: rcvd TEST response from <address>
```

A TEST response from <address> was inserted into the RIF cache.

debug source-bridge

The **debug source-bridge** command provides informational displays of source bridging activity.

Samples of messages displayed include the following:

In the following sample display, SRBn or RSRBn denotes a message associated with interface Token Ring *n*. An *n* of 99 denotes the remote side of the network.

```
SRBn: no path, s: <src MAC addr>d: <dst MAC addr>rif: <rif>
```

A bridgeable packet came in on interface Token Ring *n* but there was no where to send it. This is most likely a configuration error. For example, an interface has source bridging turned on but it is not connected to another source bridging interface or a ring group.

```
SRBn: direct forward (srn <ring>bn <bridge>trn <ring>)
```

A bridgeable packet has been forwarded from Token Ring *n* to the target ring. The two interfaces are directly linked.

```
SRBn: br dropped proxy XID,  <address>for <address>, wrong vring (rem)
SRBn: br dropped proxy TEST, <address>for <address>, wrong vring (rem)
SRBn: br dropped proxy XID,  <address>for <address>, wrong vring (local)
SRBn: br dropped proxy TEST, <address>for <address>, wrong vring (local)
SRBn: br dropped proxy XID,  <address>for <address>, no path
SRBn: br dropped proxy TEST, <address>for <address>, no path
```

A proxy explorer reply was not generated because there was no way to get there from this interface. The packet came from the node with the first <address>.

```
SRBn: br sent proxy XID,  <address>for <address>[<rif>]
```

```
SRBn: br sent proxy TEST, <address>for <address>[<rif>]
```

An appropriate proxy explorer reply was generated on behalf of the second <address>. It is sent to first <address>.

```
RSRB: sent RingXreq to <ring group>/<ip addr>
```

A Ring exchange request was sent to the indicated peer. This tells the remote side which rings this node has and requests a reply indicating which rings that side has.

```
RSRB: <label>: sent <op>to <ring group>/<ip addr>
```

A message has been sent to the indicated remote peer. Where <label> may be AHDR (active header), PHDR (passive header), HDR (normal header), or DATA (data exchange). Where <op> may be Forward, Explorer, Ring Xchg, Req, Ring Xchg, Rep, Unknown Ring Group, Unknown Peer, and Unknown Target Ring.

```
RSRB: removing bn <bridge>rn <ring>from <ring group>/<ip addr>
RSRB: added bridge <bridge>, ring <ring>for <ring group>/<ip addr>
```

The remote bridge and ring pair have been removed from or added to the local ring group table because the remote peer has changed.

```
RSRB: peer <ring group>/<ip addr>closed [last state n
RSRB: passive open <ip addr>(remote port) -><local port>
RSRB: CONN: opening peer <ring group>/<ip addr>, attempt n
RSRB: CONN: Remote closed <ring group>/<ip addr>on open
RSRB: CONN: peer <ring group>/<ip addr>open failed, <reason>[code]
```

Miscellaneous remote peer connection establishment messages.

```
RSRBn: sent local explorer, bridge <bridge>trn <ring>, [rif]
```

An explorer packet was propagated onto the local ring from the remote ring group.

```
RSRBn: ring group <ring group>not found
RSRBn: explorer rif [rif] not long enough
```

The remote source-route bridging code found the packet to be in error.

For each of the following **debug** commands there is a corresponding **undebug** command that stops the output. Enter these commands at the EXEC prompt.

debug source-event

The **debug source-event** command enables an interesting subset of the source-bridge debugging messages, including bridged packets rejected and remote peer connection activities.

debug token-event

The **debug token-event** command provides informational displays about significant Token Ring hardware events.

debug token-ring

The **debug token-ring** command invokes verbose Token Ring hardware debugging. This includes detailed displays as traffic arrives and departs the unit. Cisco Systems recommends using this feature only on router/bridges with light loads.

Source-Route Bridge Global Configuration Command Summary

Following is an alphabetical summary of the global configuration commands for source-route bridging.

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

Used to configure a vendor code access list. This is a global bridge **access-list** command for IEEE 802 address access lists.

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

Configures the access list mechanism for filtering frames by protocol type. The argument *list* is a user-selectable number in the range 200 – 299, inclusive, that identifies the list. The keyword **permit** permits the frame; the keyword **deny** denies the frame.

[**no**] **lnm snmp-only**

Prevents any LNM stations from modifying parameters in the Cisco router.

[**no**] **netbios access-list bytes** *name* {**permit**|**deny**} *offset pattern*

Defines the offset and patterns. The NetBIOS byte offset access list contains a series of offsets and hexadecimal patterns with which to match byte offsets in NetBIOS packets. The argument *name* is the name of the access list being defined. The argument *offset* is a decimal number indicating the number of bytes into the packet where the byte comparison should begin. An offset of zero points to the beginning of the NetBIOS delimiter string (0xffef) at the start of each NetBIOS packet. The argument *pattern* is a hexadecimal string of digits representing a byte pattern. The argument *pattern* must conform to certain conventions.

[**no**] **netbios access-list host** *name* {**permit**|**deny**} *pattern*

The NetBIOS station access list contains the station name with which to match, along with a permit or deny condition. Use the **netbios access-list host** global configuration command to assign the name of the access list to a station or set of stations on the network. The argument *name* is the name of the access list being defined. The argument *pattern* is a set of characters. The characters can be the name of the station, or a combination of characters and pattern matching symbols that establish a pattern for a set of NetBIOS station names.The **no** version removes an entire list, or just a single entry from a list, depending upon the argument given for *pattern*.

rif *MAC-address* [*RIF-string*] [*interface-name*|**ring-group** *ring*]
no rif *MAC-address* [*interface-name*|**ring-group** *ring*]

Inserts or removes an entry into the RIF cache. If *RIF-string* is present, it is checked for validity and used. Otherwise, the entry is flagged as having no RIF. An *interface-name* or appropriate ring-group *ring* may also be specified to indicate the direction from which this RIF entry would have arrived. The **no** version of this command removes an entry from the cache.

[**no**] **rif timeout** *minutes*

Determines the period of inactivity allowed before unused RIF cache entries are removed. The **no** version of the command resets the RIF timeout period to its default of 15 minutes.

[**no**] **source-bridge enable-80d5**

Enables or disables the conversion of frames between Token Ring LLC2 and Ethernet Type II *0x80d5 format*. When disabled, Token Ring LLC2 frames are converted to IEEE 802.3 format LLC2 frames. The list of frames undergoing this conversion may be customized by DSAP address with the **source-bridge sap-80d5** command. This command only applies to frames going through the Source-Route Translational Bridge (SR/TLB), and not to frames going across a Source-Route Transparent (SRT) bridge.

[**no**] **source-bridge largest-frame** *ring-group size*

Defines the largest frame size to communicate with all peers in the ring group.

[**no**] **source-bridge old-oui**

Causes the OUI code in Token Ring frames translated to and from Ethernet Type II to be 0x000000. The default OUI value, used when this command is not given, is 0x0000F8. This default value is compatible with the draft value used in various SRT protocol documents. The **no** version of the command restores the default value.

[**no**] **source-bridge passthrough** *ring-number*

Specifies that frames destined to *ring-number* should never be terminated with Local Acknowledgment. The **no** version allows frames to be terminated with Local Acknowledgment.

[**no**] **source-bridge remote-peer** *ring-group* **interface** *interface-name* [**lf** *size*] [**version** *number*]

Defines or removes a serial interface over which to run bridged Token Ring traffic. The keyword **lf** and the *size* argument define the largest frame size to communicate with all peers in the ring group.The keyword **version** specifies the forced RSRB protocol version number for the remote peer.

[**no**] **source-bridge remote-peer** *ring-group* **tcp** *ip-address* [**lf** *size*][**local-ack**] [**version** *number*]

Defines or removes a remote peer for the specified ring group. We would make a TCP connection to carry bridged Token Ring traffic. The keyword **lf** and the *size* argument define the largest frame size to communicate with all peers in the ring group. The keyword **local-ack** specifies that Local Acknowledgment should be used for LLC2 session going to this remote peer.The keyword **version** specifies the forced RSRB protocol version number for the remote peer.

[**no**] **source-bridge ring-group** *ring-number*

Establishes or removes a ring group.

[**no**] **source-bridge sap-80d5** *SAP*

When used in conjunction with the **source-bridge enable-80d5** command, enables or disables the translation of Token Ring LLC2 frames to Ethernet Type 2 80d5 format frames. Frames whose destination SAP is given with one of these commands will undergo the transformation into *80d5* frames when entering an Ethernet. If the **source-bridge enable-80d5** command is not given, this command has no effect. As many of these commands as are needed may be given, one SAP per line.

[**no**] **source-bridge tcp-queue-max** *number*

Sets the maximum output TCP queue length, in packets, that will be enqueued to remote source-route bridge peers. The **no** version of the command restores the default queue length of 100.

[**no**] **source-bridge transparent** *ring-group pseudo-ring bridge-number tb-group*

Establishes a Source-Route Translational Bridging (SR/TLB) link between the source-route bridged *ring-group* and the transparent bridge group *tb-group*. The *pseudo-ring* is the ring number associated with the transparent bridging domain, and *bridge-number* is the pseudo-bridge that attaches the *ring-group* to the *pseudo-ring*.

Source-Route Bridge Interface Subcommand Summary

Following is an alphabetical summary of the interface subcommands for source-route bridging.

[**no**] **access-expression** {**in**|**out**} *expression*

Defines an access expression for a given interface—for Token Ring only.

[**no**] **lnm alternate** *number*

Enables an LRM other than the default LRM to change parameters. The *number* parameter must be an integer between 0 and 3.

[**no**] **lnm crs**

Enables the Configuration Report Server (CRS), which keeps track of the current logical configuration of a Token Ring, and reports any changes to LNM. Also reports on various other activities such as the change of the Active Monitor on a Token Ring.

[**no**] **lnm loss-threshold** *number*

A configurable number which expresses the percentage loss rate in hundredths of a percent of number of frames dropped compared to number of frames forwarded.

[**no**] **lnm password** *number string*

Assigns a password to a reporting link to prevent unauthorized access from an LRM to a bridge, and to control access to the different reporting links.

[**no**] **lnm rem**

Enables the Ring Error Monitor (REM), which monitors errors reported by any station on the ring. Also monitors whether the ring is in a functional state or in a failure state.

[**no**] **lnm rps**

Enables the Ring Parameter Server (RPS), which ensures that all stations on a ring are using a consistent set of reporting parameters, and report to LNM when any new station joins a Token Ring.

[**no**] **lnm softerr** *number*

Controls the frequency of error reports sent from stations on a Token Ring to the Ring Error Monitor. The *number* is a time interval in tens of milliseconds between error messages. The **no** version of the command restores the timer value to its default of 200, or two seconds.

mac-address *ieee-address*

Sets the MAC layer address. The argument *ieee-address* is a 48-bit IEEE MAC address written as a dotted triple of four-digit hexadecimal numbers. This command forces the use of a different MAC address on the specified interface, thereby avoiding the TI MAC firmware problem. It is up the network administrator to ensure that no other host on the network is using that MAC address.

[**no**] **multiring** {*protocol-keyword* | **all** | **other**}

Enables or disables the specified interface's ability to collect and use source-route (RIF) information for routable protocols. The argument *protocol-keyword* is one of: **apollo**, **ap-pletalk**, **clns**, **decnet**, **ip**, **vines**, or **xns**. The **all** keyword enables the multiring for all frames; the **other** keyword enables the multiring for any frame not included in the previous list.

[**no**] **netbios input-access-filter bytes** *name*

Defines an access list filter on incoming messages. The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list bytes** global configuration commands. Use the **no** version of the command with the appropriate name to remove the entire access list.

[**no**] **netbios input-access-filter host** *name*

Defines an access list filter on incoming messages. The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list host** global configuration commands. Use the **no** version of the command with the ap-propriate argument to remove the entire access list.

[**no**] **netbios output-access-filter bytes** *name*

Defines an access list filter on outgoing messages. The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list bytes** global configuration commands. Use the **no** version of the command to remove the entire access list.

[**no**] **netbios output-access-filter host** *name*

Defines an access list filter on outgoing messages. The argument *name* is the name of a NetBIOS access filter previously defined with one or more of the **netbios access-list** global configuration commands. Use the **no** version of the command to remove the entire access list.

[**no**] **source-bridge** *local-ring bridge-number target-ring*

Enables and disables source bridging on a specific interface.

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

Assigns an access list to a particular interface for filtering the Token Ring or IEEE 802 source addresses. Use this variation of the **source-bridge** interface subcommand to do this. The argument *list* is the access list number. The **no** version of this command removes the application of the access list.

source-bridge input-lsap-list *list*

Filters IEEE 802-encapsulated packets on input. The access list specifying the type codes to be filtered is given by this variation of the **source-bridge** interface subcommand. The argument *list* is the access list number. This access list is applied to all IEEE 802 frames received on that interface prior to the source-routing process. Specifying a zero disables the filter.

source-bridge input-type-list *list*

Filters SNAP-encapsulated packets on input. Use the access list specifying the type codes to be filtered The argument *list* is the access list number. This access list is then applied to all SNAP frames received on that interface prior to the source-routing process. Specify a zero for *list* to disable the application of the access list on the bridge group.

[**no**] **source-bridge max-hops** *count*

Limits the maximum number of source-route bridge hops of your network. The argument *count* determines the number of bridges an explorer packet may traverse. The command **no source-bridge max-hops** resets the count back to the maximum value.

[**no**] **source-bridge old-sna**

Enables or disables a workaround for some source-route bridging behavior exhibited by older SNA nodes.

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

Assigns an access list to a particular interface for filtering the Token Ring or IEEE 802 destination addresses. Use this variation of the **source-bridge** interface subcommand to do this. The argument *list* is the access list number. The **no** version of this command removes the application of the access list.

source-bridge output-lsap-list *list*

Filters IEEE 802-encapsulated packets on output. The access list specifying the type codes to be filtered is given by this variation of the **source-bridge** interface subcommand. The argument *list* is the access list number. This access list is applied just before sending out a frame to an interface. Specifying a zero disables the filter.

source-bridge output-type-list *list*

Filters SNAP-encapsulated on output. Use the access list specifying the type codes to be filtered. The argument *list* is the access list number. This access list is then applied just before sending out a frame to an interface. Specify a zero for *list* to disable the application of the access list on the bridge group.

[**no**] **source-bridge proxy-explorer**

Enables and disables the proxy explorer function. The default is disabled.

[**no**] **source-bridge route-cache**

Enables fast switching to allow for faster implementations of local source-route bridging between 4/16 Megabit Token Ring cards in the same Cisco router/bridge. By default, the system enables fast switching in the source-route bridging software. The **no** version of the command disables this feature.

[**no**] **source-bridge spanning**

Manually changes the forwarding state of spanning explorer packets; the **no** form disables forwarding.