Chapter 1
Configuring Source-Route Bridging

1

This chapter describes routing and bridging in Token Ring environments. These topics are included in this chapter:

- The fundamental concepts of 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.
- Filtering datagrams by protocol type, vendor code, and configuring NETBIOS access control filters.

Source-Route Bridging Overview

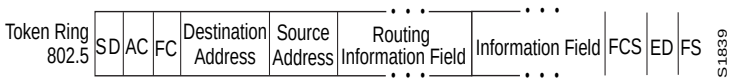
The Cisco bridging software includes source-route bridging capability. This ability allows the Cisco router/bridge to simultaneously act as a Level 3 router and a Level 2 source-route bridge. This allows protocols such as Novell or XNS to be routed on Token Rings, while other protocols such as SNA or 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 RIF is inserted into the MAC header immediately following the source address field in every frame by the source station, 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 that 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 that involve multiple router/bridges separated by non-Token Ring segments by either encapsulating the Token Ring traffic inside IP datagrams passed over a TCP connection between two Cisco router/bridges, or by using HDLC over a serial line 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.

Configuring Source-Route Bridging

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

Step 1: Enable use of the Routing Information Field (RIF) for routed protocols with the

multiring command.

Step 2: 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 says "Source Route Bridge capable," then you may proceed with configuration. Otherwise, you need to contact Cisco Systems for a hardware and firmware field upgrade.

Step 3: Define the types of explorer packets to use: either spanning tree or the default all rings explorer packets.

The source-route bridging software supports filtering of frames. Filtering can be done by protocol type or by vendor code. It is also possible to configure access control filters for packets transmitted across a Token Ring bridge using the NETBIOS interface. Additionally, EXEC-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. This feature allows for faster implementations of local source route bridging between 4/16 megabit Token Ring cards (CSC-R16) in the same Cisco router/bridge. To disable this feature, use the **source-bridge route-cache** interface subcommand. The full syntax of this command follows:

source-bridge route-cache

no source-bridge route-cache

Example:

These commands disable use of fast source-route bridging between two Token Ring units.

```
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
```

Configuring the Source-Route Information Field

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

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. A RIF is built up of ring and bridge numbers.

Figure 1-2 Basic RIF Format

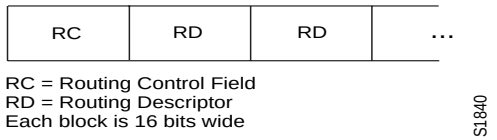


Figure 1-3 illustrates the routing control format for the RIF. Descriptions of each field follow.

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, and Cisco maximum)
  - 100: 8144 bytes (Token Bus)

— 101: 11407 bytes

— 110: 17800 bytes

— 111: 65535 (initial values)

Figure 1-4 describes the routing descriptor format of the RIF string.

Figure 1-4 Routing Descriptor Format

- Ring Number—Unique decimal ring number within the bridged network.
- Bridge Number—Unique decimal 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 they 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 **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 non-multiring-capable device speaking a different network protocol. If the earlier **multiring** command (pre-8.3 release) was not not specified, nodes speaking one particular protocol would be able to communicate through the Cisco router, but nodes speaking the different protocol could not. The reverse was true when the multiring capability was specified on the interface.

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.

Note: In 8.3 or later releases of the software, the command **multiring all** is equivalent to the previous **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 Time-Out 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. 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 time-out 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, it may be necessary to add static information to the RIF cache of the router/bridge.

To enter static source route information into the RIF cache, use this following variation of the **rif** global configuration command (negative form of the command included):

rif *MAC-address* [*RIF-string*] [*interface-name* | **ring-group** *ring*]

no rif *MAC-address* [*RIF-string*] [*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.

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 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-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.

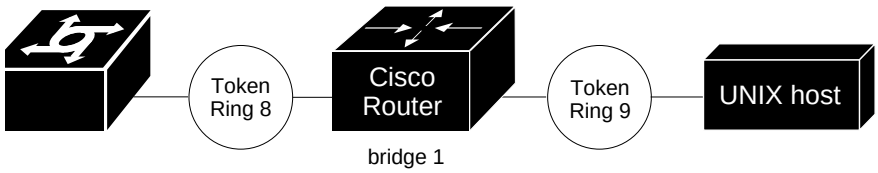
Note: Input to the **source-bridge** configuration subcommand is in decimal format. RIF displays and input are in hexadecimal format, and IBM source-route bridges use hexadecimal for input. It is essential that bridge and ring numbers are consistent for proper network operation. This means you must explicitly 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.

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

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 follows.

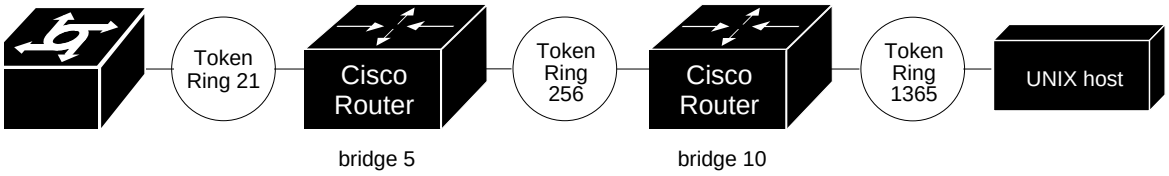
Figure 1-5 Assigning a RIF to a Source-Route Bridge




```
!
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.

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 nonToken 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 non-routed protocols will be source-bridged, however.

Enabling Local Source-Route Bridging

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

source-bridge *local-ring bridge-number target-ring*

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.

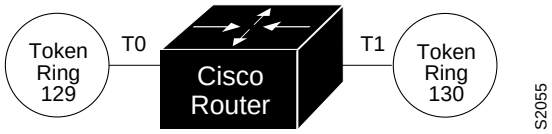
The argument *target-ring* is the decimal ring number of the destination ring on this router/bridge. It must also be unique within the bridged Token Ring network.

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

Example:

The following example configures this simple two-port bridge.

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-rd** interface subcommand. The full command syntax follows:

source-bridge max-rd *count*

no source-bridge max-rd

The argument *count* determines the number of route descriptors that may appear in the RIF. It is one more than the number of bridges an explorer packet may traverse. The typical maximum for interoperability with IBM equipment is 7 (eight rings and seven bridges).

The command **no source-bridge max-rd** 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-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 command. The full command syntax follows:

source-bridge ring-group *ring-group-number*

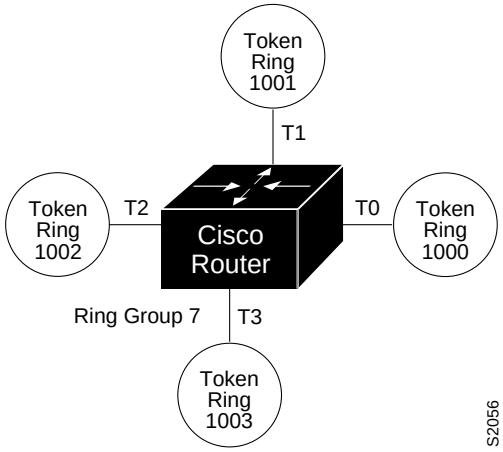
no source-bridge ring-group *ring-group-number*

To configure a specific interface as part of a ring group, its target ring number parameter is set to the ring group number.

Example:

Following is 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 sections have discussed local source-route bridging, 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 nonToken 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: One is to 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.

The second method for setting up a remote source-route bridge is to use a dedicated serial line between two routers attached to Token Rings. This method is recommended when you are running source-route bridge traffic over a slow serial line (56 kilobits per second or less). You do not have the flexibility of the TCP approach, but you do have better performance since there is less of the overhead associated with TCP.

Configuring Remote Source-routing over TCP

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

- 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."
- 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.
- 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 (the negative form of the command is also listed):

source-bridge remote-peer *ring-group* **tcp** *ip-address* [**lf** *size*]

no source-bridge remote-peer *ring-group* **tcp** *ip-address*

This command is used to identify the IP address of a peer in our 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. This argument is useful in preventing time-outs 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 megabit 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, 1470, 2052, 4472, 8144, 11454, and 17,800 bytes.

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*.

```
!
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 131.108.4.4 255.255.255.0
!
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. You use the **source-bridge tcp-queue-max** 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 defeat this limit.

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-Routing over Point-to-Point Serial

To configure a remote source-route bridge to use a point-to-point serial line, 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, and 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 serial connection (the negative form of the command is also included):

source-bridge remote-peer *ring-group* **interface** *interface-name* [**lf** *size*]

no source-bridge remote-peer *ring-group* **interface** *interface-name*

This command is used to identify the interface over which to send source-route bridged traffic to another Cisco router/bridge in our ring group. The interface must be serial, and must use the HDLC encapsulation.

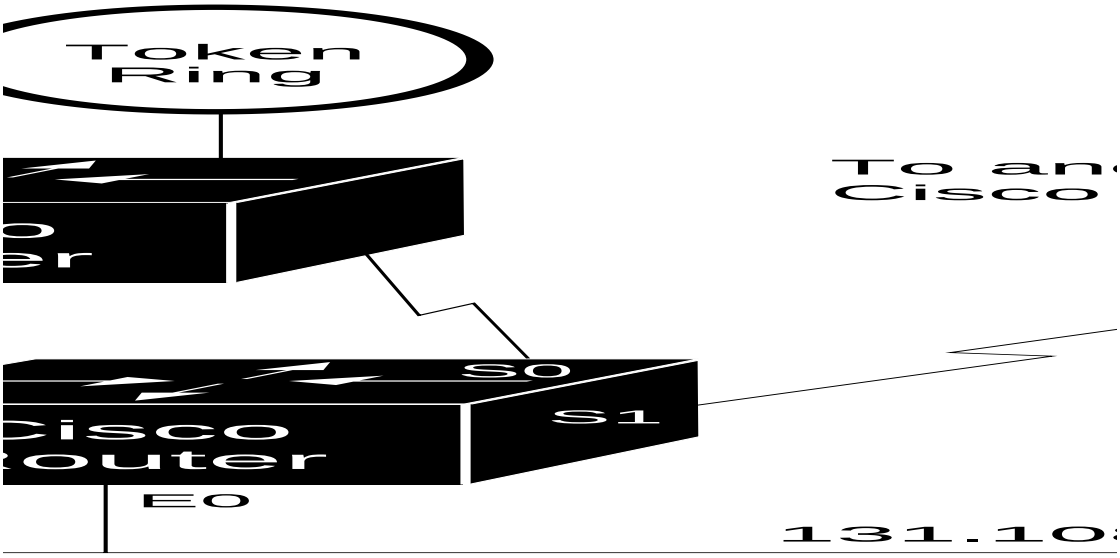
The keyword **lf** specifies the maximum size frame to be sent to this remote peer. The router negotiates all transit routes down to this size or lower. This argument is useful in preventing time-outs in end hosts, by reducing the amount of data they have to transmit in a fixed interval. For example, 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 megabit 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, 1470, 2052, 4472, 8144, 11454, and 17,800 bytes.

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

Example:

The following is an example configuration of a router/bridge configured for remote source-route bridging using both the serial and TCP transport methods.

Figure 1-9 Remote Source-Route Bridge Using Both Serial and TCP Transport Methods




```
!
source-bridge ring-group 5
source-bridge remote-peer 5 interface serial0
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 serials 0 and 1 and they have been configured to use the serial transport. The peers must also belong to the same ring group.

Configuring the Largest Sized 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 this ring group. The full syntax of the command follows:

source-bridge largest-frame *ring-group size*

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 this size or lower. This argument is useful in preventing time-outs 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 megabit 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, 1470, 2052, 4472, 8144, 11454, and 17,800 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. Proxy explorers can be used to limit the amount of explorer traffic propagating through the source-bridge network, especially across low bandwidth serial lines. The use of proxy explorer is most useful for multiple connections to a single mode.

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
!
```

Configuring Administrative Filtering

Source routing bridges normally filter datagrams according to the routing information contained in the datagram. That is, a bridge will not forward a datagram back to its originating network segment or any other network segment that the datagram 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 802.5 or SNAP
- Token Ring Vendor code

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 Appendix C.

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*

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 802.5-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 802.2 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.

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 (described in a following section).

```
!
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 802.5 Frames on Input

You can filter IEEE 802.5-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 802.5 frames received on that interface prior to the source-routing process.

Example:

This command specifies access list 203.

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

Filtering IEEE 802.5 Frames on Output

The software allows you to filter IEEE 802.5-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.

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.

Example:

This command specifies access list 202.

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

Filtering SNAP Frames on Output

To filter SNAP-encapsulated 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.

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

Access lists for token ring- and IEEE 802.5-encapsulated packets affect only source-route bridging functions. *S*uch 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 802.5 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 a*nd *mask a*re 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 802.5 standard uses this bit to indicate whether a RIF is present, and not as part of the source address.

Filtering Source Addresses

To configure filtering on IEEE 802.5 source addresses, assign an access list to a particular interface for filtering the token ring or IEEE 802.5 source addresses. Use this variation of the **source-bridge** interface subcommand to do this (following syntax includes negative form of the command):

source-bridge input-address-list *list*

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

The argument *list* is the access list number. See the section "Filtering Destination Addresses" for an example of command use.

Filtering Destination Addresses

To configure filtering on IEEE 802.5 destination addresses, assign an access list to a particular interface for filtering the token ring or IEEE 802.5 destination addresses. Use this variation of the **source-bridge** interface subcommand to do this (negative form of the command included):

source-bridge output-address-list *list*

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

The argument *list* is the access list number.

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.FFFF
access-list 700 permit 0000.0000.0000   FFFF.FFFF.FFFF
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).

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 which 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 1-1 explains the pattern matching symbols that can be used.

Table 1-1 Station Name Pattern Matching Characters

Character	Action
*	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 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 then 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 bytes 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.

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 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. 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 wild card 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:

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

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.

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
!
```

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 four-megabit (CSC-R) Token Ring card.

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

source-bridge old-sna

no source-bridge old-sna

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.

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 i*s 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 LANmanager 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 LANmanager software can be configure to ignore frame-copied errors.

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

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 which would abort those connections. This configuration is shown in Figure 1-10.

Figure 1-10 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
```

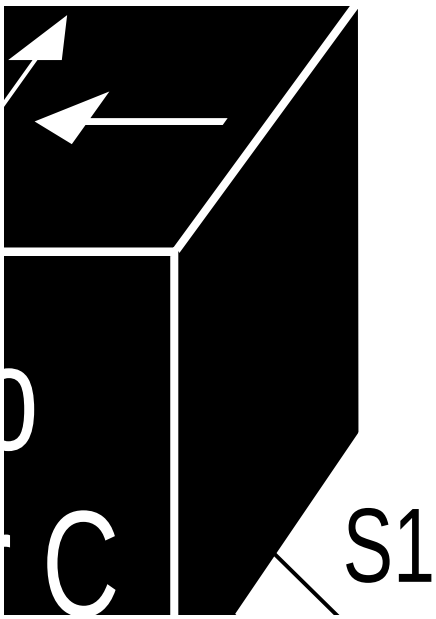
```
network 204.31.10.0
!
hostname RouterB
!
```

Remote Source-Route Bridge—Complex Configuration

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.

This configuration is shown in Figure 1-11.

Figure 1-11 Remote Source-Route Bridge—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
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 131.21.17.1 255.255.255.0
appletalk address 17.1
appletalk zone Twilight
!
interface Serial 2
ip address 131.21.18.1 255.255.255.0
appletalk address 18.1
appletalk zone Twilight
!
router igrp 109
network 131.21.0.0
!
hostname RouterC
!
```

Configuration for Router/Bridge D:

```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
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 131.21.19.1 255.255.255.0
appletalk address 19.1
appletalk zone Twilight
!
router igrp 109
network 131.21.0.0
!
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 131.21.17.2 255.255.255.0
appletalk address 17.2
appletalk zone Twilight
!
interface Serial 2
ip address 131.21.18.2 255.255.255.0
appletalk address 18.2
appletalk zone Twilight
!
router igrp 109
network 131.21.0.0
!
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
```

Table 1-2 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:

```
Local Interfaces:          max       receive             transmit
       srn bn  trn rg px sp rd      cnt:bytes           cnt:bytes       drops
 TR0    1  1    5  *  *  *  8        5:233               0:0               0
 TR1    2  1    5  *  *  *  8        4:224               0:0               0
Ring Group 5:
This peer: TCP 131.108.2.29.
Peers:               state    pkts_rx    pkts_tx    drops     TCP q len
 TCP 131.108.2.68    open          0          0        0         0
 TCP 131.108.2.29    -             0          0        0         0
 TCP 131.108.161.2   open          0          0        0         0
Rings:
    bridge 1  ring 1    local  TokenRing0          forwards: 0
    bridge 1  ring 2    local  TokenRing1          forwards: 0
    bridge 2  ring 102  remote TCP 131.108.161.2   forwards: 0
    bridge 2  ring 101  remote TCP 131.108.161.2   forwards: 0
    bridge 1  ring 6    remote TCP 131.108.2.68    forwards: 0
    bridge 1  ring 7    remote TCP 131.108.2.68    forwards: 0
```

Table 1-3 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	Indicates the group in which the interface is configured.
rg	Indicates a 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 (*).
Ring Group:	Describes the ring group.
This peer:	Lists the address and address type of this peer.
Peers:	Lists the addresses and address types of the ring group peers.
state	Lists the current state of the peer, open or closed. A hyphen indicates this router.
pkts_rx	Lists the number of packets received.
pkts_tx	Lists the number of packets transmitted.
drops	Lists the number of dropped packets.
TCP q len	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 address or interface type, and the number of packets forwarded.

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 personnel for system troubleshooting.

Displaying Token Ring Interface Statistics

The **show interface** 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 interface

Refer to the section "Token Ring Interface Support" in the "Configuring the Interfaces" chapter 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 Customer Engineering, to track system problems. For each **debug** command there is a corresponding **debug** command that stops the output.

The **debug rif** command provides informational displays for entries entering and leaving the RIF cache. Enter this command at the EXEC prompt.

debug rif

Descriptions of the messages with 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.

The **debug source-bridge** command provides informational displays of source bridging activity. Enter this command at the EXEC prompt:

debug source-bridge

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.

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

debug source-event

The **debug token-event** command provides informational displays about significant Token Ring hardware events. Enter this command at the EXEC prompt:

debug token-event

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. Enter this command at the EXEC prompt:

debug token-ring

Source-Route Bridge Global Configuration Command Summary

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

[**no**] **rif** *mac-address* [*rif-string*][**interface**|**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** or appropriate ring-group **ring** may also be specified to indicate the direction from which this RIF entry would have arrived.

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

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

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

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.

[**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 remote-peer** *ring-group* **tcp** *ip-address* [**lf** *size*]

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.

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

Establishes or removes a ring group.

Source-Route Bridge Interface Subcommand Summary

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

[**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**, **appletalk**, **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**] **source-bridge** *local-ring bridge-number target-ring*

Enables and disables source bridging on a specific interface.

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

Allows the system administrator to limit the maximum RIF size that this bridge will deal with. *The argument count* determines the number of route descriptors that may appear in any explorer RIF. The command **no source-bridge max-rd** resets the count back to the maximum value.

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

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

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

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

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

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