BCRAN Quick Reference Sheets

Wide-Area Network Technologies

WAN Technologies

- **Asynchronous Transfer Mode (ATM)**
  - —Cell relay technology
  - —One physical interface can support many virtual circuits
  - —1.544 Mbps to 10 Gbps interfaces
  - —Uses fixed 53 byte cell
  - —Can support different traffic classes, such as variable bit rate (VBR), constant bit rate (CBR), unspecified bit rate (UBR), and available bit rate (ABR)
  - —Can support any Layer 3 protocol
- **Dedicated Connections**
  - —Typically 56 kbps or T1 connections
  - —Supported using circuit switching in the service provider network
  - —56 kbps to 45 Mbps interface speeds
  - —One-to-one mapping of physical to logical connections
  - —More expensive than packet or circuit switched connections of similar speeds
  - —Can support any Layer 3 protocol
- **Frame Relay**
  - —Packet switching technology
  - —Can support interfaces from 56 kbps to 45 Mbps
  - —Permanent virtual circuits (PVCs) are built between Frame Relay endpoints
  - —One physical interface can support many PVCs
  - —Can support any Layer 3 protocol
- **Integrated Services Digital Network (ISDN)**
  - —Circuit switching technology
  - —Two interface types
    - Basic Rate Interface (BRI)
      - One 16 kbps D channel
      - Two 64 kbps B channels
    - Primary Rate Interface (PRI)
      - One 64 kbps D channel
      - 23 64 kbps B channels in North America
      - 30 64 kbps B channels elsewhere
  - —BRI commonly used as a backup connection
  - —Can support any Layer 3 protocol

Working with Asynchronous Connections

Using Group-Async Interfaces

A Group-Async interface allows many asynchronous interfaces to share the same configuration. This is most widely used to support a remote access dialup solution.

This command creates a Group-Async virtual interface:

```
interface Group-Async number
```

This command associates a range of asynchronous lines to a Group-Async interface. This command is an interface configuration command and is available only under Group-Async interfaces:

```
group-range start end
```

Using the modemcap Database

The modemcap database contains a base configuration for many popular modems. The following commands use the modemcap database to configure modems.

This line configuration command automatically determines the modem type and then uses information in the modemcap database to configure it correctly:

```
modem autoconfigure discovery
```

The following line configuration command looks up the modem type specified and applies the configuration from the modemcap database:

```
modem autoconfigure type modem-type
```

It is also possible to add to or edit existing entries in the modemcap database. The following table details the relevant modemcap database editing commands.

Command	Description
modemcap edit *modem-name attribute value*	Edits a particular value within a modemcap entry
modemcap entry *modem-name*	Adds a modemcap entry to the modem database
show modemcap [*entry-name*]	Shows the current contents of the modemcap database

Using Reverse Telnet to Troubleshoot Modems

A *reverse Telnet* is a Telnet session that connects the user to an asynchronous serial line on the router. This can be used to interact directly with a modem for troubleshooting. A reverse Telnet is established by Telneting to any IP address on the router on a specific TCP port. The following ports ranges are used as the Telnet destination.

Character Mode Telnet	2000–2999
Line Mode TCP	4000–4999
Binary Mode Telnet	6000–6999
Xremote	9000–9999

To determine the exact port number to use, find the correct line in the output of **show line**. The leftmost column is the line number; simply add that number to the desired range in the preceding table.

For example, the command **telnet 192.168.1.1 2001** connects you to the first asynchronous line on a router whose IP address is 192.168.1.1.

NOTE For the Telnet session to successfully establish, you must Telnet to the IP address of an interface in an up/up state.

Once connected to a modem, you can enter AT commands directly. The following table lists some important and helpful commands.

AT&F	Reloads factory defaults
ATA	Answers the incoming call
ATDT *phone-number*	Uses tone dialing to place a call to the number specified
ATE0	Disables local echo
ATM0	Disables the modem speaker
ATS0=*x*	Specifies the ring on which to auto answer incoming calls
ATZ	Resets the modem

Physical Line Configuration

Many characteristics of a physical line can be manually configured. The following line configuration commands are used to specify the operation of router lines.

Command	Description
speed *speed*	Configures the speed of modem to router communication
flowcontrol {**none** | **software** | **hardware**}	Configures flow control between the router and modem
modem {**dialin** | **dialout** | **inout**}	Specifies the direction in which the modem can be used
stopbits {**0** | **1**}	Configures the stopbits on the physical interface
physical-layer async	Configures a synchronous/asynchronous interface for asynchronous operation
autoselect {**ppp** | **during-login**}	Enables the detection and use of PPP after a line is connected

Chat Scripts

Chat scripts are used to talk to or through a modem. Constructed using a simple expect-send syntax, chat scripts can prove useful. A chat script can be launched at several instances:

- Line activation
- Line connection
- Line reset
- New call
- Dialer startup

An example of chat script creation follows:

```
Router(config)# chat-script CS-SAMPLE ABORT ERROR ABORT BUSY "" "ATZ" OK
  "ATDT14155551111" TIMEOUT 40 CONNECT \c
```

To associate a chat script to an interface or remote host, use one of the following commands:

- **dialer map**
- **script activation**
- **script connection**
- **script dialer**
- **script reset**
- **script startup**

Chat scripts can include escape characters. These escape characters are replaced when a script is executed. The following escape characters are supported.

Character	Description
\\	The \ character
\"	Sends the " in a double quoted send string
\c	Suppresses a carriage return at the end of the connect string
\d	Two-second delay
\K	Inserts a break
\n	A newline character
\N	Sends a null character
\p	Pauses for 1/4 of a second
\r	Sends a return
\s	Sends a space
\t	A tab character
\T	Replaced by the phone number from the dialer string or map

Configuring PPP with PAP and CHAP

Benefits of PPP

PPP offers many benefits, including

- Bidirectional authentication through PAP or CHAP
- Multiprotocol support
- Flexible client IP address assignment
- Bandwidth aggregation with multilink PPP
- Link fragmentation and interleaving
- Link quality monitoring

Components of PPP

The following table describes the components of PPP.

Component	Description
LCP	The Link Control Protocol (LCP) is used to test, establish, and monitor the PPP connection.
Authentication	The Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP) are responsible for authenticating each end of a PPP connection; the implementation of these is optional, but strongly recommended.
NCPs	Network Control Protocols (NCPs) exist for each protocol supported over PPP; they establish and configure protocol specific parameters. Example NCPs include IPCP, IPXCP, ATalkCP, BridgeCP, and CDPCP.

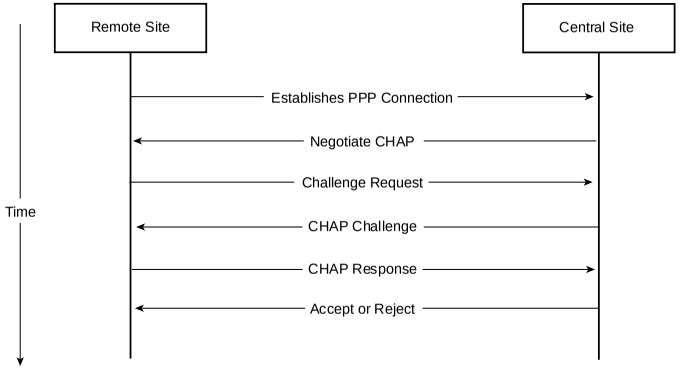
The state of LCP and any configured NCPs can be viewed with the EXEC command **show interfaces**.

Differences Between PAP and CHAP

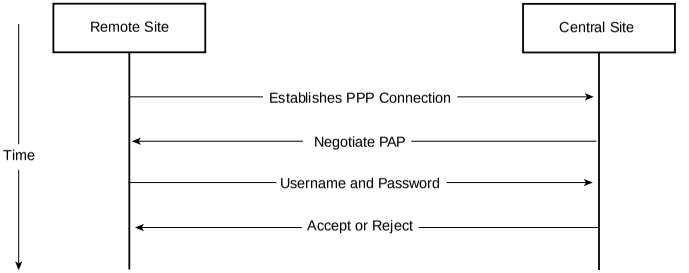
The following table summarizes the differences between the Password Authentication Protocol (PAP) and the Challenge Handshake Authentication Protocol (CHAP).

CHAP	PAP
Passwords never sent over the network	Passwords sent as clear text
Provides two-way authentication	One-way authentication; can be run separately in both directions to establish two-way authentication

In CHAP, a three-way handshake authenticates users.



With PAP, because there is no attempt at secrecy, the process is much simpler.



Enabling PPP Encapsulation

The following interface configuration command enables PPP encapsulation:

```
encapsulation ppp
```

Configuring Authentication

Basic PPP authentication can be performed with the two following commands:

```
username username password password
ppp authentication {pap | chap | pap chap | chap pap}
```

In the following example, an access server is configured to attempt CHAP first, then fall back to PAP:

```
Router(config)# username jane password astro
Router(config)# username fred password dino
Router(config)# interface Group-Async 1
Router(config-if)# encapsulation ppp
Router(config-if)# ppp authentication chap pap
```

Configuring Compression

For compression to be successfully negotiated, it must be enabled in both directions. Compression is enabled using the following interface configuration command:

```
compress [predictor | stac | mppc [ignore-pfc]]
```

The state of compression can be viewed using the EXEC command **show compress**. If compression has been negotiated on a PPP link, the **show interfaces** EXEC command displays the CCP state as Open.

IP Address Assignment with PPP

An IP address can be assigned to a PPP client in several ways:

- TACACS+ or RADIUS server
- IP pool defined on the access server
- Dynamic Host Configuration Protocol (DHCP)
- Via the **ppp** EXEC command
- Via the **peer default ip address** interface configuration command
- PPP client configuration

The default address allocation method can be controlled with the global configuration command:

```
ip address-pool {dhcp-proxy-client | local}
```

An IP address pool is configured using the following command:

```
ip local pool pool-name start-address end-address
```

After the pool is created, it can be applied with the following interface configuration command:

```
peer default ip address pool pool-name
```

To configure the access server to use DHCP to determine addresses available for client assignment, use this global configuration command:

```
ip dhcp-server {ip-address | name}
```

After the DHCP servers are configured and the default address allocation method set to DHCP, the following interface configuration command enables IP address assignment from the DHCP retrieved addresses:

```
peer default ip address pool dhcp
```

A specific address can be assigned to users dialing into a particular access server interface with the following interface configuration command:

```
peer default ip address ip-address
```

Configuring Multilink PPP

The configuration of multilink PPP (MLP) depends on the implementation scenario. To properly bundle asynchronous interfaces, PPP authentication must be implemented. The following configuration example configures MLP on an ISDN BRI interface:

```
Router(config)# interface bri0
Router(config-if)# encapsulation ppp
Router(config-if)# ppp authentication chap pap
Router(config-if)# ppp multilink
```

To configure MLP on a dialer interface, the **ppp multilink** command must be applied to the physical and dialer interface. The following configuration sample shows this configuration:

```
Router(config)# interface bri 0
Router(config-if)# encapsulation ppp
Router(config-if)# ppp multilink
Router(config-if)# dialer pool-member 1
Router(config-if)# interface Dialer 0
Router(config-if)# encapsulation ppp
Router(config-if)# ppp authentication chap pap
Router(config-if)# ppp multilink
Router(config-if)# dialer pool 1
```

The configuration of MLP over synchronous interfaces can be accomplished in one of two ways. The first, shown here, uses virtual templates to configure the MLP group:

```
Router(config)# multilink virtual-template 1
Router(config)# interface Virtual-Template 1
Router(config-if)# ip address 10.1.1.1 255.255.255.252
Router(config-if)# ppp authentication pap
Router(config-if)# ppp multilink
Router(config-if)# interface Serial 0
Router(config-if)# encapsulation ppp
Router(config-if)# ppp multilink
Router(config-if)# interface Serial 1
Router(config-if)# encapsulation ppp
Router(config-if)# ppp multilink
```

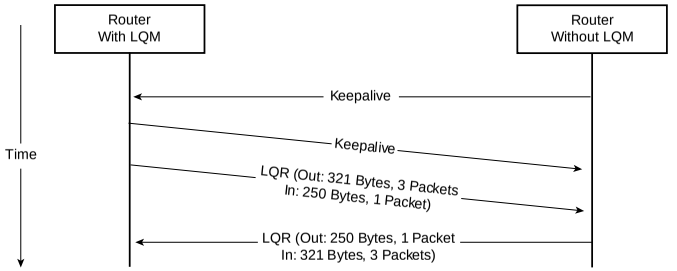
The second MLP over synchronous interfaces option does not involve the creation of a virtual template interface. The advantage to this type of configuration is that multiple distinct MLP bundles can exist on the same router. This configuration is as follows:

```
Router(config)# interface Serial 0
Router(config-if)# encapsulation ppp
```

```
Router(config-if)# ppp multilink
Router(config-if)# multilink-group 1
Router(config-if)# interface Serial 1
Router(config-if)# encapsulation ppp
Router(config-if)# ppp multilink
Router(config-if)# multilink-group 1
Router(config-if)# interface Multilink 1
Router(config-if)# ip address 10.1.1.1 255.255.255.252
Router(config-if)# ppp authentication pap
Router(config-if)# ppp multilink
```

Enabling and Using PPP Link Quality Monitoring

When PPP link quality monitoring (LQM) is enabled, link quality reports (LQRs) are sent instead of the regular interface keepalives. If LQM determines that the quality of an interface has dropped below the configured threshold, the interface is taken down. LQM can be enabled on one or both ends of the PPP connection.



To enable LQM on a PPP interface, use the following interface configuration command:

ppp quality *percentage*

Client Host Route Creation

By default, the access server installs a host route (a route with a 32-bit mask) for each connected PPP client. To disable this functionality, use the interface configuration command as follows:

no peer neighbor-route

Verification of a PPP Configuration

The following commands can be used to view the status of a configured PPP configuration:

- **show compress**
- **show interfaces** [*type number*]
- **show ppp multilink**
- **show users** [**all**]

Configuring Dial-on-Demand Routing

Dial-on-Demand Routing Implementation Options

Dial-on-demand routing (DDR) can be implemented in one of two manners:

- **Legacy DDR**—In legacy DDR, the dialer configuration is placed directly on the physical interface. Although this simplifies the configuration somewhat, the lack of abstraction restricts the configuration possibilities.
- **Dialer Profiles**—With dialer profiles, virtual dialer interfaces are configured, and then bound to a group of physical interfaces. This permits more dialer interfaces to exist than physical interfaces. It also permits the router to operate around failed interfaces.

Configuring DDR

Many of the commands in legacy DDR and dialer profiles are shared between the two configurations. The following table details most of the commands shared between these two configurations.

Command	Description
dialer fast-idle *seconds*	Deactivates idle interfaces when physical interface contention exists
dialer idle-timeout *seconds*	Disconnects DDR sessions that are idle

Command	Description
dialer load-threshold *load* [**inbound** | **outbound** | **either**]	Used with multilink PPP to establish additional dialer connections during periods of high network traffic
dialer map *protocol address phone-number*	Associates a Layer 3 address with a remote phone number
dialer string *phone-number*	Statically configures a remote phone number on an interface
dialer-group *number*	Groups a dialer list with a dialer interface for the selection of interesting traffic
dialer-list *number* **protocol** *protocol* {**permit** | **deny**} [*keyword value*]	Creates a dialer list that determines what traffic activates a dialer interface
encapsulation ppp	Enables PPP encapsulation on an interface

The following commands are specific to dialer profiles and provide much of their additional functionality.

Command	Description
dialer pool *number*	Associates a dialer interface with a pool of physical interfaces
dialer pool-member *number*	Places a physical interface into a dialer pool
dialer rotary-group *number*	Places a physical interface into a rotary group
interface dialer *number*	Creates a dialer interface; if this interface is to be associated with a configured rotary group, the interface number must match the rotary group number

The following configuration snippet configures two BRI interfaces as a member of a dialer pool and then configures two dialer interfaces for IP routing:

```
Router(config)# interface BRI 0
Router(config-if)# dialer in-band
Router(config-if)# dialer pool-member 10
Router(config-if)# interface BRI 1
Router(config-if)# dialer in-band
Router(config-if)# dialer pool-member 10
Router(config-if)# exit
Router(config)# dialer-list 1 protocol ip permit
Router(config)# interface Dialer 1
Router(config-if)# ip address 10.1.1.1 255.255.255.0
Router(config-if)# encapsulation ppp
Router(config-if)# dialer pool 10
Router(config-if)# dialer-group 1
Router(config-if)# dialer map ip 10.1.1.2 4155551111
Router(config)# interface Dialer 2
Router(config-if)# ip address 10.1.2.1 255.255.255.0
Router(config-if)# encapsulation ppp
Router(config-if)# dialer pool 10
Router(config-if)# dialer-group 1
Router(config-if)# dialer map ip 10.1.2.2 4155552222
```

Dialer Pools and Rotary Groups

Dialer pools and rotary groups have a similar goal; the separation of physical and logical interfaces. One difference to their implementation exists. With rotary groups, only one dialer interface is allowed to use the members of the group. In contrast, with dialer pools, many dialer interfaces are allowed to share the pool members.

To configure rotary groups, the dialer interface number must match the rotary group number, as shown in this abbreviated example:

```
Router(config)# interface BRI 0
Router(config-if)# dialer rotary-group 12
Router(config-if)# interface BRI 1
Router(config-if)# dialer rotary-group 12
Router(config-if)# interface Dialer 12
```

This one-to-one mapping is the downside of rotary groups. The following example updates the previous to use dialer pools with two dialer interfaces:

```
Router(config)# interface BRI 0
Router(config-if)# dialer pool-member 14
Router(config-if)# interface BRI 1
Router(config-if)# dialer pool-member 14
Router(config-if)# interface Dialer 1
Router(config-if)# dialer pool 14
Router(config-if)# interface Dialer 2
Router(config-if)# dialer pool 14
```

Determining Interesting Traffic

A router uses the idea of interesting traffic to determine if a dormant DDR link should be established. The **dialer-list** global configuration command defines interesting traffic. By using different dialer list numbers, interesting traffic can vary by interface. The complete syntax for the **dialer-list** command is as follows:

dialer-list *list-number* **protocol** *protocol* {**permit** | **deny**} [**list** *access-list-number*]

The **dialer-list** command is used regardless of the protocol you want to define as interesting. Valid protocols include appletalk, bridge, clns, clns_es, clns_is, decnet, decnet_router-L1, decnet_router-L2, decnet_node, ip, ipx, vines, and xns.

The **list** option to this command provides increased granularity through the association of an access list to a dialer list. Any traffic that is permitted by the access list qualifies as interesting and in turn enables the DDR interface.

NOTE Remember that ALL access lists include an implicit deny any as the last entry.

The following is an example access list that selects IP as interesting while excluding Open Shortest Path First (OSPF):

```
access-list 100 deny ospf any any
access-list 100 permit ip any any
dialer-list 1 protocol ip list 100
```

NOTE Although interesting traffic is used to establish a DDR connection, after the connection is established, all traffic is allowed to flow. The idle timeout, however, is reset only when interesting traffic is sent or received.

Dialer Map Classes

A dialer map class is a means to apply an identical configuration to multiple dialer interfaces or destinations. Map class configuration begins with the following command:

map-class dialer *map-class-name*

Once in map class configuration mode, a limited selection of dialer commands is available to you. The following table documents these commands.

dialer callback-server {**dialstring** | **username**}	Enables callback for the incoming connections
dialer enable-timeout *seconds*	Configures the time period before a router reuses a physical interface
dialer fast-idle *seconds*	The idle disconnect timeout used when Dialer interfaces are waiting for an available physical interface
dialer idle-timeout *seconds*	Configures when an idle DDR connection is disconnected
dialer isdn {**speed** *speed* | **spc**}	Sets the data rate of the ISDN B channel
dialer wait-for-carrier-time *seconds*	Determines how long the router waits after dialing for a connection to be established

To apply a map class use the class option to the **dialer-map** interface configuration command, as shown in the following example:

```
Router(config)# interface Dialer 1
Router(config-if)# dialer map ip 10.1.2.1 4155551111 class MC-EXAMPLE
```

Verification of a Dial-on-Demand Routing Configuration

The following commands can be used to diagnose and troubleshoot a DDR configuration:

- **show interfaces** [*type number*]
- **show dialer**

- **show dialer map**
- **show dialer sessions**
- **show users**

Configuring Frame Relay

Characteristics of Frame Relay

- Packet switching—Packets from many connections are multiplexed onto a single physical circuit.
- High speed—Frame Relay typically supports connections from 56 kbps to 45 Mbps, with a full T1 (1.544 Mbps) being the most commonly deployed access port.
- No mechanism for retries—Frames that encounter errors in the network are dropped and left for the upper layer protocols to recover.
- Virtual circuits—One physical connection can support many virtual circuits (VC); each VC is identified by a data-link connection identifier (DLCI).
- Access and guaranteed data rates might be different—The port speed, the speed at which the router connects to the network, might be much faster than the committed information rate (CIR), a per-VC value that represents the amount of traffic guaranteed by the service provider to reach its destination.

Frame Relay Network Design Points

- Many VCs can be supported on a single physical interface.
  - —Frame Relay subinterfaces can be multipoint or point-to-point.

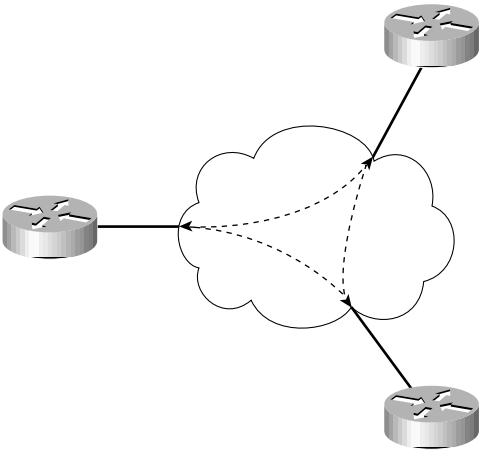
    Advantages of multipoint interfaces include more efficient use of IP address space.

    Advantages of point-to-point subinterfaces include the following:

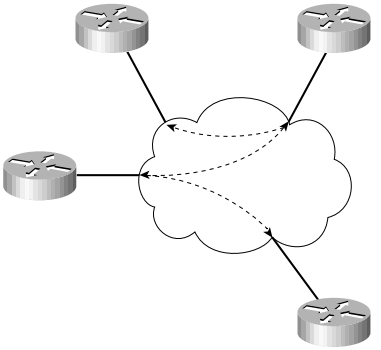
    Broadcasts are not replicated.

    Overcomes split horizon problems.

    Guaranteed routing protocol interoperability.
  - —The physical interface always acts as a multipoint interface.
- Frame Relay networks can be fully or partially meshed.
  - —In a full mesh, each Frame Relay endpoint has a VC to every other Frame Relay endpoint.
  - —The number of VCs required is equal to n(n–1) / 2.



—In a partial mesh, endpoints are connected as needed.



—A hub and spoke topology is essentially a partial mesh in which all the sites are connected to one central (hub) site.

- DLCIs are of local—router to Frame Relay switch—significance only.
  —Valid DLCI numbers differ according to the Local Management Interface (LMI) type used:
  ANSI: 16 to 991
  Cisco: 16 to 1007
  Q933a: 16 to 991
- Virtual interfaces, called subinterfaces, can be used to create virtual point-to-point connections on a per-VC basis.
  —Serial0.16 is an example subinterface name.
- Data sent in excess of the CIR has no guarantee of making it to the destination.
  —This traffic is marked with the Discard Eligible (DE) bit by the Frame Relay network.
- Frame Relay Traffic Shaping (FRTS) should be used to "smooth" traffic down to the CIR, or slightly above.
- Frame Relay map classes make it possible to apply the same configuration to many VCs.
- Must be mindful of routing protocol operation; things like split horizon, OSPF network type, and IS-IS requirements.

Configuring Frame Relay

To enable Frame Relay encapsulation on a serial interface, use the following:

```
encapsulation frame-relay [cisco | ietf]
```

The [**cisco** | **ietf**] argument specifies the LMI type for all VC connected to that interface. The default is **cisco**. For multipoint interfaces, the **ietf** option to the **frame-relay map** command enables IETF encapsulation on a per-VC basis.

To create a subinterface, use the one of the following two commands:

```
Router(config)# interface Serial0.16 multipoint
```

or

```
Router(config)# interface Serial0.16 point-to-point
```

Cisco routers support three Frame Relay LMI types. The LMI type can be configured on an interface basis with the following command:

```
frame-relay lmi-type {cisco | ansi | q933a}
```

When no Frame Relay LMI type is explicitly configured, the router autosenses the appropriate LMI type.

When configuring a multipoint Frame Relay interface, DLCI-to-address and protocol mappings must be manually configured. The following is an example using IP with two VCs:

```
Router(config-if)# frame-relay map ip 192.168.1.1 16 broadcast
Router(config-if)# frame-relay map ip 192.168.1.2 17 broadcast
```

NOTE The **broadcast** argument indicates that the router should replicate broadcast packets over this VC. Without this argument, most routing protocols do not operate correctly.

When you use a point-to-point subinterface, only one VC can be used. The following command associates a DLCI with a point-to-point subinterface:

```
Router(config-subif)# frame-relay interface-dlci dlci
```

Configuring Frame Relay Traffic Shaping (FRTS)

While configuring FRTS, you should know a few variables (see the following table).

Variable	Meaning
Bc	Burst Committed: The amount of data to burst per Tc.
Be	Burst Excess: The amount of data to burst in excess of the contracted rate per Tc.
CIR	This is the contracted rate of traffic guaranteed to make it through the Frame Relay network.
EIR	The amount of traffic in addition to the CIR to send into the network.
MinCIR	The reduced transmission rate used when a BECN is received on a VC. By default MinCIR = CIR / 2.
Tc	This is the time interval used in tuning the performance of FRTS; it defaults to 1/8 of a second.

Two sets of commands can be used to configure FRTS. Both groups of commands are entered on a Frame Relay map class and then applied to a VC. Examples of both groups follow using a map class called MC-EXAMPLE, with a CIR of 128 k and EIR of 64 k:

```
Router(config)# map-class frame-relay MC-EXAMPLE
Router(config-map-class)# frame-relay traffic-rate 128000 192000
Router(config-map-class)# interface Serial0
Router(config-if)# frame-relay traffic-shaping
Router(config-if)# interface Serial0.16 point-to-point
Router(config-subif)# frame-relay interface-dlci 16
Router(config-fr-dlci)# class MC-EXAMPLE
```

or

```
Router(config)# map-class frame-relay MC-EXAMPLE
Router(config-map-class)# frame-relay bc 16000
Router(config-map-class)# frame-relay be 8000
Router(config-map-class)# interface Serial0
Router(config-if)# frame-relay traffic-shaping
Router(config-if)# interface Serial0.16 point-to-point
Router(config-subif)# frame-relay interface-dlci 16
Router(config-fr-dlci)# class MC-EXAMPLE
```

NOTE When the **frame-relay traffic-shaping** command is applied, all VCs without traffic shaping parameters set are shaped to 56 kbps.

When the command **frame-relay adaptive-shaping becn** is entered, the router automatically slows transmission on a VC upon receipt of a BECN. The new rate, called Min-CIR, is by default half of the CIR. This value can be changed with the map class configuration command **frame-relay mincir** *mincir*.

Configuring Compression

Compression can be applied to an interface or subinterface using the following command:

```
frame-relay payload-compression {FRF9 | data-stream | packet-by-packet} stac
```

or on a per-VC basis for multipoint interfaces with the following command:

```
frame-relay map protocol dlci compress [active | passive]
```

Verifying the Frame Relay Configuration

The following commands help you determine if your Frame Relay configuration is correct and operating properly:

- **show frame-relay lmi** [**interface** *type number*]
- **show frame-relay map**
- **show frame-relay pvc** [**interface** *type number*] [*dlci*]
- **show frame-relay traffic**
- **show interfaces** [*type number*]
- **show traffic-shape** [*type number*]
- **show traffic-shape statistics** [*type number*]

Network Redundancy and Backup Connections

Tools for Wide-Area Network Redundancy

- Backup interfaces
- Floating static routes
- Dialer watch

Backup Interface Design Points

- The backup interface is down and unusable until needed.
- The backup interface can be used to provide additional bandwidth.
- A delay can be configured before the backup interface is enabled and disabled.
- A dialer profile can be specified as a backup interface.

Configuring Backup Interfaces

The commands for enabling and tuning backup interfaces are detailed in the following table.

Command	Description
backup interface *type number*	Assigns an interface as a backup interface
backup delay {*enable-delay* | **never**} {*disable-delay* | **never**}	Configures the delay before enabling and disabling a backup interface
backup load {*enable-threshold* | **never**} {*disable-threshold* | **never**}	Configures a threshold for enabling and disabling the backup interface for additional bandwidth

The following configuration example configures Serial 1 as a backup interface to Serial 0. This example also configures the router to enable the backup interface when the primary interfaces load exceeds 75 percent and to disable it once it falls below 50 percent:

```
Router(config)# interface Serial 0
Router(config-if)# backup interface Serial 1
Router(config-if)# backup load 75 50
```

Floating Static Route Design Points

- Care must be taken to ensure the route is not used unless a failure occurs.
- No provision for load balancing or delaying failover and fallback.

Configuring Floating Static Routes

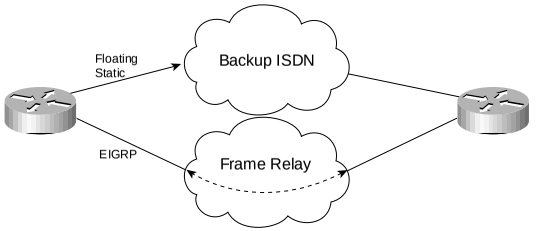
A *floating static route* is a static route that has a higher administrative distance (AD) than a dynamically learned route. IBGP has the highest default AD of any routing protocol at 200. Setting the AD of a floating static route at 210 helps ensure that it is only used for redundancy purposes. The syntax used to create a floating static route is as follows:

```
ip route network mask {next-hop | interface-type number} distance
```

The following is an example of using this command to create a floating static route:

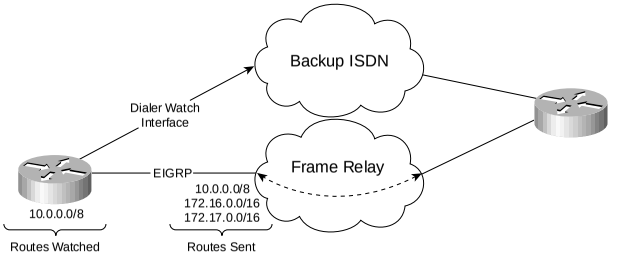
```
Router(config)# ip route 0.0.0.0 0.0.0.0 172.16.1.1 210
```

Once configured, the router uses this floating static route only during a failure of the primary dynamically learned route.



Dialer Watch Design Points

- Configures the router to monitor the existence of a dynamic route or routes
- No provision for load balancing
- Can delay fallback
- Backup interface activated after primary failure, without delay



Configuring Dialer Watch

The following commands configure dialer watch.

The following command defines the routes to watch. Multiple routes can be watched by entering multiple instances of this command. All watched routes must be absent before the dialer watch activates the backup interface:

```
dialer watch-list group-number ip ip-address address-mask
```

This command is applied to the dial-on-demand routing (DDR) interface to be used as a backup interface. The *group-number* argument must match that used in the **dialer watch-list** command:

```
dialer watch-group group-number
```

Applied to the dialer watch backup interface, this command configures the delay before the backup interface is shut down after the primary interface is restored:

```
dialer watch-disable seconds
```

The following configuration example implements dialer watch. This example monitors the route to 10.0.0.0/8 and waits 120 seconds before disabling the backup interface:

```
Router(config)# dialer watch-list 1 ip 10.0.0.0 255.0.0.0
Router(config)# interface Dialer 0
Router(config-if)# dialer watch-group 1
Router(config-if)# dialer watch-disable 120
```

Verification of Network Redundancy

The following commands can monitor and verify the configuration of backup interfaces, floating static routes, and dialer watch:

- **show backup**
- **show dialer** [**interface** *type number*]
- **show interfaces** [**interface** *type number*]
- **show ip route**
- **show ip route static**

Configuring Compression

Types of Compression

- Frame Relay Payload Compression
- HDLC Link Compression
- PPP Link Compression (Microsoft Point-to-Point Compression [MPPC])
- RTP Header Compression

Applying Compression

To apply compression for all virtual circuits (VCs) on an interface, use the interface configuration command:

```
frame-relay payload-compress {packet-by-packet | frf9 stac
  [hardware-options] | data-stream stac [hardware-options]}
```

To apply compression on a per-VC basis, use the **payload-compress** argument and the **frame-relay map** interface configuration command. The following is an example of the use of this argument:

```
Router(config-if)# frame-relay map ip 10.1.1.1 16 broadcast
  payload-compress packet-by-packet
```

Compression can be applied to an interface with HDLC encapsulation using the following interface configuration command:

```
compress {predictor | stac}
```

To apply compression on a PPP link, use the following command in interface configuration mode:

```
compress [predictor | stac | mppc [ignore-pfc]]
```

RTP header compression can be used on point-to-point connections. It substantially reduces the overhead related to RTP. The following command enables RTP header compression on HDLC or PPP links:

```
ip rtp header-compression
```

RTP header compression can be enabled for Frame Relay at the interface level with this command:

```
frame-relay ip rtp header-compression
```

Monitoring and Verification of Compression

Use the following commands to determine the state of compression:

- **show compress**
- **show interfaces** [*type number*]
- **show ip rtp header-compression** [*type number*] [**detail**]

Configuring Queuing

Types of Queuing

- **Class-Based Weighted Fair Queuing (CBWFQ)**
  - —Uses classes to group traffic types by protocol, by access lists, by IP Precedence, by IP ToS, by DiffServ bits, by packet length, and by input interface
  - —Uses policy maps to apply queuing to classes
  - —Integrates other queuing strategies
- **Custom Queuing (CQ)**
  - —Uses access lists and protocol types to classify traffic
  - —17 queues; queue 0 reserved for system use
  - —Can set queue priority
  - —All queues serviced eventually
- **First-In, First-Out (FIFO) Queuing**
  - —Default for interfaces > 2 Mbps
- **Low Latency Queuing (LLQ)**
  - —Strict priority queuing within CBWFQ
  - —Can starve other queues
- **Priority Queuing (PQ)**
  - —Uses access lists and protocol types to classify traffic
  - —High priority queues always emptied first
  - —Four queues: high, medium, normal, and low
  - —Can starve lower priority queues
- **Weighted Fair Queuing (WFQ)**
  - —Default for interface < 2 Mbps
  - —Favors interactive traffic
  - —IP Precedence aware

Implementing CBWFQ

The first step in configuring CBWFQ is to define traffic classes. To create a class, use the following command:

```
class-map class-name
```

Then, traffic can be placed in a class using one of the following commands:

```
match access-group access-list
match input-interface type number
match protocol protocol
```

After the class map is created, you can apply queuing techniques to the class using a policy map. The syntax to create a policy map is as follows:

```
policy-map policy-map
```

To configure a traffic class, enter the policy map configuration command as follows:

```
class class-name
```

The commands in the following table configure the queuing characteristics for a class within CBWFQ.

Command	Description
bandwidth {*bandwidth* | **percent** *percent*}	Assigns a bandwidth to the traffic class for CQ-like queuing; can be assigned as an amount of bandwidth or a percent of interface bandwidth.
Fair-queue	Enables WFQ for the traffic class.
no fair-queue	Disables WFQ and enables FIFO queuing for the class.
priority *bandwidth*	Assigns a maximum bandwidth for strict priority queuing; this is LLQ.

After a policy map is created and configured, use the following command to apply it to an interface:

```
service-policy output policy-map
```

The following is an example of a CBWFQ configuration that creates a strict priority queue (LLQ) for Telnet traffic, assigns 64 kbps for IPX traffic, and uses WFQ for any unclassified traffic types:

```
Router(config)# access-list 100 permit tcp any any eq 23
Router(config)# access-list 100 permit tcp any eq 23 any
Router(config)# class-map MC-TELNET
Router(config-cmap)# match access-group 100
Router(config-cmap)# class-map MC-IPX
Router(config-cmap)# match protocol ipx
Router(config-cmap)# policy-map PM-E0-OUT
Router(config-pmap)# class MC-TELNET
Router(config-pmap-c)# priority 64
Router(config-pmap-c)# class MC-IPX
Router(config-pmap-c)# bandwidth 64
Router(config-pmap-c)# class class-default
Router(config-pmap-c)# fair-queue
Router(config-pmap-c)# interface Ethernet 0
Router(config-if)# service-policy output PM-E0-OUT
```

Configuring CQ

During the configuration of CQ, queue lists are created and then applied to an interface. To create, customize, and apply a custom queue list, use the commands shown in the following table.

Command	Description
queue-list *list-number* **default** *queue-number*	Assigns the default queue; traffic not explicitly place in a queue ends up here.
queue-list *list-number* **interface** *type number queue-number*	Assigns traffic to a specific queue based on the input interface.
queue-list *list-number* **protocol** *protocol queue-number* [*queue-keyword value*]	Assigns traffic to a specific queue based on the protocol, or several protocol keywords, including • fragments • list • gt • lt • tcp • udp
queue-list *list-number* **queue** *queue-number* **byte-count** *byte-count*	Limits how many bytes are emptied from the queue before moving on to the next queue; entire packets are taken until byte-count is exceeded.

Command	Description
queue-list *list-number* **queue** *queue-number* **limit** *queue-limit*	Configures the size of the queue in packets.
custom-queue-list *queue-list*	Applies a queue list an interface.

The following configuration example configures CQ to give 25 percent of available bandwidth to HTTP and HTTPS traffic:

```
Router(config)# access-list 100 permit tcp any any eq 80
Router(config)# access-list 100 permit tcp any eq 80 any
Router(config)# access-list 100 permit tcp any any eq 443
Router(config)# access-list 100 permit tcp any eq 443 any
Router(config)# queue-list 1 protocol ip 1 list 100
Router(config)# queue-list 1 default 2
Router(config)# queue-list 1 queue 1 byte-count 1500
Router(config)# queue-list 1 queue 2 byte-count 4500
Router(config)# interface Serial 0
Router(config-if)# custom-queue-list 1
```

Enabling FIFO Queuing

FIFO queuing is the default queuing method for interfaces faster than 2 Mbps. To enable FIFO queuing for other interfaces, use the following interface or policy map class configuration command:

```
no fair-queue
```

Configuring PQ

Similar to CQ, a PQ configuration entails the creation of a queue list and then application of that queue list to an interface. The commands to create and apply priority queue lists are detailed in the following table.

Command	Description
priority-list *list-number* **default** {**high** | **medium** | **normal** | **low**}	Configures the default queue for traffic not explicitly placed into another queue
priority-list *list-number* **interface** *type number* {**high** | **medium** | **normal** | **low**}	Places packets into a specific queue based on the input interface
priority-list *list-number* **protocol** *protocol* {**high** | **medium** | **normal** | **low**} [*keyword value*]	Assigns traffic to a specific queue based on the protocol, or several protocol keywords, including • fragments • list • gt • lt • tcp • udp
priority-list *list-number* **queue-list** [*high-limit* [*medium-limit* [*normal-limit* [*low-limit*]]]]	Specifies how many packets can be held in each of the queues
priority-group *list-number*	Assigns a priority queue list to an interface

The following configuration example creates a PQ configuration that gives strict priority to IPX traffic. The queue size is doubled for the high priority queue as well:

```
Router(config)# priority-list 1 protocol ipx high
Router(config)# priority-list 1 default normal
Router(config)# priority-list 1 queue-list 40 40 60 80
Router(config)# interface Serial 0
Router(config)# priority-group 1
```

Enabling and Tuning WFQ

WFQ is the default for serial interfaces slower than 2 Mbps. To enable WFQ for other interfaces or for traffic classes within CBWFQ, use the following configuration command:

```
fair-queue
```

The complete syntax for this command is

```
fair-queue [congestive-discard-threshold [dynamic-queues [reservable-queues]]]
```

Using the extended syntax lets you configure the number and size of the queues used by WFQ.

Monitoring and Verifying of a Queuing Configuration

Use the following commands to view the status a queuing configuration:

- **show queue** *type number*
- **show queueing** [**interface** *type number* | **custom** | **priority** | **fair** [**interface** *type number*]]
- **show policy-map** [*policy-map*]
- **show policy-map** *policy-map* **class** *class-map*
- **show policy-map interface** *type number* {**input** | **output**}

Configuring Network Address Translation (NAT)

Key NAT Terminology

Term	Definition
Inside global address	This is the inside IP address as seen outside the network.
Inside local address	This is the inside IP address as seen inside the network.
Outside global address	This is the outside IP address as seen on the outside network.
Outside local address	This is the outside IP address as seen inside the network.
Rotary pool	Used in TCP load distribution, the rotary pool is the group of internal servers that receive connections.

Types of NAT

- One-to-one NAT
- Many-to-many NAT
- Many-to-one NAT (also known as overload, Port Address Translation [PAT])
- TCP load distribution
- Translation of overlapping address spaces

Configuring NAT

Before a NAT configuration functions, the inside and outside of the network must be defined using the following interface configuration commands:

```
ip nat inside
```

and

```
ip nat outside
```

A simple one-to-one NAT can be configured with the command:

```
ip nat inside source static inside-ip outside-ip
```

The following command configures a range of outside addresses for use with many-to-many NAT or translation of overlapping address spaces:

```
ip nat pool pool-name start-address end-address netmask netmask
```

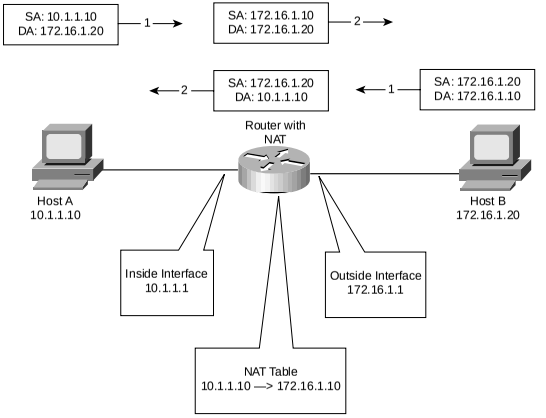
The syntax of the following **ip nat inside source list** command indicates which addresses on the inside interface should be processed by NAT and to bind those addresses to an outside pool:

```
ip nat inside source list acl-number pool pool-name [overload]
```

The **ip nat inside destination** command is used with TCP load distribution and has the following syntax:

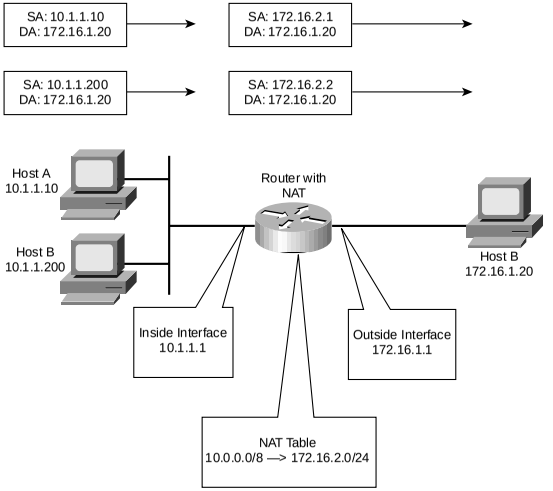
```
ip nat inside destination list acl-number pool pool-name rotary
```

The following is an example of a one-to-one NAT configuration that maps 10.1.1.10 to 172.16.1.10, as shown in the figure.



```
Router(config)# interface Ethernet 0
Router(config-if)# ip address 10.1.1.1 255.255.255.0
Router(config-if)# ip nat inside
Router(config-if)# interface Ethernet 1
Router(config-if)# ip address 172.16.1.1 255.255.255.0
Router(config-if)# ip nat outside
Router(config-if)# exit
Router(config)# ip nat inside source static 10.1.1.10 172.16.1.10
```

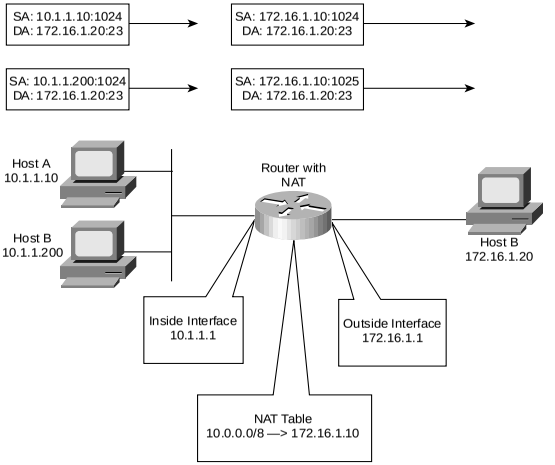
The following is an example of a many-to-many NAT configuration that translates 10.0.0.0 through 10.255.255.255 to 172.16.2.0 through 172.16.2.255, as shown in the figure.




```
Router(config)# interface Ethernet 0
Router(config-if)# ip address 10.1.1.1 255.255.255.0
Router(config-if)# ip nat inside
Router(config-if)# interface Ethernet 1
Router(config-if)# ip address 172.16.1.1 255.255.255.0
Router(config-if)# ip nat outside
```

```
Router(config-if)# exit
Router(config)# access-list 1 permit 10.0.0.0 0.255.255.255
Router(config)# ip nat pool NAT-POOL 172.16.2.0 172.16.2.255 netmask
  255.255.255.0
Router(config)# ip nat inside source list 1 pool NAT-POOL
```

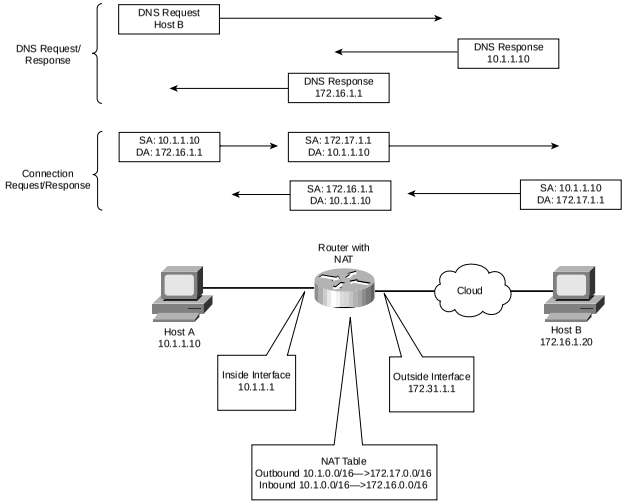
The following is an example of many-to-one NAT that translates addresses 10.0.0.0 through 10.255.255.255 to the address 172.16.1.10, as illustrated in the figure.




```
Router(config)# interface Ethernet 0
Router(config-if)# ip address 10.1.1.1 255.255.255.0
Router(config-if)# ip nat inside
Router(config-if)# interface Ethernet 1
Router(config-if)# ip address 172.16.1.1 255.255.255.0
Router(config-if)# ip nat outside
Router(config-if)# exit
Router(config)# access-list 1 permit 10.0.0.0 0.255.255.255
Router(config)# ip nat pool NAT-POOL 172.16.1.10 172.16.1.10 netmask
  255.255.255.0
Router(config)# ip nat inside source list 1 pool NAT-POOL overload
```

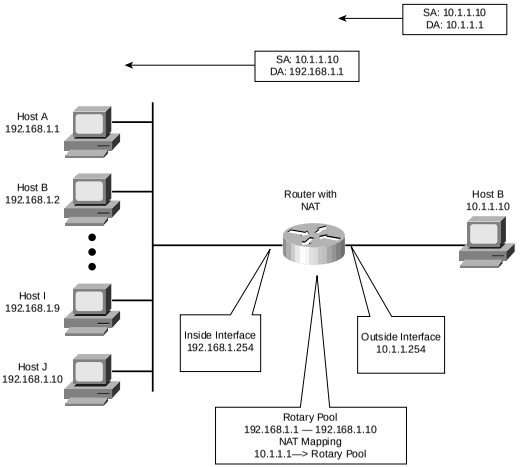
The following is an example of a configuration that leverages NAT to overcome the connectivity problems where two networks have overlapping address spaces. In this configuration, two networks have the 10.1.0.0/16 network. Traffic sent from inside to outside from the 10.1.0.0/16 network is translated to 172.17.0.0/16. At the same time, traffic from outside to inside from 10.1.0.0/16 (the other one) is translated to 172.16.0.0/16.

For this solution to work, the router performing NAT must also modify certain DNS responses. Specifically, the router passively waits for DNS responses and alters their contents to match the current NAT translations. As shown in the following figure, the router alters the DNS response, which includes 10.1.1.10 (a duplicate address) with the address in the NAT table of 172.16.1.1.




```
Router(config)# interface Ethernet 0
Router(config-if)# ip address 10.1.1.1 255.255.255.0
Router(config-if)# ip nat inside
Router(config-if)# interface Ethernet 1
Router(config-if)# ip address 172.31.1.1 255.255.255.0
Router(config-if)# ip nat outside
Router(config-if)# exit
Router(config)# access-list 1 permit 10.1.0.0 0.0.255.255
Router(config)# ip nat pool NAT-POOL-IN 172.16.0.0 172.16.0.0 netmask
  255.255.0.0
Router(config)# ip nat pool NAT-POOL-OUT 172.17.0.0 172.17.0.0 netmask
  255.255.0.0
Router(config)# ip nat inside source list 1 pool NAT-POOL-OUT
Router(config)# ip nat outside source list 1 pool NAT-POOL-IN
```

NAT can also be used to load balance TCP connections. The example that follows maps the virtual server IP address 10.1.1.1 to the servers 192.168.1.1 through 192.168.1.10.




```
Router(config)# interface Ethernet 0
Router(config-if)# ip address 192.168.1.254 255.255.255.0
Router(config-if)# ip nat inside
Router(config-if)# interface Ethernet 1
Router(config-if)# ip address 10.1.1.254 255.255.255.0
```

```
Router(config-if)# ip nat outside
Router(config-if)# exit
Router(config)# access-list 1 permit 10.1.1.1 0.0.0.0
Router(config)# ip nat pool NAT-POOL 192.168.1.1 192.168.1.10 netmask
  255.255.255.0 type rotary
Router(config)# ip nat inside destination list 1 pool NAT-POOL
```

Tuning NAT Timeouts

The command used to tune NAT-related timeouts differs depending on the configuration. To change the timeout related to simple one-to-one NAT translations use the following:

ip nat translation timeout *timeout-seconds*

When your configuration includes overload NAT or PAT, use the following commands to tune protocol specific timeouts:

- **ip nat translation udp-timeout** {*seconds* | **never**}
- **ip nat translation dns-timeout** {*seconds* | **never**}
- **ip nat translation tcp-timeout** {*seconds* | **never**}
- **ip nat translation icmp-timeout** {*seconds* | **never**}
- **ip nat translation pptp-timeout** {*seconds* | **never**}

Verifying the NAT Configuration

The following commands help you determine if your NAT configuration is operating properly:

- **clear ip nat translation** {* | [**inside** *global-ip local-ip*] [**outside** *local-ip global-ip*]}
- **clear ip nat translation** *protocol* **inside** *global-ip global-port local-ip local-port* [**outside** *local-ip global-ip*]
- **debug ip nat** [**detailed** | *acl-number*]
- **show ip nat statistics**
- **show ip nat translation** [**verbose**]