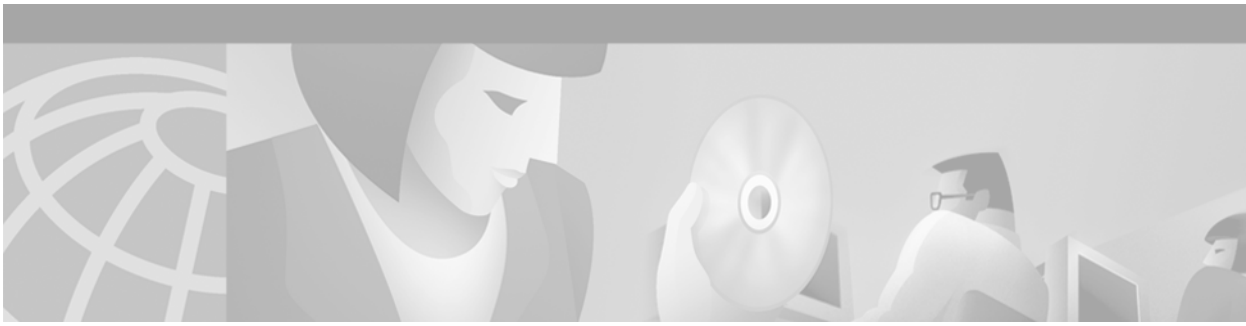

Configuring Asynchronous Lines and Interfaces

This chapter describes how to configure asynchronous line features in the following main sections:

- How to Configure Asynchronous Interfaces and Lines
- How to Configure Other Asynchronous Line and Interface Features
- Configuration Examples for Asynchronous Interfaces and Lines

Perform these tasks, as required, for your particular network.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the "Identifying Supported Platforms" section in the "Using Cisco IOS Software" chapter.

For a complete description of the commands in this chapter, refer to the *Cisco IOS Dial Technologies Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

How to Configure Asynchronous Interfaces and Lines

To configure an asynchronous interface, perform the tasks described in the following sections as required:

- Configuring a Typical Asynchronous Interface (As required)
- Creating a Group Asynchronous Interface (As required)
- Configuring Asynchronous Rotary Line Queueing (As required)
- Configuring Autoselect (As required)

Configuring a Typical Asynchronous Interface

To configure an asynchronous interface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface async number`	Brings up a single asynchronous interface and enters interface configuration mode.
Step 2	`Router(config-if)# description description`	Provides a description for the interface.
Step 3	`Router(config-if)# ip address address mask`	Specifies an IP address.
Step 4	`Router(config-if)# encapsulation ppp`	Enables PPP to run on the asynchronous interfaces in the group.
Step 5	`Router(config-if)# async default routing`	Enables the router to pass routing updates to other routers over the AUX port configured as an asynchronous interface.
Step 6	`Router(config-if)# async mode dedicated`	Places a line into dedicated asynchronous mode using Serial Line Internet Protocol (SLIP) or PPP encapsulation.
Step 7	`Router(config-if)# dialer in-band`	Specifies that dial-on-demand routing (DDR) is to be supported.
Step 8	`Router(config-if)# dialer map protocol next-hop-address`	Configures a serial interface to call one or multiple sites or to receive calls from multiple sites.
Step 9	`Router(config-if)# dialer-group`	Controls access by configuring an interface to belong to a specific dialing group.
Step 10	`Router(config-if)# ppp authentication chap pap list-name`	Enables Challenge Handshake Authentication Protocol (CHAP) and Password Authentication Protocol (PAP) authentication on the interface. Replace the *list-name* variable with a specified authentication list name.[1]
Step 11	`Router(config-if)# exit`	Return to global configuration mode.

1. To create a string used to name the following list of authentication methods tried when a user logs in, refer to the **aaa authentication ppp** command. Authentication methods include RADIUS, TACACS+, and Kerberos.

The "Interface and Line Configuration Examples" and "Asynchronous Interface As the Only Network Interface Example" sections later in this chapter contain examples of how to configure an asynchronous interface.

Monitoring and Maintaining Asynchronous Connections

This section describes the following monitoring and maintenance tasks that you can perform on asynchronous interfaces:

- Monitoring and maintaining asynchronous activity
- Debugging asynchronous interfaces
- Debugging PPP

To monitor and maintain asynchronous activity, use the following commands in privileged EXEC mode as needed:

Command	Purpose
`Router# clear line line-number`	Returns a line to its idle state.
`Router# show async bootp`	Displays parameters that have been set for extended BOOTP requests.
`Router# show async status`	Displays statistics for asynchronous interface activity.
`Router# show line [line-number]`	Displays the status of asynchronous line connections.

To debug asynchronous interfaces, use the following debug command in privileged EXEC mode:

Command	Purpose
`Router# debug async {framing | state | packets}`	Displays errors, changes in interface state, and log input and output.

To debug PPP links, use the following debug commands in privileged EXEC mode as needed:

Command	Purpose
`Router# debug ppp negotiation`	Enables debugging of PPP protocol negotiation process.
`Router# debug ppp error`	Displays PPP protocol errors.
`Router# debug ppp packet`	Displays PPP packets sent and received.
`Router# debug ppp chap`	Displays errors encountered during remote or local system authentication.

Creating a Group Asynchronous Interface

Create a group asynchronous interface to project a set of core protocol characteristics to a range of asynchronous interfaces. Configuring the asynchronous interfaces as a group saves you time. Analog modem calls cannot enter the access server without this configuration.

To configure a group asynchronous interface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface async number`	Brings up a single asynchronous interface and enters interface configuration mode.
Step 2	`Router(config-if)# ip unnumbered loopback number`	Configures the asynchronous interfaces as unnumbered and assigns the IP address of the loopback interface to them to conserve IP addresses.[1]
Step 3	`Router(config-if)# encapsulation ppp`	Enables PPP to run on the asynchronous interfaces in the group.

	Command	Purpose
Step 4	`Router(config-if)# async mode interactive`	Configures interactive mode on the asynchronous interface.
Step 5	`Router(config-if)# ppp authentication chap pap` *list-name*	Enables CHAP and PAP authentication on the interface. Replace the *list-name* variable with a specified authentication list name.[2]
Step 6	`Router(config-if)# peer default ip address pool` *poolname*	Assigns dial-in clients IP addresses from an address pool.[3]
Step 7	`Router(config-if)# no cdp enable`	Disables the Cisco Discovery Protocol (CDP) on the interface.
Step 8	`Router(config-if)# group-range` *low-end-of-range high-end-of-range*	Specifies the range of asynchronous interfaces to include in the group, which is usually equal to the number of modems you have in the access server.
Step 9	`Router(config-if)# exit`	Returns to global configuration mode.

1. You can also specify the Ethernet interface to conserver address space. In this case, enter the **ip unnumbered ethernet 0** command.
2. To create a string used to name the following list of authentication methods tried when a user logs in, refer to the **aaa authentication ppp** command. Authentication methods include RADIUS, TACACS+, and Kerberos.
3. To create an IP address pool, refer to the **ip local pool** global configuration command.

The "Group and Member Asynchronous Interface Examples" section later in this chapter contains an example of how to configure a group interface.

Verifying the Group Interface Configuration

To verify the group interface configuration and check if one of the asynchronous interfaces is up, use the **show interface async** command:

```
Router# show interface async 1

Async1 is up, line protocol is up
modem(slot/port)=1/0, csm_state(0x00000204)=CSM_IC4_CONNECTED, bchan_num=18
modem_status(0x0002): VDEV_STATUS_ACTIVE_CALL.

  Hardware is Async Serial
  Interface is unnumbered.  Using address of FastEthernet0 (10.1.1.10)
  MTU 1500 bytes, BW 115 Kbit, DLY 100000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set, keepalive not set
  DTR is pulsed for 5 seconds on reset
  LCP Open
  Open: IPCP
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/5, 0 drops; input queue 1/5, 0 drops
  5 minute input rate 37000 bits/sec, 87 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     31063 packets input, 1459806 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     33 packets output, 1998 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

If you are having trouble, enter one of the following **debug** commands and then send a call into the access server. Interpret the output and make configuration changes accordingly.

- **undebug all**
- **debug ppp negotiation**
- **debug ppp authentication**
- **debug modem**
- **debug ip peer**

```
Router# undebug all
All possible debugging has been turned off
Router# debug ppp negotiation
PPP protocol negotiation debugging is on
Router# debug ppp authentication
PPP authentication debugging is on
Router# debug modem
Modem control/process activation debugging is on
Router# debug ip peer
IP peer address activity debugging is on
Router# show debug
General OS:
  Modem control/process activation debugging is on
Generic IP:
  IP peer address activity debugging is on
PPP:
  PPP authentication debugging is on
  PPP protocol negotiation debugging is on
Router#
*Mar  1 21:34:56.958: tty4: DSR came up
*Mar  1 21:34:56.962: tty4: Modem: IDLE->READY
*Mar  1 21:34:56.970: tty4: EXEC creation
*Mar  1 21:34:56.978: tty4: set timer type 10, 30 seconds
*Mar  1 21:34:59.722: tty4: Autoselect(2) sample 7E
*Mar  1 21:34:59.726: tty4: Autoselect(2) sample 7EFF
*Mar  1 21:34:59.730: tty4: Autoselect(2) sample 7EFF7D
*Mar  1 21:34:59.730: tty4: Autoselect(2) sample 7EFF7D23
*Mar  1 21:34:59.734: tty4 Autoselect cmd: ppp negotiate
*Mar  1 21:34:59.746: tty4: EXEC creation
*Mar  1 21:34:59.746: tty4: create timer type 1, 600 seconds
*Mar  1 21:34:59.786: ip_get_pool: As4: using pool default
*Mar  1 21:34:59.790: ip_get_pool: As4: returning address = 172.20.1.101
*Mar  1 21:34:59.794: tty4: destroy timer type 1 (OK)
*Mar  1 21:34:59.794: tty4: destroy timer type 0
*Mar  1 21:35:01.798: %LINK-3-UPDOWN: Interface Async4, changed state to up
*Mar  1 21:35:01.834: As4 PPP: Treating connection as a dedicated line
*Mar  1 21:35:01.838: As4 PPP: Phase is ESTABLISHING, Active Open
*Mar  1 21:35:01.842: As4 LCP: O CONFREQ [Closed] id 1 len 25
*Mar  1 21:35:01.846: As4 LCP:    ACCM 0x000A0000 (0x0206000A0000)
*Mar  1 21:35:01.850: As4 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  1 21:35:01.854: As4 LCP:    MagicNumber 0x64E923A8 (0x050664E923A8)
*Mar  1 21:35:01.854: As4 LCP:    PFC (0x0702)
*Mar  1 21:35:01.858: As4 LCP:    ACFC (0x0802)
*Mar  1 21:35:02.718: As4 LCP: I CONFREQ [REQsent] id 3 len 23
*Mar  1 21:35:02.722: As4 LCP:    ACCM 0x000A0000 (0x0206000A0000)
*Mar  1 21:35:02.726: As4 LCP:    MagicNumber 0x00472467 (0x050600472467)
*Mar  1 21:35:02.726: As4 LCP:    PFC (0x0702)
*Mar  1 21:35:02.730: As4 LCP:    ACFC (0x0802)
*Mar  1 21:35:02.730: As4 LCP:    Callback 6  (0x0D0306)
*Mar  1 21:35:02.738: As4 LCP: O CONFREJ [REQsent] id 3 len 7
*Mar  1 21:35:02.738: As4 LCP:    Callback 6  (0x0D0306)
*Mar  1 21:35:02.850: As4 LCP: I CONFREQ [REQsent] id 4 len 20
```

```
*Mar  1 21:35:02.854: As4 LCP:    ACCM 0x000A0000 (0x0206000A0000)
*Mar  1 21:35:02.854: As4 LCP:    MagicNumber 0x00472467 (0x050600472467)
*Mar  1 21:35:02.858: As4 LCP:    PFC (0x0702)
*Mar  1 21:35:02.858: As4 LCP:    ACFC (0x0802)
*Mar  1 21:35:02.862: As4 LCP: O CONFACK [REQsent] id 4 len 20
*Mar  1 21:35:02.866: As4 LCP:    ACCM 0x000A0000 (0x0206000A0000)
*Mar  1 21:35:02.870: As4 LCP:    MagicNumber 0x00472467 (0x050600472467)
*Mar  1 21:35:02.870: As4 LCP:    PFC (0x0702)
*Mar  1 21:35:02.874: As4 LCP:    ACFC (0x0802)
*Mar  1 21:35:03.842: As4 LCP: TIMEout: State ACKsent
*Mar  1 21:35:03.842: As4 LCP: O CONFREQ [ACKsent] id 2 len 25
*Mar  1 21:35:03.846: As4 LCP:    ACCM 0x000A0000 (0x0206000A0000)
*Mar  1 21:35:03.850: As4 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  1 21:35:03.854: As4 LCP:    MagicNumber 0x64E923A8 (0x050664E923A8)
*Mar  1 21:35:03.854: As4 LCP:    PFC (0x0702)
*Mar  1 21:35:03.858: As4 LCP:    ACFC (0x0802)
*Mar  1 21:35:03.962: As4 LCP: I CONFACK [ACKsent] id 2 len 25
*Mar  1 21:35:03.966: As4 LCP:    ACCM 0x000A0000 (0x0206000A0000)
*Mar  1 21:35:03.966: As4 LCP:    AuthProto CHAP (0x0305C22305)
*Mar  1 21:35:03.970: As4 LCP:    MagicNumber 0x64E923A8 (0x050664E923A8)
*Mar  1 21:35:03.974: As4 LCP:    PFC (0x0702)
*Mar  1 21:35:03.974: As4 LCP:    ACFC (0x0802)
*Mar  1 21:35:03.978: As4 LCP: State is Open
*Mar  1 21:35:03.978: As4 PPP: Phase is AUTHENTICATING, by this end
*Mar  1 21:35:03.982: As4 CHAP: O CHALLENGE id 1 len 26 from "nas-1"
*Mar  1 21:35:04.162: As4 CHAP: I RESPONSE id 1 len 26 from "krist"
*Mar  1 21:35:04.170: As4 AUTH: Started process 0 pid 47
*Mar  1 21:35:04.182: As4 CHAP: O SUCCESS id 1 len 4
*Mar  1 21:35:04.186: As4 PPP: Phase is UP
*Mar  1 21:35:04.190: As4 IPCP: O CONFREQ [Not negotiated] id 1 len 10
*Mar  1 21:35:04.194: As4 IPCP:    Address 172.20.1.2 (0x0306AC140102)
*Mar  1 21:35:04.202: As4 CDPCP: O CONFREQ [Closed] id 1 len 4
*Mar  1 21:35:04.282: As4 IPCP: I CONFREQ [REQsent] id 1 len 40
*Mar  1 21:35:04.282: As4 IPCP:    CompressType VJ 15 slots CompressSlotID (0x02
06002D0F01)
*Mar  1 21:35:04.286: As4 IPCP:    Address 0.0.0.0 (0x030600000000)
*Mar  1 21:35:04.290: As4 IPCP:    PrimaryDNS 0.0.0.0 (0x810600000000)
*Mar  1 21:35:04.294: As4 IPCP:    PrimaryWINS 0.0.0.0 (0x820600000000)
*Mar  1 21:35:04.298: As4 IPCP:    SecondaryDNS 0.0.0.0 (0x830600000000)
*Mar  1 21:35:04.302: As4 IPCP:    SecondaryWINS 0.0.0.0 (0x840600000000)
*Mar  1 21:35:04.306: As4 IPCP: O CONFREJ [REQsent] id 1 len 10
*Mar  1 21:35:04.310: As4 IPCP:    CompressType VJ 15 slots CompressSlotID (0x02
06002D0F01)
*Mar  1 21:35:04.314: As4 CCP: I CONFREQ [Not negotiated] id 1 len 15
*Mar  1 21:35:04.318: As4 CCP:    MS-PPC supported bits 0x00000001 (0x1206000000
01)
*Mar  1 21:35:04.318: As4 CCP:    Stacker history 1 check mode EXTENDED (0x11050
00104)
*Mar  1 21:35:04.322: As4 LCP: O PROTREJ [Open] id 3 len 21 protocol CCP
*Mar  1 21:35:04.326: As4 LCP:  (0x80FD0101000F12060000000111050001)
*Mar  1 21:35:04.330: As4 LCP:  (0x04)
*Mar  1 21:35:04.334: As4 IPCP: I CONFACK [REQsent] id 1 len 10
*Mar  1 21:35:04.338: As4 IPCP:    Address 172.20.1.2 (0x0306AC140102)
*Mar  1 21:35:04.342: As4 LCP: I PROTREJ [Open] id 5 len 10 protocol CDPCP (0x82
0701010004)
*Mar  1 21:35:04.342: As4 CDPCP: State is Closed
*Mar  1 21:35:05.186: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async4, ch
anged state to up
*Mar  1 21:35:05.190: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:35:05.190: As4 PPP: Trying to negotiate NCP for Link cdp
*Mar  1 21:35:05.194: As4 CDPCP: State is Closed
*Mar  1 21:35:05.198: As4 CDPCP: TIMEout: State Closed
*Mar  1 21:35:05.202: As4 CDPCP: State is Listen
*Mar  1 21:35:06.202: As4 IPCP: TIMEout: State ACKrcvd
```

```
*Mar  1 21:35:06.206: As4 IPCP: O CONFREQ [ACKrcvd] id 2 len 10
*Mar  1 21:35:06.206: As4 IPCP:    Address 172.20.1.2 (0x0306AC140102)
*Mar  1 21:35:06.314: As4 IPCP: I CONFACK [REQsent] id 2 len 10
*Mar  1 21:35:06.318: As4 IPCP:    Address 172.20.1.2 (0x0306AC140102)
*Mar  1 21:35:07.274: As4 IPCP: I CONFREQ [ACKrcvd] id 2 len 34
*Mar  1 21:35:07.278: As4 IPCP:    Address 0.0.0.0 (0x030600000000)
*Mar  1 21:35:07.282: As4 IPCP:    PrimaryDNS 0.0.0.0 (0x810600000000)
*Mar  1 21:35:07.286: As4 IPCP:    PrimaryWINS 0.0.0.0 (0x820600000000)
*Mar  1 21:35:07.286: As4 IPCP:    SecondaryDNS 0.0.0.0 (0x830600000000)
*Mar  1 21:35:07.290: As4 IPCP:    SecondaryWINS 0.0.0.0 (0x840600000000)
*Mar  1 21:35:07.294: As4 IPCP: O CONFNAK [ACKrcvd] id 2 len 34
*Mar  1 21:35:07.298: As4 IPCP:    Address 172.20.1.101 (0x0306AC140165)
*Mar  1 21:35:07.302: As4 IPCP:    PrimaryDNS 172.20.5.100 (0x8106AC140564)
*Mar  1 21:35:07.306: As4 IPCP:    PrimaryWINS 172.20.5.101 (0x8206AC140565)
*Mar  1 21:35:07.310: As4 IPCP:    SecondaryDNS 172.20.6.100 (0x8306AC140664)
*Mar  1 21:35:07.314: As4 IPCP:    SecondaryWINS 172.20.6.101 (0x8406AC140665)
*Mar  1 21:35:07.426: As4 IPCP: I CONFREQ [ACKrcvd] id 3 len 34
*Mar  1 21:35:07.430: As4 IPCP:    Address 172.20.1.101 (0x0306AC140165)
*Mar  1 21:35:07.434: As4 IPCP:    PrimaryDNS 172.20.5.100 (0x8106AC140564)
*Mar  1 21:35:07.438: As4 IPCP:    PrimaryWINS 172.20.5.101 (0x8206AC140565)
*Mar  1 21:35:07.442: As4 IPCP:    SecondaryDNS 172.20.6.100 (0x8306AC140664)
*Mar  1 21:35:07.446: As4 IPCP:    SecondaryWINS 172.20.6.101 (0x8406AC140665)
*Mar  1 21:35:07.446: ip_get_pool: As4: validate address = 172.20.1.101
*Mar  1 21:35:07.450: ip_get_pool: As4: using pool default
*Mar  1 21:35:07.450: ip_get_pool: As4: returning address = 172.20.1.101
*Mar  1 21:35:07.454: set_ip_peer_addr: As4: address = 172.20.1.101 (3) is redun
dant
*Mar  1 21:35:07.458: As4 IPCP: O CONFACK [ACKrcvd] id 3 len 34
*Mar  1 21:35:07.462: As4 IPCP:    Address 172.20.1.101 (0x0306AC140165)
*Mar  1 21:35:07.466: As4 IPCP:    PrimaryDNS 172.20.5.100 (0x8106AC140564)
*Mar  1 21:35:07.470: As4 IPCP:    PrimaryWINS 172.20.5.101 (0x8206AC140565)
*Mar  1 21:35:07.474: As4 IPCP:    SecondaryDNS 172.20.6.100 (0x8306AC140664)
*Mar  1 21:35:07.474: As4 IPCP:    SecondaryWINS 172.20.6.101 (0x8406AC140665)
*Mar  1 21:35:07.478: As4 IPCP: State is Open
*Mar  1 21:35:07.490: As4 IPCP: Install route to 172.20.1.101
*Mar  1 21:35:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:36:12.614: tty0: timer type 1 expired
*Mar  1 21:36:12.614: tty0:  Exec timer (continued)
*Mar  1 21:36:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:37:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:38:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:39:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:40:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:41:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:42:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
*Mar  1 21:43:25.038: As4 PPP: Unsupported or un-negotiated protocol. Link cdp
```

Configuring Asynchronous Rotary Line Queueing

The Cisco IOS Asynchronous Rotary Line Queueing feature allows Telnet connection requests to busy asynchronous rotary groups to be queued so that users automatically obtain the next available line, rather than needing to try repeatedly to open a Telnet connection. The Cisco IOS software sends a periodic message to the user to update progress in the connection queue.

This feature allows users to make effective use of the asynchronous rotary groups on a Cisco router to access legacy mainframes or other serial devices with a limited number of asynchronous ports that might be used by a large number of users. Users that are unable to make a Telnet connection on the first attempt are assured of eventual success in an orderly process. They are no longer required to guess when a line might be available and to retry manually again and again.

Connections are authenticated using the method specified for the line configurations for the asynchronous rotary group. If a connection is queued, authentication is done prior to queueing and no authentication is done when the connection is later established.

Make sure you comply with the following requirements when configuring asynchronous rotary line queueing:

- Configure more virtual terminal lines than will ever be used by waiting asynchronous rotary connection attempts. Even when the queue is at its maximum, there must be at least one virtual terminal line available so that system operators or network administrators can use Telnet to access the router to show, debug, or configure system performance.
- When adding lines to a rotary group, all lines must be either queued or not queued. A mixture of queued and unenqueued lines in the same rotary group is not supported and can result in unexpected behavior.
- All lines within a queued rotary group need to use the same authentication method. Using different authentication methods within the same rotary group can result in unexpected behavior.

To configure asynchronous rotary line queueing, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router (config)# line [aux | console | tty | vty] line-number [ending-line-number]`	Starts line configuration mode on the line type and numbers specified.
Step 2	`Router(config-line)# rotary group [queued | round-robin]`	Enables asynchronous rotary line queueing on the designated line or group of lines. The optional **round-robin** keyword selects a round-robin port selection algorithm instead of the default (**queued**) linear port selection algorithm.

See the "Rotary Group Examples" section for configuration examples.

Verifying Asynchronous Rotary Line Queueing

To verify operation of asynchronous rotary line queueing, perform the following tasks:

- Use the **show line** command in EXEC mode to check the status of the vty lines.
- Use the **show line async-queue** command in EXEC mode to check the status of queued connection requests.

Troubleshooting Asynchronous Rotary Lines

If asynchronous rotary line queueing is not operating correctly, use the following **debug** commands in privileged EXEC mode to determine where the problem may lie:

- **debug async async-queue**
- **debug ip tcp transactions**
- **debug modem**

Refer to the *Cisco IOS Debug Command Reference* for information about these commands.

Monitoring and Maintaining Asynchronous Rotary Line Queues

To display queued lines and to remove lines from the queue, use the following commands in EXEC mode as needed:

Command	Purpose
`Router# show line async-queue rotary-group`	Displays which lines are queued.
`Router# clear line async-queue rotary-group`	Clears all rotary queues or the specified rotary queue. If the *rotary-group* argument is not specified, all rotary queues are removed.

Configuring Autoselect

Autoselect is used by the access server to sense the protocol being received on an incoming line and to launch the appropriate protocol. Autoselect can be used for AppleTalk Remote Access (ARA), PPP, or SLIP.

When using Autoselect, "login" authentication is bypassed, so if security is required, it must be performed at the protocol level, that is, the AppleTalk Remote Access Protocol (ARAP) or PPP authentication. SLIP does not offer protocol layer authentication.

To configure the Cisco IOS software to allow an ARA, PPP, or SLIP session to start automatically, use the following command in line configuration mode:

Command	Purpose
`Router(config-line)# autoselect {arap | ppp | slip | during login}`	Configures a line to automatically start an ARA, PPP, or SLIP session.

The **autoselect** command enables the Cisco IOS software to start a process automatically when a start character is received.

The **autoselect** command bypasses the login prompt and enables the specified session to begin automatically. However, when the **autoselect** command is entered with the **during login** keyword, the username or password prompt appears without the need to press the Return key; thus "login" users will get a prompt right away without needing to press the Return key. While the username or password prompt is displayed, you can choose either to answer these prompts or to send packets from an autoselected protocol.

Normally a router avoids line and modem noise by clearing the initial data received within the first one or two seconds. However, when the autoselect PPP feature is configured, the router flushes characters initially received and then waits for more traffic. This flush causes timeout problems with applications that send only one carriage return. To ensure that the input data sent by a modem or other asynchronous device is not lost after line activation, enter the **flush-at-activation** line configuration command.

Note When the **autoselect** command is used, the activation character should be set to the default Return, and exec-character-bits should be set to 7. If you change these defaults, the application cannot recognize the activation request.

See the "High-Density Dial-In Solution Using Autoselect and EXEC Control Example" section for an example that makes use of the autoselect feature.

Verifying Autoselect PPP

The following trace appears when the **debug modem** and **debug ppp negotiation** commands are enabled. As PPP calls pass through the access server, you should see this output.

When autoselect is used, "login" authentication is bypassed. If security is required, it must be performed at the protocol level (that is, ARAP or PPP authentication). SLIP does not offer protocol layer authentication.

```
22:21:02: TTY1: DSR came up
22:21:02: tty1: Modem: IDLE->READY
22:21:02: TTY1: Autoselect started
22:21:05: TTY1: Autoselect sample 7E
22:21:05: TTY1: Autoselect sample 7EFF
22:21:05: TTY1: Autoselect sample 7EFF7D
22:21:05: TTY1 Autoselect cmd: ppp default
22:21:05: TTY1: EXEC creation
%LINK-3-UPDOWN: Interface Async1, changed state to up
22:21:07: ppp: sending CONFREQ, type = 2 (CI_ASYNCMAP), value = A0000
22:21:07: ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 23BE13AA
22:21:08: PPP Async1: state = REQSENT fsm_rconfack(0xC021): rcvd id 0x11
22:21:08: ppp: config ACK received, type = 2 (CI_ASYNCMAP), value = A0000
22:21:08: ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 23BE13AA
22:21:08: ppp: config ACK received, type = 7 (CI_PCOMPRESSION)
22:21:08: ppp: config ACK received, type = 8 (CI_ACCOMPRESSION)
22:21:08: PPP Async1: received config for type = 0x2 (ASYNCMAP) value = 0x0 acked
22:21:08: PPP Async1: received config for type = 0x5 (MAGICNUMBER) value = 0x2A acked
22:21:08: PPP Async1: received config for type = 0x7 (PCOMPRESSION) acked
22:21:08: PPP Async1: received config for type = 0x8 (ACCOMPRESSION) acked
22:21:08: ipcp: sending CONFREQ, type = 3 (CI_ADDRESS), Address = 172.16.1.1
22:21:08: ppp Async1: ipcp_reqci: rcvd COMPRESSTYPE (rejected) (REJ)
22:21:08: ppp Async1: Negotiate IP address: her address 0.0.0.0 (NAK with address
172.16.1.100) (NAK)
22:21:08: ppp: ipcp_reqci: returning CONFREJ.
22:21:08: PPP Async1: state = REQSENT fsm_rconfack(0x8021): rcvd id 0x9
22:21:08: ipcp: config ACK received, type = 3 (CI_ADDRESS), Address = 172.16.1.1
22:21:08: ppp Async1: Negotiate IP address: her address 0.0.0.0 (NAK with address
172.16.1.100) (NAK)
22:21:08: ppp: ipcp_reqci: returning CONFNAK.
22:21:09: ppp Async1: Negotiate IP address: her address 172.16.1.100 (ACK)
22:21:09: ppp: ipcp_reqci: returning CONFACK.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up
```

Verifying Autoselect ARA

The following trace appears when the **debug modem** and **debug arap internal** commands are enabled. As ARA version 2.0 calls pass through the access server, this output is displayed.

```
20:45:11: TTY3: DSR came up
20:45:11: tty3: Modem: IDLE->READY
20:45:11: TTY3: EXEC creation
20:45:11: TTY3: Autoselect(2) sample 1
20:45:11: TTY3: Autoselect(2) sample 11B
20:45:12: TTY3: Autoselect(2) sample 11B02
20:45:18: ARAP: ---------- SRVRVERSION ----------
20:45:19: ARAP: ---------- ACKing 0 ----------
20:45:19: ARAP: ---------- AUTH_CHALLENGE ----------
20:45:21: ARAP: ---------- ACKing 1 ----------
20:45:21: ARAP: ---------- AUTH_RESPONSE ----------
20:45:21: ARAP: ---------- STARTINFOFROMSERVER ----------
20:45:22: ARAP: ---------- ACKing 2 ----------
22:45:22: ARAP: ---------- ZONELISTINFO ----------
```

```
22:45:22: ARAP: ---------- ZONELISTINFO ----------
22:45:22: ARAP: ---------- ZONELISTINFO ----------
```

The following trace is for ARA version 1.0 calls:

```
22:31:45: TTY1: DSR came up
22:31:45: tty1: Modem: IDLE->READY
22:31:45: TTY1: Autoselect started
22:31:46: TTY1: Autoselect sample 16
22:31:46: TTY1: Autoselect sample 1610
22:31:46: TTY1: Autoselect sample 161002
22:31:47: ARAP: ---------- SRVRVERSION ----------
22:31:47: ARAP: ---------- ACKing 0 ----------
22:31:47: ARAP: ---------- AUTH_CHALLENGE ----------
22:31:47: ARAP: ---------- ACKing 1 ----------
22:31:47: ARAP: ---------- AUTH_RESPONSE ----------
22:31:47: ARAP: ---------- STARTINFOFROMSERVER ----------
22:31:48: ARAP: ---------- ACKing 2 ----------
22:31:48: ARAP: ---------- ZONELISTINFO ----------
22:31:48: ARAP: ---------- ZONELISTINFO ----------
22:31:49: ARAP: ---------- ZONELISTINFO ----------
```

How to Configure Other Asynchronous Line and Interface Features

This section describes the following asynchronous line and interface configurations:

- Configuring the Auxiliary (AUX) Port
- Establishing and Controlling the EXEC Process
- Enabling Routing on Asynchronous Interfaces
- Configuring Dedicated or Interactive PPP and SLIP Sessions
- Conserving Network Addresses
- Using Advanced Addressing Methods for Remote Devices
- Optimizing Available Bandwidth

Configuring the Auxiliary (AUX) Port

The AUX (auxiliary) port is typically configured as an asynchronous serial interface on routers without built-in asynchronous interfaces. To configure the AUX port as an asynchronous interface, configure it first as an auxiliary line with the **line aux 1** global configuration command.

The AUX port sends a data terminal ready (DTR) signal only when a Telnet connection is established. The auxiliary port does not use request to send/clear to send (RTS/CTS) handshaking for flow control. To understand the differences between standard asynchronous interfaces and AUX ports configured as an asynchronous interface, refer to Table 4. To enable the auxiliary port, use the following command in global configuration mode:

Command	Purpose
`Router(config)# line aux line-number`	Enables the auxiliary serial DTE port.

You cannot use the auxiliary (AUX) port as a second console port. To use the AUX port as a console port, you must order a special cable from your technical support personnel.

On an access server, you can configure any of the available asynchronous interfaces (1 through 8, 16, or 48). The auxiliary port (labeled AUX on the back of the product) can also be configured as an asynchronous serial interface, although performance on the AUX port is much slower than on standard asynchronous interfaces and the port does not support some features.

Table 4 illustrates why asynchronous interfaces permit substantially better performance than AUX ports configured as asynchronous interfaces.

Table 4 Differences Between the Asynchronous Port and the Auxiliary (AUX) Port

Feature	Asynchronous Interface	Auxiliary Port
Maximum speed	115200 bps	38400 bps
DMA buffering support[1]	Yes	No
PPP framing on chip[2]	Yes	No
IP fast switching[3]	Yes	No

1. Direct Memory Access (DMA) buffering moves data packets directly to and from system memory without interrupting the main CPU. This process removes overhead from the CPU and increases overall system performance.
2. PPP framing on a hardware chip removes overhead from the CPU on the router, which enables the router to sustain 115200 bps throughput on all asynchronous ports simultaneously.
3. After the destination of the first IP packet is added to the fast switching cache, it is fast switched to and from other interfaces with minimal involvement from the main processor.

On routers without built-in asynchronous interfaces, only the AUX port can be configured as an asynchronous serial interface. To configure the AUX port as an asynchronous interface, you must also configure it as an auxiliary line with the **line aux 1** command. Access servers do not have this restriction. Use the line command with the appropriate line configuration commands for modem control, such as speed.

Only IP packets can be sent across lines configured for SLIP. PPP supports transmission of IP, Internet Packet Exchange (IPX), and AppleTalk packets on an asynchronous serial interface.

See the "Line AUX Configuration Example" section for an example that shows how to configure the AUX port.

Establishing and Controlling the EXEC Process

By default, the Cisco IOS software starts an EXEC process on all lines. However, you can control EXEC processes, as follows:

- Turn the EXEC process on or off. (A serial printer, for example, should not have an EXEC session started.)
- Set the idle terminal timeout interval.

The EXEC command interpreter waits for a specified amount of time to receive user input. If no input is detected, the EXEC facility resumes the current connection. If no connections exist, it returns the terminal to the idle state and disconnects the incoming connection.

To control the EXEC process, use the following commands in line configuration mode:

	Command	Purpose
Step 1	`Router(config-line)# exec`	Turns on EXEC processes.
Step 2	`Router(config-line)# exec-timeout` *minutes* [*seconds*]	Sets the idle terminal timeout interval.

See the "High-Density Dial-In Solution Using Autoselect and EXEC Control Example" section for an example of configuring control over the EXEC process.

Enabling Routing on Asynchronous Interfaces

To route IP packets on an asynchronous interface, use one of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# async dynamic routing`	Configures an asynchronous interface for dynamic routing. Use this command to manually bring up PPP from an EXEC session.
`Router(config-if)# async default routing`	Automatically configures an asynchronous interface for routing. Use this command to enable two routers to communicate over an asynchronous dial backup link.

The **async dynamic routing** command routes IP packets on an asynchronous interface, which permits you to enable the Interior Gateway Routing Protocol (IGRP), Routing Information Protocol (RIP), and Open Shortest Path First (OSPF) routing protocols for use when the user makes a connection using the **ppp** or **slip** EXEC commands. The user must, however, specify the **/routing** keyword at the SLIP or PPP command line.

For asynchronous interfaces in interactive mode, the **async default routing** command causes the **ppp** and **slip** EXEC commands to be interpreted as though the **/route** switch had been included in the command. For asynchronous interfaces in dedicated mode, the **async dynamic routing** command enables routing protocols to be used on the line. Without the **async default routing** command, there is no way to enable the use of routing protocols automatically on a dedicated asynchronous interface.

See the following sections for examples of enabling routing on asynchronous interfaces:

- Asynchronous Interface As the Only Network Interface Example
- IGRP Configuration Example

Configuring Dedicated or Interactive PPP and SLIP Sessions

You can configure one or more asynchronous interfaces on your access server (and one on a router) to be in dedicated network interface mode. In dedicated mode, an interface is automatically configured for SLIP or PPP connections. There is no user prompt or EXEC level, and no end-user commands are required to initiate remote-node connections. If you want a line to be used only for SLIP or PPP connections, configure the line for dedicated mode.

In interactive mode, a line can be used to make any type of connection, depending on the EXEC command entered by the user. For example, depending on its configuration, the line could be used for Telnet or XRemote connections, or SLIP or PPP encapsulation. The user is prompted for an EXEC command before a connection is initiated.

You can configure an asynchronous interface to be in dedicated network mode. When the interface is configured for dedicated mode, the end user cannot change the encapsulation method, address, or other parameters.

To configure an interface for dedicated network mode or to return it to interactive mode, use one of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# async mode dedicated`	Places the line into dedicated asynchronous network mode.
`Router(config-if)# async mode interactive`	Returns the line to interactive mode.

By default, no asynchronous mode is configured. In this state, the line is not available for inbound networking because the SLIP and PPP connections are disabled.

See the "Dedicated Asynchronous Interface Configuration Example" section for an example of how to configure a dedicated asynchronous interface.

Conserving Network Addresses

When asynchronous routing is enabled, you might need to conserve network addresses by configuring the asynchronous interfaces as *unnumbered*. An unnumbered interface does not have an address. Network resources are therefore conserved because fewer network numbers are used and routing tables are smaller.

To configure an unnumbered interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# ip unnumbered type number`	Conserves IP addresses by configuring the asynchronous interfaces as unnumbered, and assigns the IP address of the interface type that you want to leverage.

Whenever the unnumbered interface generates a packet (for example, a routing update), it uses the address of the specified interface as the source address of the IP packet. It also uses the address of the specified interface to determine which routing processes are sending updates over the unnumbered interface.

You can use the IP unnumbered feature even if the system on the other end of the asynchronous link does not support it. The IP unnumbered feature is transparent to the other end of the link because each system bases its routing activities on information in the routing updates it receives and on its own interface address.

See the "Network Address Conservation Using the ip unnumbered Command Example" section for an example of how to conserve network addresses.

Using Advanced Addressing Methods for Remote Devices

You can control whether addressing is dynamic (the user specifies the address at the EXEC level when making the connection) or whether default addressing is used (the address is forced by the system). If you specify dynamic addressing, the router must be in interactive mode and the user will enter the address at the EXEC level.

It is common to configure an asynchronous interface to have a default address and to allow dynamic addressing. With this configuration, the choice between the default address or dynamic addressing is made by the users when they enter the **slip** or **ppp** EXEC command. If the user enters an address, it is used, and if the user enters the **default** keyword, the default address is used.

This section describes the following optional tasks:

- Assigning a Default Asynchronous Address
- Allowing an Asynchronous Address to Be Assigned Dynamically

Assigning a Default Asynchronous Address

To assign a permanent default asynchronous address, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# peer default ip address` *ip-address*	Assigns a default IP address to an asynchronous interface.

Use the **no** form of this command to disable the default address. If the server has been configured to authenticate asynchronous connections, you are prompted for a password after you enter the **slip default** or **ppp default** EXEC command before the line is placed into asynchronous mode.

The assigned default address is implemented when the user enters the **slip default** or **ppp default** EXEC command. The transaction is validated by the TACACS server, when enabled, and the line is put into network mode using the address that is in the configuration file.

Configuring a default address is useful when the user is not required to know the IP address to gain access to a system (for example, users of a server that is available to many students on a campus). Instead of each user being required to know an IP address, they only need to enter the **slip default** or **ppp default** EXEC command and let the server select the address to use.

See the section "Making Additional Remote Node Connections" in the chapter "Configuring Asynchronous SLIP and PPP" in this publication for more information about the **slip** and **ppp** EXEC commands.

See the following sections for examples:

- Modem Asynchronous Group Example
- Configuring Specific IP Addresses for an Interface
- IP and PPP Asynchronous Interface Configuration Example

Allowing an Asynchronous Address to Be Assigned Dynamically

When a line is configured for dynamic assignment of asynchronous addresses, the user enters the **slip** or **ppp** EXEC command and is prompted for an address or logical host name. The address is validated by TACACS, when enabled, and the line is assigned the given address and put into asynchronous mode.

Assigning asynchronous addresses dynamically is useful when you want to assign set addresses to users. For example, an application on a personal computer that automatically dials in using Serial Line Internet Protocol (SLIP) and polls for electronic mail messages can be set up to dial in periodically and enter the required IP address and password.

To assign asynchronous addresses dynamically, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# async dynamic address`	Allows the IP address to be assigned when the protocol is initiated.

The dynamic addressing features of the internetwork allow packets to get to their destination and back regardless of the access server, router, or network they are sent from. For example, if a host such as a laptop computer moves from place to place, it can keep the same address no matter where it is dialing in from.

Logical host names are first converted to uppercase and then sent to the TACACS server for authentication.

See the following sections for examples of configurations that allow asynchronous addresses to be assigned dynamically:

- Access Restriction on the Asynchronous Interface Example
- Asynchronous Routing and Dynamic Addressing Configuration Example
- Network Address Conservation Using the ip unnumbered Command Example

Optimizing Available Bandwidth

Asynchronous lines have relatively low bandwidth and can easily be overloaded, resulting in slow traffic across these lines.

To optimize available bandwidth, perform either of the following optional tasks:

- Configuring Header Compression
- Forcing Header Compression at the EXEC Level

Configuring Header Compression

One way to optimize available bandwidth is by using TCP header compression. Van Jacobson TCP header compression (defined by RFC 1144) can increase bandwidth availability two- to five-fold when compared to lines not using header compression. Theoretically, it can improve bandwidth availability by a ratio of seven to one.

To configure header compression, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# ip tcp header-compression [on | off | passive]`	Configures Van Jacobson TCP header compression on the asynchronous link.

Forcing Header Compression at the EXEC Level

On SLIP interfaces, you can force header compression at the EXEC prompt on a line on which header compression has been set to passive. This option allows more efficient use of the available bandwidth and does not require entering privileged configuration mode.

To implement header compression, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# ip tcp header-compression passive`	Allows status of header compression to be assigned at the user level.

For PPP interfaces, the **passive** option functions the same as the **on** option.

See the following sections for examples of header compression:

- TCP Header Compression Configuration Example
- Network Address Conservation Using the ip unnumbered Command Example
- IGRP Configuration Example

Configuration Examples for Asynchronous Interfaces and Lines

This section provides the following asynchronous interface configuration examples:

- Interface and Line Configuration Examples
- Line AUX Configuration Example
- Rotary Group Examples
- Dedicated Asynchronous Interface Configuration Example
- Access Restriction on the Asynchronous Interface Example
- Group and Member Asynchronous Interface Examples
- Asynchronous Interface Address Pool Examples
- IP and SLIP Using an Asynchronous Interface Example
- IP and PPP Asynchronous Interface Configuration Example
- Asynchronous Routing and Dynamic Addressing Configuration Example
- TCP Header Compression Configuration Example
- Network Address Conservation Using the ip unnumbered Command Example
- Asynchronous Interface As the Only Network Interface Example
- Routing on a Dedicated Dial-In Router Example
- IGRP Configuration Example

Interface and Line Configuration Examples

This section contains the following examples:

Asynchronous Interface Backup DDR Configuration Example

The following is an example of one asynchronous interface configuration on a Cisco AS2511-RJ access server that is used in an asynchronous backup DDR scenario:

```
interface async 1
 description ASYNC LINE 5293731 TO HIGHWAY
 encapsulation ppp
 async default routing
 async mode dedicated
 dialer in-band
 dialer map ip 192.168.10.2 name Router2 broadcast
 dialer-group 1
 ppp authentication chap
```

Passive Header Compression and Default Address Example

The following configuration shows interface and line configuration. The interface is configured with access lists, passive header compression, and a default address. The line is configured for TACACS authentication.

```
interface async 1
 ip access-group 1 in
 ip access-group 1 out
 ip tcp header-compression passive
 async default ip address 172.31.176.201

line 1
 login tacacs
 location 457-5xxx
 exec-timeout 20 0
 password XXXXXXXX
 session-timeout 20
 stopbits 1
```

High-Density Dial-In Solution Using Autoselect and EXEC Control Example

The following example configures a Cisco AS5800 access server, which is used as a high-density dial-in solution:

```
line 1/2/00 1/9/71
 session-timeout 30
 exec-timeout 30 0
 absolute-timeout 240
 autoselect during-login
 autoselect ppp
```

```
 modem InOut
 transport preferred none
 transport input all
```

Asynchronous Line Backup DDR Configuration Example

The following example configures one asynchronous line on a Cisco AS2511-RJ access server that is used in an asynchronous backup DDR scenario:

```
line 1
 modem InOut
 speed 115200
 transport input all
 flowcontrol hardware
```

Line AUX Configuration Example

In the following example, the asynchronous interface corresponds to the AUX port. Use the **show line** command to determine which asynchronous interface corresponds to the AUX port. The IP address on the AUX ports of both routers are in the same subnet

```
interface Async1
 ip address 192.168.10.1 255.255.255.0
 encapsulation ppp
 async dynamic routing
 async mode dedicated
!
no ip classless
ip route 0.0.0.0 0.0.0.0 Async1 /Default route points to the Async1 (AUX port) interface.
!
!
logging buffered
!
line con 0
 exec-timeout 0 0
line aux 0
 modem InOut
 transport input all
 rxspeed 38400
 txspeed 38400
```

Rotary Group Examples

The following example establishes a rotary group consisting of virtual terminal lines 2 through 4 and defines a password on those lines. By using Telnet to connect to TCP port 3001, the user gets the next free line in the rotary group. The user need not remember the range of line numbers associated with the password.

```
line vty 2 4
 rotary 1
 password letmein
 login
```

The following example enables asynchronous rotary line queueing:

```
line 1 2
 rotary 1 queued
```

The following example enables asynchronous rotary line queueing using the round-robin algorithm:

```
line 1 2
 rotary 1 queued round-robin
```

Dedicated Asynchronous Interface Configuration Example

The following example shows how to assign an IP address to an asynchronous interface and place the line in dedicated network mode. Setting the stop bit to 1 is a performance enhancement.

```
line 20
 location Department PC Lab
 stopbits 1
 speed 19200
!
interface async 20
 async default ip address 172.18.7.51
 async mode dedicated
```

Access Restriction on the Asynchronous Interface Example

The following example shows how to allow most terminal users access to anything on the local network, but restrict access to certain servers designated as asynchronous servers:

```
! access list for normal connections
access-list 1 permit 192.168.0.0 0.0.255.255
!
access-list 2 permit 192.168.42.55
access-list 2 permit 192.168.111.1
access-list 2 permit 192.168.55.99
!
line 1
 speed 19200
 flow hardware
 modem inout
interface async 1
 async mode interactive
 async dynamic address
 ip access-group 1 out
 ip access-group 2 in
```

Group and Member Asynchronous Interface Examples

The following examples are included in this section:

- Asynchronous Group Interface Examples
- Modem Asynchronous Group Example
- High-Density Dial-In Solution Using an Asynchronous Group

Asynchronous Group Interface Examples

The following example shows how to create an asynchronous group interface 0 with group interface members 2 through 7, beginning in global configuration mode:

```
interface group-async 0
 group-range 2 7
```

The following example shows how you need to configure asynchronous interfaces 1, 2, and 3 separately if you do not have a group interface configured:

```
interface Async1
 ip unnumbered Ethernet0
 encapsulation ppp
 async default ip address 172.30.1.1
 async mode interactive
 async dynamic routing
!
interface Async2
 ip unnumbered Ethernet0
 encapsulation ppp
 async default ip address 172.30.1.2
 async mode interactive
 async dynamic routing
!
interface Async3
 ip unnumbered Ethernet0
!
 encapsulation ppp
 async default ip address 172.30.1.3
 async mode interactive
 async dynamic routing
```

The following example configures the same interfaces, but from a single group asynchronous interface:

```
interface Group-Async 0
 ip unnumbered Ethernet0
 encapsulation ppp
 async mode interactive
 async dynamic routing
 group-range 1 3
 member 1 async default ip address 172.30.1.1
 member 2 async default ip address 172.30.1.2
 member 3 async default ip address 172.30.1.3
```

Modem Asynchronous Group Example

To configure a group asynchronous interface, specify the group async number (an arbitrary number) and the group range (beginning and ending asynchronous interface number).

The following example shows the process of creating and configuring a group asynchronous interface for asynchronous interfaces 1 through 96 on a Cisco AS5300 access server, which is loaded with ninety-six 56K MICA technologies modems:

```
interface group-async 1
 ip unnumbered ethernet 0
 encapsulation ppp
 async mode interactive
 ppp authentication chap pap
 peer default ip address pool default
 group-range 1 96
```

High-Density Dial-In Solution Using an Asynchronous Group

The following example configures a Cisco AS5800 access server that is used as a high-density dial-in solution:

```
interface group-async 0
 ip unnumbered FastEthernet0/2/0
 encapsulation ppp
 async mode interactive
 peer default ip address pool default
 no cdp enable
 ppp authentication chap
 hold-queue 10 in
 group-range 1/2/00 1/9/71
```

Asynchronous Interface Address Pool Examples

The following sections provide examples of the use of Dynamic Host Configuration Protocol (DHCP) and local pooling mechanisms:

- DHCP Pooling Example
- Local Pooling Example
- Configuring Specific IP Addresses for an Interface

DHCP Pooling Example

The following global configuration example enables DHCP proxy-client status on all asynchronous interfaces on the access server:

```
ip address-pool dhcp-proxy-client
```

The following global configuration example shows how to specify which DHCP servers are used on your network. You can specify up to four servers using IP addresses or names. If you do not specify servers, the default is to use the IP limited broadcast address of 255.255.255.255 for transactions with any and all discovered DHCP servers.

```
ip dhcp-server jones smith wesson
```

The following interface configuration example illustrates how to disable DHCP proxy-client functionality on asynchronous interface 1:

```
async interface
interface 1
 no peer default ip address
```

Local Pooling Example

The following example shows how to select the IP pooling mechanism and how to create a pool of local IP addresses that are used when a client dials in on an asynchronous line. The default address pool comprises IP addresses 172.30.0.1 through 172.30.0.28.

```
! This command tells the access server to use a local pool.
```

```
ip address-pool local
! This command defines the ip address pool.
! The address pool is named group1 and comprised of addresses.
! 172.30.0.1 through 172.30.0.28 inclusive
ip local-pool group1 172.30.0.1 172.30.0.28
```

Configuring Specific IP Addresses for an Interface

The following example shows how to configure the access server so that it will use the default address pool on all interfaces except interface 7, on which it will use an address pool called lass:

```
ip address-pool local
ip local-pool lass 172.30.0.1
 async interface
 interface 7
 peer default ip address lass
```

IP and SLIP Using an Asynchronous Interface Example

The following example configures IP and SLIP on asynchronous interface 6. The IP address for the interface is assigned to Ethernet 0, interactive mode has been enabled, and the IP address of the client PC running SLIP has been specified.

IP and the appropriate IP routing protocols have already been enabled on the access server or router.

```
interface async 6
 ip unnumbered ethernet 0
 encapsulation slip
 async mode interactive
 async default ip address 172.18.1.128
```

IP and PPP Asynchronous Interface Configuration Example

The following example configures IP and PPP on asynchronous interface 6. The IP address for the interface is assigned to Ethernet 0, interactive mode has been enabled, and the IP address of the client PC running PPP has been specified. IP and the appropriate IP routing protocols have already been enabled on the access server or router.

```
interface async 6
 ip unnumbered ethernet 0
 encapsulation ppp
 async mode interactive
 peer default ip address 172.18.1.128
```

Asynchronous Routing and Dynamic Addressing Configuration Example

The following example shows a simple configuration that allows routing and dynamic addressing.

With this configuration, if the user specifies **/routing** in the EXEC **slip** or **ppp** command, routing protocols will be sent and received.

```
interface async 6
 async dynamic routing
 async dynamic address
```

TCP Header Compression Configuration Example

The following example configures asynchronous interface 7 with a default IP address, allowing header compression if it is specified in the **slip** or **ppp** connection command entered by the user or if the connecting system sends compressed packets.

```
interface async 7
 ip address 172.31.79.1
 async default ip address 172.31.79.2
 ip tcp header-compression passive
```

Network Address Conservation Using the ip unnumbered Command Example

The following example shows how to configure your router for routing using unnumbered interfaces. The source (local) address is shared between the Ethernet 0 and asynchronous 6 interfaces (172.18.1.1). The default remote address is 172.18.1.2.

```
interface ethernet 0
 ip address 172.18.1.1 255.255.255.0
!
interface async 6
 ip unnumbered ethernet 0
 async dynamic routing
! Default address is on the local subnet.
 async dynamic address
 async default ip address 172.18.1.2
 ip tcp header-compression passive
```

The following example shows how the IP unnumbered configuration works. Although the user is assigned an address, the system response shows the interface as unnumbered, and the address entered by the user will be used only in response to BOOTP requests.

```
Router> slip /compressed 10.11.11.254
Password:
Entering async mode.
Interface IP address is unnumbered, MTU is 1500 bytes.
Header compression is On.
```

Asynchronous Interface As the Only Network Interface Example

The following example shows how one of the asynchronous lines can be used as the only network interface. The router is used primarily as a terminal server, but is at a remote location and dials in to the central site for its only network connection.

```
ip default-gateway 10.11.12.2
interface ethernet 0
 shutdown
interface async 1
 async dynamic routing
 ip tcp header-compression on
 async default ip address 10.11.16.12
 async mode dedicated
 ip address 10.11.12.32 255.255.255.0
```

Routing on a Dedicated Dial-In Router Example

The following example shows how a router is set up as a dedicated dial-in router. Interfaces are configured as IP unnumbered to conserve network resources, primarily IP addresses.

```
ip routing
interface ethernet 0
 ip address 10.129.128.2 255.255.255.0
!
interface async 1
 ip unnumbered ethernet 0
 async dynamic routing
! The addresses assigned with SLIP or PPP EXEC commands are not used except
! to reply to BOOTP requests.
! Normally, the routers dialing in will have their own address and not use BOOTP at all.
 async default ip address 10.11.11.254
!
interface async 2
 ip unnumbered ethernet 0
 async default ip address 10.11.12.16
 ip tcp header-compression passive
 async mode dedicated
!
! Run RIP on the asynchronous lines because few implementations of SLIP
! understand IGRP. Run IGRP on the Ethernet (and in the local network).
!
router igrp 110
 network 10.11.12.0
! Send routes from the asynchronous lines on the production network.
 redistribute RIP
! Do not send IGRP updates on the asynchronous interfaces.
 passive-interface async 1
!
 router RIP
 network 10.11.12.0
 redistribute igrp
 passive-interface ethernet 0
! Consider filtering everything except a default route from the routing
! updates sent on the (slow) asynchronous lines.
 distribute-list 1 out
 ip unnumbered async 2
 async dynamic routing
```

IGRP Configuration Example

In the following example, only the Interior Gateway Routing Protocol (IGRP) TCP/IP routing protocol is running; it is assumed that the systems that are dialing in to use routing will either support IGRP or have some other method (for example, a static default route) of determining that the router is the best place to send most of its packets.

```
router igrp 111
 network 10.11.12.0
interface ethernet 0
 ip address 10.11.12.92 255.255.255.0
!
interface async 1
 async default ip address 10.11.12.96
 async dynamic routing
 ip tcp header-compression passive
 ip unnumbered ethernet 0

line 1
 modem ri-is-cd
```