Configuring the Airline Product Set

This chapter describes how to configure the Airline Product Set (ALPS). For a complete description of the ALPS commands in this chapter, refer to the "Airline Product Set Configuration Commands" chapter in the *Cisco IOS Bridging and IBM Networking Command Reference* (Volume 1 of 2). To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

This chapter contains the following sections:

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 Platform Support for Cisco IOS Software Features" section on page li in the "Using Cisco IOS Software" chapter.

ALPS Overview

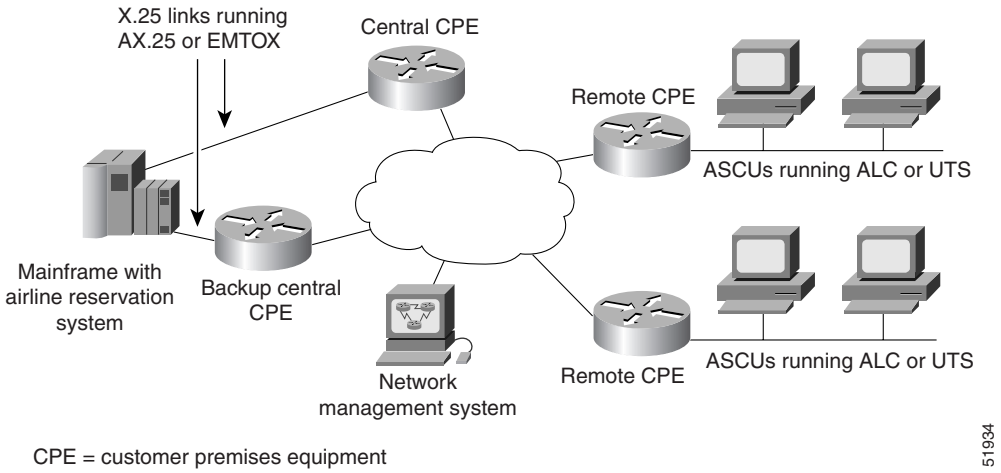
ALPS is a tunneling mechanism that transports airline protocol data across a Cisco router-based TCP/IP network to a mainframe. This feature provides connectivity between agent set control units (ASCUs) and a mainframe host that runs the airline reservation system.

The ALPS feature was released in three phases. The first two phases of ALPS enabled the network migration to TCP/IP without requiring any changes in the hardware or software of the endstations (ASCUs and mainframes). ALPS phase I and II utilized a new protocol, ALPS Tunneling Protocol (ATP), to tunnel airline protocol traffic (P1024B Airline Control [ALC] or P1024C Universal Terminal Support [UTS] data) through the TCP/IP network between peer Cisco routers. ALPS phase I provided support for the ALC protocol and the transport of the data from the ASCUs to a reservations system on an IBM mainframe. ALPS phase II provided support for the UTS protocol and the transport of the data from the ASCUs to a reservations system on a Unisys host system.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Figure 1 shows a basic ALPS topology with ALC, UTS, AX.25 and Exchange of Mixed Traffic over X.25 SVCs (EMTOX) protocols. Three major components provide the end-to-end transportation of airline protocol traffic across the network: the P1024B ALC or P1024C UTS protocol, the TCP-based transport protocol, and the AX.25/EMTOX access to the mainframe.

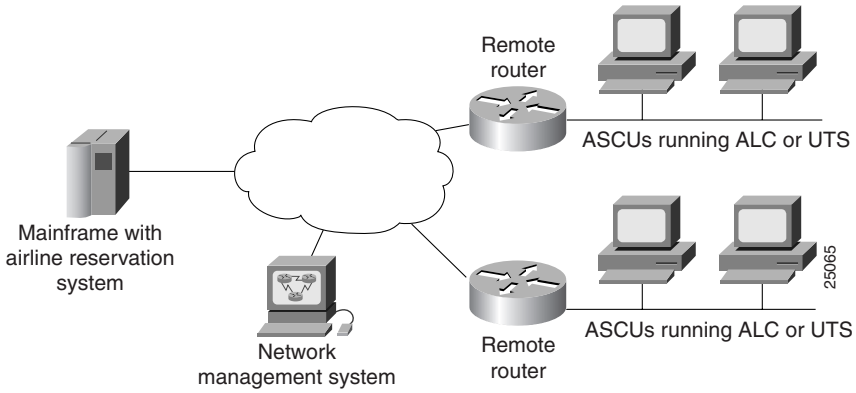
Figure 1 ALPS with ALC and UTS Architecture



ALPS phase III provides support for Mapping of Airline Traffic over Internet Protocol (MATIP). MATIP is an industry standard protocol for transporting airline protocol traffic across a TCP/IP network. This enhancement enables the end-to-end delivery of ALC and UTS data streams between a Cisco router and the mainframe using TCP/IP. ALPS with MATIP removes the X.25 (AX.25 or EMTOX) requirements for communication with the host reservation system by enabling TCP/IP communication between the router and the airline host reservation system.

Figure 2 shows the basic ALPS topology and the MATIP architecture implemented in Phase III. Three major components provide the end-to-end transportation of airline protocol traffic across the network: the P1024B ALC or P1024C UTS protocol, the TCP/IP-based MATIP protocol conversion, and the TCP/IP access to the mainframe.

Figure 2 ALPS with MATIP Architecture



In Cisco IOS Release 12.1(2)T and later, ALPS supports service messages additions and extensions to the ALPS P1024B ALC protocol support. The additions include customized options to configure the format, address, and sending of service messages. The ALPS ALC support is extended to be more scalable. The ALPS ASCU debug support is extended to include trace capability for the six-bit International Programmable Airline Reservation System (IPARS) format.

The Cisco ALPS feature provides the following benefits:

- Provides an end-to-end solution for airlines and central reservation systems.
- Allows airlines to replace their existing hardware and software with Cisco routers because the ALPS feature is integrated in the Cisco IOS software. For customers who already use Cisco routers, this feature allows them to consolidate networking overhead and functionality.
- Enables the end-to-end delivery of ALC and UTS data between a remote router or gateway and the mainframe using TCP/IP encapsulation.
- Eliminates network overhead for error detection and transmission logic associated with X.25 links.
- Replaces IBM front-end processors (FEPs) with Channel Interface Processors (CIPs).
- Eliminates the use of dedicated, leased, slow-speed ALC and UTS serial lines and migrates the reservation system networks to a modern networking paradigm.Once the mainframe reservation system is enabled to use TCP/IP, new applications can be written for PCs or network computers (NCs).
- Supports standards-based MATIP protocol for transporting data across the TCP/IP network.

In Cisco IOS Release 12.1(2)T and later, ALPS includes the following debug, ALC, and service message enhancements.

Debug Enhancement

The ALPS ASCU debug support additions provide new capabilities that enable you to display **debug alps ascu** command trace output in IPARS format.

ALC Enhancements

The ALPS ALC protocol stack includes the following extensions:

- Automatic ASCU reset
- T1 timer range increase
- Modification of the accepted ASCU IA value list

Service Message Enhancements

The additions to the ALPS service messages provide new capabilities that enable you to:

- Specify sita or apollo service message format
- Disable the forwarding of service messages for ALPS circuit status changes
- Specify where to retrieve the terminal address for dropped-data service messages
- Disable specific service messages
- Configure service message text with an increased character length

In Cisco IOS Release 12.1(3)T and later, ALPS includes the following ALC enhancement.

ALC Enhancement

The ALPS ALC protocol stack includes the following extensions:

- Nonpolled ALC ASCU support

The ALPS feature supports only type A conversational protocol traffic. The ALPS feature does not support MATIP type A host-to-host protocol traffic and MATIP type B messaging protocol traffic.

Remote routers must have the Cirrus Logic CD2430 chipset on a synchronous serial interface module to connect to the ALC or UTS ASCUs. The CD2430 chipset is supported on the following router platforms:

- Cisco 2520, 2521, 2522, and 2523
- Cisco 2600 series
- Cisco 3600 series
- Cisco 4500
- Cisco 4700

Note The Cisco 4500 and Cisco 4700 platforms must have a high-density, low-speed serial card installed. Sixteen low-speed ports are available for performing the remote router functions.

The ALPS feature supports the following standards, MIBs and RFCs:

Standards

- *P1024B Communication Control Protocol Specification*, Societe Internationale de Telecommunications Aeronautiques
- *P1024C Communication Control Protocol Specification*, Societe Internationale de Telecommunications Aeronautiques
- *MATIP Implementation Guide*, Societe Internationale de Telecommunications Aeronautiques

MIBs

The ALPS feature supports the CISCO-ALPS-MIB and the following MIB enhancements:

- Extensions to the alpsIfP1024Table
- Extension to the alpsAscuTable
- Addition of Simple Network Management Protocol (SNMP) notifications for ALPS circuit open request failure and ALPS circuit open request with a partial rejection

For descriptions of supported MIBs and how to use them, see the Cisco MIB website on Cisco.com.

RFCs

- RFC 2351, *Mapping of Airline Reservation, Ticketing, and Messaging Traffic over IP*, May 1998

ALPS Configuration Task List

See the following sections for configuration tasks for the ALPS feature. Each task in the list indicates if the task is optional or required. The tasks in the "Configuring the Remote Routers" section on page 5 are the only required tasks for ALPS with MATIP.

For a complete description of the ALPS commands in this feature module, refer to the "Airline Product Set Configuration Commands" chapter in the *Cisco IOS Bridging and IBM Networking Command Reference* (Volume 1 of 2). To locate documentation of other commands, use the command reference master index or search online.

See the "ALPS Configuration Examples" section on page 13 for more information.

Configuring the Remote Routers

Note To configure ALPS with MATIP, you must perform only the following tasks. The tasks also apply to EMTOX and AX.25, but are not required.

Perform the tasks in the following sections to configure the ALPS feature on the remote routers:

Specifying the ALPS Local Peer IP Address

You must identify an IP address as an ALPS local peer on the remote router. Only one ALPS local peer is permitted on a router.

To specify the ALPS local peer IP address, use the following commands in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# alps local-peer` *ipaddress* `[promiscuous]`	Specifies an IP address to use as the ALPS local peer on the remote router.
Step 2	`Router(config)# alps keepalive [interval` *time*`] [retry` *count*`]`	Enables TCP keepalives for ALPS TCP peer connections.

Specifying the ALPS Remote Peer IP Address

You must specify a partner IP address (remote peer) on the remote router. The peer connection may be permanent or dynamic (established on demand). You can configure an ATP connection to be permanent or dynamic by configuring the optional **dynamic** keyword.

Note MATIP sessions are dynamic, whether or not the **dynamic** keyword is configured. To simulate a permanent connection in MATIP, configure the **dynamic** keyword with an *inact-timer* value of zero.

To specify the partner IP address for one or more TCP peer connections to the configured IP address, use the following command in global configuration mode:

Command	Purpose
`Router(config)# alps remote-peer` *ip-addr* [**protocol** {**atp** | **matip-a**}] [**status-interval** *interval*] [**status-retry** *retries*] [**dynamic** [*inact-timer*] [**no-circuit** *no-circ-timer*]] [**tcp-qlen** [*num*]]	Specifies the partner IP address. If you select the ATP protocol, you must configure the data center routers.

Specifying the ALPS Circuit

An ALPS circuit is a communication path across a TCP connection for one or more ASCUs. The ALPS circuit must have a configured association with an ALPS remote peer to establish a connection to the host. Additionally, an ALPS circuit configuration may specify a different remote peer as a backup peer to the host. Each MATIP circuit maps to a single TCP connection. For ATP, ALPS circuits can be multiplexed across to a single TCP connection.

To specify an ALPS circuit, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# alps circuit` *name*	Specifies an ALPS circuit at the remote router and enters ALPS circuit submode.
Step 2	`Router(config-alps-circ)# alps primary-peer` *ip-addr* [**backup-peer** *ip-addr*]	Specifies the primary TCP peer and an optional backup peer for this ALPS circuit.
Step 3	`Router(config-alps-circ)# alps local-hld` *loc-hld* **remote-hld** *rem-hld*	Specifies the local high-level designator (HLD) for this ALPS circuit. The **remote-hld** keyword is not applicable for ALPS with MATIP. The *loc-hld* is the hld of the device that is being replaced. The *rem-hld* is the hld of the host mainframe.
Step 4	`Router(config-alps-circ)# alps hostlink` *number* {**ax25** *lcn* | **emtox** *x121-addr*} [**winout** *val1*] [**winin** *val2*] [**ops** *val3*] [**ips** *val4*]	Specifies information required to establish an X.25 virtual circuit at the central CPE.
Step 5	`Router(config-alps-circ)# alps connection-type permanent` [*retry-timer*]	(Optional) Specifies that this circuit should be established when the circuit is enabled.
Step 6	`Router(config-alps-circ)# alps lifetime-timer` *timer*	(Optional) Specifies how long messages can be queued in the ALPS circuit queue.
Step 7	`Router(config-alps-circ)# alps service-msg-interval` *seconds*	(Optional) Specifies the interval between the transmission of a service message to an ASCU and the transmission of a PLEASE RETRY message. The PLEASE RETRY message is transmitted only to ASCUs that use circuits with a dynamic connection type.
Step 8	`Router(config-alps-circ)# alps service-msg-list` *list*	(Optional) Defines the service message list to be used for this circuit.
Step 9	`Router(config-alps-circ)# alps matip-close-delay` *time*	(Optional) Specifies the interval between the closing and reopening of the MATIP circuit connection.

	Command	Purpose
Step 10	`Router(config-alps-circ)# alps idle-timer timer`	(Optional) Specifies (for dynamic circuits) the length of time that can elapse before an idle circuit is disabled.
Step 11	`Router(config-alps-circ)# alps mpx {group | single} hdr {a1a2 | none}`	(Optional) Specifies the multiplexing and the ASCU identification header for this circuit.
Step 12	`Router(config-alps-circ)# alps enable-circuit`	Enables the circuit.

Specifying Each ASCU

You must configure each ASCU within the context of the serial interface configuration. You must configure ASCU addressing information and association with an ASCU. You can configure the timers, maximum frame sizes, retry values, and polling mode optional configuration parameters for each ASCU. Appropriate default parameters are used for unspecified parameters. Once you configure the first ASCU, you can configure additional ASCUs using only Steps 8 through 14.

To specify an ASCU, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Configures an interface and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation [alc | uts]`	Specifies the protocol to be used on the serial interface.
Step 3	`Router(config-if)# alps t1 delay`	(Optional) Specifies the timeout delay between the transmission of an ALC poll message and the receipt of the first character of the poll message response.
Step 4	`Router(config-if)# alps t2 delay`	(Optional—ALC only) Specifies the timeout delay between receipt of the first character of the response to a poll message and the receipt of a Go Ahead message. Applies to ALC, only.
Step 5	`Router(config-if)# alps n1 errors`	(Optional) Specifies the threshold of consecutive errors logged before an ASCU is declared down.
Step 6	`Router(config-if)# alps n2 polls`	(Optional) Specifies the number of polls that must be correctly replied to before an ASCU is declared up.
Step 7	`Router(config-if)# alps n3 value`	(Optional—UTS only) Specifies the maximum number of retransmissions of an unacknowledged output data message to an ASCU. Applies only to UTS.
Step 8	`Router(config-if)# alps servlim polls`	(Optional) Specifies the number of cycles of the active poll list to execute before polling the next ASCU on the inactive poll list.
Step 9	`Router(config-if)# transmitter-delay delay`	Specifies number of padding characters added to the end of the frame (minimum dead-time after transmitting a packet).

	Command	Purpose
Step 10	`Router(config-if)# half-duplex`	Specifies half-duplex mode on a serial interface. This command specifies whether hardware flow control (constant or switched Request to Send [RTS]) is to be used between a DTE and DCE device. • If half-duplex is specified for a DTE, the DTE raises RTS and waits for the DCE to raise Clear to Send (CTS) before sending. • If half-duplex is specified for a DCE, the DCE waits for the DTE to raise RTS, then the DCE raises CTS to allow the DTE to send. • If full-duplex is specified, RTS is assumed and CTS is not monitored. **Note** ALPS supports the serial interface commands that are available if half-duplex mode is specified. This support applies to an interface that is configured as data circuit-terminating equipment (DCE) and data terminal equipment (DTE).
Step 11	`Router(config-if)# alps poll-pause` *msec*	(Optional) Specifies the minimum interval, in milliseconds, between initiations of the polling cycle.
Step 12	`Router(config-if)# alps service-msg data-drop` {**msg-term** | **config-term**}	(Optional) Specifies where to retrieve the terminal address to use when a service message is sent to an ASCU as a result of a dropped data message from a terminal.
Step 13	`Router(config-if)# alps service-msg format` {**sita** | **apollo**}	(Optional) Specifies the protocol format of service messages sent from the router to an ASCU.
Step 14	`Router(config-if)# alps service-msg status-change`	(Optional) Specifies that service messages for ALPS circuit status changes will be sent to ASCUs on the serial interface.
Step 15	`Router(config-if)# alps ascu` *id*	Specifies a physical ASCU identity (the ASCU interchange address value for ALC) and enters ALPS ASCU submode.
Step 16	`Router(config-alps-ascu)# alps default-circuit` *name*	Specifies the ALPS circuit that this ASCU uses.
Step 17	`Router(config-alps-ascu)# alps a1-map` *a1-value* **a2-map** *a2-value*	Specifies the A1 and A2 logical ASCU identification information.
Step 18	`Router(config-alps-ascu)# alps retry-option` [**resend** | **reenter**]	(Optional) Specifies the retry option when an ALC message with a bad cyclic check character (CCC) is received.
Step 19	`Router(config-alps-ascu)# alps max-msg-length` *value*	(Optional) Specifies maximum input message length.

	Command	Purpose
Step 20	`Router(config-alps-ascu)# alps error-display number1 number2`	(Optional) Specifies where error messages are displayed: • For P1024B ALC, the *number1* argument specifies the terminal address (TA) where these service messages are sent and the *number2* argument specifies the screen line number where service messages are displayed. • For P1024C UTS, the *number1* argument specifies the screen line number where service messages are displayed and *number2* argument specifies the column number where service messages are displayed.
Step 21	`Router(config-alps-ascu)# alps auto-reset`	(Optional) Automatically resets non-responsive ALC ASCUs in the DOWN state.
Step 22	`Router(config-alps-ascu)# alps alias alias-interchange-address`	(Optional) Specifies that an ALC ASCU is to operate in nonpolling mode and specifies the parent ASCU interchange address to which this ASCU is aliased.
Step 23	`Router(config-alps-ascu)# alps enable-ascu`	Polls the ASCU.

Configuring the Data Center Router

Note These tasks apply to EMTOX and AX.25, only.

Perform the tasks in the following sections to configure the ALPS feature on the data center router:

- Specifying the ALPS Host Local Peer Address, page 9
- Specifying AX.25, page 10
- Specifying EMTOX, page 10

Specifying the ALPS Host Local Peer Address

You must identify an IP address to use as the ALPS local peer IP address. Only one ALPS host local peer is permitted on a router. The promiscuous option, which allows any remote router to connect, is recommended at the central CPE.

To specify the ALPS host local peer address, use the following command in global configuration mode:

Command	Purpose
`Router(config)# alps local-peer ip-address [promiscuous]`	Specifies the IP address of the local peer.

Specifying AX.25

To enable AX.25 on an X.25 interface, the ALPS host HLD and hostlink number must be configured and AX.25 must be specified on an X.25 serial interface. At circuit-establishment time, the remote router forwards the host HLD, the logical channel number (LCN), and the hostlink number for the permanent virtual circuit (PVC), to be used for the ASCU group.

To configure AX.25 on an X.25 interface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Configures an interface and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation x25`	Specifies a serial interface as an X.25 device.
Step 3	`Router(config-if)# alps host-hld hld host-link num {{ax25 [damp-tmr value]} | {emtox x.121 [pseudo-conv]}} [life-tmr value]`	Enables ALPS on the X.25 interface.

Specifying EMTOX

To enable EMTOX on an X.25 interface, the host HLD and the hostlink number must be configured and EMTOX must be specified on an X.25 serial interface. At circuit-establishment time, the remote router forwards the X.121 address to be used as the calling address in the X.25 call and the host HLD and the hostlink number. If the host performs a call out, a correlation between the X.121 called address and a remote router peer IP address must be configured.

To configure EMTOX on an X.25 interface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Configures an interface and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation x25`	Specifies a serial interface as an X.25 device.
Step 3	`Router(config-if)# alps host-hld hld host-link num {{ax25 [damp-tmr value]} | {emtox x.121 [pseudo-conv]}} [life-tmr value]`	Enables ALPS on the X.25 interface.
Step 4	`Router(config-if)# alps translate x.121-addr ip-addr`	Maps an X.121 address to an IP address on a remote peer.

Customizing the Service Messages

You can customize the contents of the service messages and service message list. To specify the service message number and the content of the message, use the following command in global configuration mode:

Command	Purpose
`Router(config)# alps service-msg-list list number number msg`	Specifies service message numbers and content.

Note The default service message is used if no service message list number is specified. If you configure a particular service message on a list, the default service message still is used for the rest of the messages on that list.

Note Once the **alps service-msg-list number** command has been configured, you can define the service message list to be used on the circuit by configuring the **alps service-msg-list** command.

Note You can configure the handling of service messages using the **alps service-msg data-drop**, **alps service-msg format**, and **alps service-msg status-change** interface configuration-level commands.

Table 8 shows the default service message text strings:

Table 8 Service Message Default Text Strings

Message Number	Event	Text String
1	ALPS circuit to host is opened.	CONNECTION UP
2	X.25 virtual circuit at the host has been cleared.	DISC BY THE HOST
3	X.25 interface at the host is down.	HOST ISOLATED
4	No response from the host router when trying to establish a connection.	NETWORK PROBLEM
5	Connection to host was disconnected because of inactivity.	READY TO CONNECT
6	Network is congested.	CONGESTION
7	Network congestion has cleared.	PLEASE PROCEED
8	Network operator has disabled the path to the host.	DISC BY NET OPERAT

Customizing the Alarm Notifications

You can enable and customize alarms (error messages) and SNMP traps. To enable and customize alarms for the ALPS ASCUs, circuits, or peers, use the following commands in global configuration mode:

Command	Purpose
`Router(config)# alps enable-alarms ascu [interface id]`	Enables alarms for the ALPS ASCUs.
`Router(config)# alps enable-alarms circuit [name]`	Enables alarms for the ALPS circuits.
`Router(config)# alps enable-alarms peer [ip-address]`	Enables alarms for the ALPS peers.

Updating a Circuit

You can clear or update the circuits on the ALPS network. If a specific name is entered, the update action will be executed only on a configured circuit with that name; otherwise, the action will be performed on all configured circuits. If the circuit uses the ATP protocol, an update consists of a closing and reopening

of the ALPS circuit (the same action performed when clearing the circuit). If the circuit is a MATIP circuit, the update results in the sending of a configuration update (in the form of a MATIP Session Open command). You can update the circuit only on enabled or active (opened or opening state) ALPS circuits.

To update one or more ALPS circuits, use the following command in EXEC mode:

Command	Purpose
Router# **alps update-circuit** [*name*]	Specifies name of circuit to update.

Verifying ALPS

Perform the tasks in the following steps to verify the components of the ALPS network:

Step 1 Verify that the connection between the router and the ASCU is up by polling the ASCU. Enter the **show alps ascu** command and check the state field. UP indicates that the ASCU is responding to the polling. DOWN indicates that the connection is not responding to the polling.

```
router# show alps ascu

interface    dlc   id   a1   a2   circuit    pkt_tx  pkt_rx  state
-------------------------------------------------------------------------------
Serial6      ALC   42   60   70   CKT_ALC_1  416     416     UP
Serial6      ALC   45   60   72   CKT_ALC_1  600     600     UP
Serial6      ALC   48   62   78   CKT_ALC_2  0       0       DOWN
Serial7      UTS   21   22   13   CKT_UTS    4830    4830    UP
```

Step 2 Verify that the peer between the router and the host is connected. Enter the **show alps peer** command and check the state field. OPENED indicates that the circuit is connected. DISCONN indicates that the circuit is disconnected.

```
router# show alps peers

local_peer : ip_address = 192.168.25.2

ip_address     conn_id             state    pkt_t  pkt_rx
-------------------------------------------------------------------------------
192.168.20.3   MATIP_A_CKT_UTS     OPENED   1023   1023
192.168.70.2   MATIP_A_CKT_ALC_1   OPENED   4852   4757
192.168.70.2   MATIP_A_CKT_ALC_2   OPENED   1      1
192.168.70.3   MATIP_A_CKT_ALC_1   DISCONN  0      0
192.168.70.3   MATIP_A_CKT_ALC_2   DISCONN  0      0
```

Step 3 Verify that the ALPS circuit to the peer host is open and connected. Enter the **show alps circuits** command and check the state field. OPEN indicates that the circuit is connected. INOP indicates that the circuit is disconnected.

```
router# show alps circuits

name         pri_peer       curr_peer      dlc   state  pkt_tx  pkt_rx
-------------------------------------------------------------------------------
ALC_EMTOX    192.168.45.2   192.168.45.2   ALC   OPEN   944     944
UTS_AX25     192.168.45.2   192.168.45.2   UTS   OPEN   425     425
```

Monitoring and Maintaining ALPS

To monitor the status of the ALPS feature, use the following commands in EXEC mode:

Command	Purpose
Router# **show alps ascu** [*interface*] [*id*] [**detail**]	Displays the status of the ALPS ASCU.
Router# **show alps circuits** [**peer** *ip address*] [**name** *name*] [**detail**]	Displays the status of the ALPS circuits.
Router# **show alps peers** [**ipaddress** *addr*] [**detail**]	Displays the status of the ALPS remote peers.

ALPS Configuration Examples

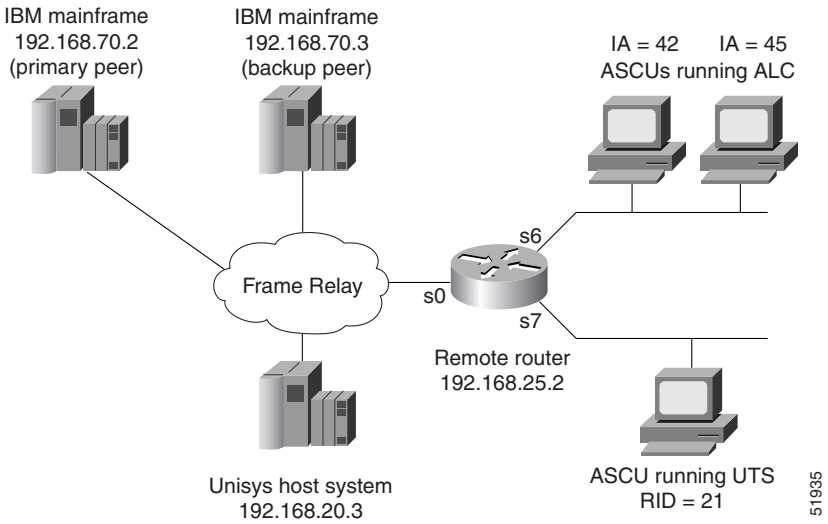
This section provides the following configuration examples:

- ALPS with MATIP Configuration for ALC and UTS Example, page 14
- ALPS Configuration for ALC and AX.25 Example, page 16
- ALPS Configuration for UTS and EMTOX Example, page 18

ALPS with MATIP Configuration for ALC and UTS Example

Figure 3 shows a simple example of a router topology for the ALPS with MATIP feature. The configuration corresponding to this topology follows.

Figure 3 Router Topology for the ALPS with MATIP Configuration Example



IA = interchange address
RID = remote identifier

ALC/UTS Router Configuration

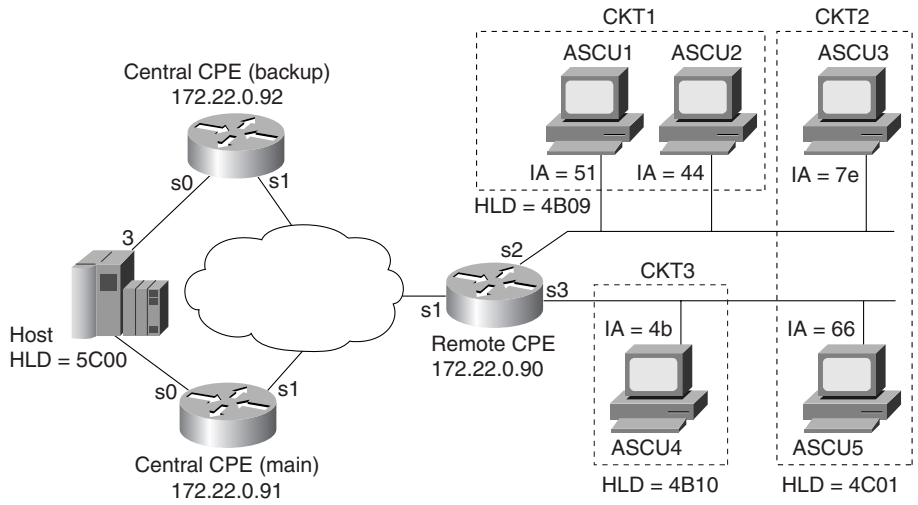
```
(config)# hostname alps-rcpe
(config)# alps local-peer 192.168.25.2
(config)# alps keepalive interval 45 retry 2
(config)# alps remote-peer 192.168.20.3 protocol matip-a dynamic status-interval 60
(config)# alps remote-peer 192.168.70.2 protocol matip-a dynamic 0 no-circuit 10
(config)# alps remote-peer 192.168.70.3 protocol matip-a dynamic 45
(config)# alps enable-alarms peer 192.168.70.2
(config)# alps enable-alarms ascu
!
(config)# alps circuit CKT_ALC_1
(config-alps-circ)# alps primary-peer 192.168.70.2 backup-peer 192.168.70.3
(config-alps-circ)# alps connection-type permanent
(config-alps-circ)# alps local-hld 2525
(config-alps-circ)# alps enable-circuit
!
(config)# alps circuit CKT_UTS
(config-alps-circ)# alps primary-peer 192.168.20.3
(config-alps-circ)# alps mpx single
(config-alps-circ)# alps idle-timer 90
(config-alps-circ)# alps local-hld 2527
(config-alps-circ)# alps enable-circuit
(config-alps-circ)# alps service-msg-interval 2
!
(config)# interface Loopback0
(config-if)# ip address 192.168.25.2 255.255.255.0


(config)# interface Serial0
(config-if)# ip address 210.100.50.2 255.255.255.0
```

```
(config-if)# encapsulation frame-relay IETF
(config-if)# frame-relay map ip 210.100.60.2 40
(config-if)# frame-relay map ip 210.100.70.2 50
!
(config)# interface Serial6
(config-if)# encapsulation alc
(config-if)# alps t1 6
(config-if)# alps t2 8
(config-if)# alps poll-pause 100
(config-if)# clockrate 9600
!
(config-if)# alps ascu 42
(config-alps-ascu)# alps default-circuit CKT_ALC_1
(config-alps-ascu)# alps a1-map 60 a2-map 70
(config-alps-ascu)# alps enable-ascu
!
(config-if)# alps ascu 45
(config-alps-ascu)# alps default-circuit CKT_ALC_1
(config-alps-ascu)# alps a1-map 60 a2-map 72
(config-alps-ascu)# alps enable-ascu
!
(config)# interface Serial7
(config-if)# encapsulation uts
(config-if)# alps n3 4
(config-if)# alps poll-pause 125
(config-if)# clockrate 4800
!
(config-if)# alps ascu 21
(config-alps-ascu)# alps default-circuit CKT_UTS
(config-alps-ascu)# alps a1-map 22 a2-map 13
(config-alps-ascu)# alps enable-ascu
!
```

ALPS Configuration for ALC and AX.25 Example

Figure 4 shows a simple router topology for the ALPS feature with ALC encapsulation. The configuration for this topology follows.

Figure 4 Router Topology for the ALPS Configuration for ALC Encapsulation Example



Remote CPE Configuration

```
(config)# alps local-peer 172.22.0.90
(config)# alps keepalive interval 60
(config)# alps remote-peer 172.22.0.91
(config)# alps remote-peer 172.22.0.92 dynamic 60
(config)# alps service-msg-list 1 number 2 TERMINAL OFF
!
(config)# alps circuit CKT1
(config-alps-circ)# alps primary-peer 172.22.0.91 backup-peer 172.22.0.92
(config-alps-circ)# alps local-hld 4B09 remote-hld 5C00
(config-alps-circ)# alps connection-type permanent 30
(config-alps-circ)# alps lifetime-timer 3
(config-alps-circ)# alps hostlink 3 ax25 120 winout 3 winin 3
(config-alps-circ)# alps service-msg-interval 3
(config-alps-circ)# alps service-msg-list 1
(config-alps-circ)# alps enable-circuit
!
(config)# alps circuit CKT2
(config-alps-circ)# alps primary-peer 172.22.0.91 backup-peer 172.22.0.92
(config-alps-circ)# alps local-hld 4C01 remote-hld 5C00
(config-alps-circ)# alps hostlink 3 ax25 1500 winout 4 winin 5
(config-alps-circ)# alps enable-circuit
!


(config)# alps circuit CKT3
(config-alps-circ)# alps primary-peer 172.22.0.91
(config-alps-circ)# alps local-hld 4B10 remote-hld 5C00
(config-alps-circ)# alps connection-type permanent 30
(config-alps-circ)# alps lifetime-timer 6
```

```
(config-alps-circ)# alps hostlink 3 ax25 905
(config-alps-circ)# alps enable-circuit
!
(config)# interface serial 1
(config-if)# ip address 172.22.0.90 255.255.255.0
!
(config)# interface serial 2
(config-if)# encapsulation alc
(config-if)# alps t1 3
(config-if)# alps t2 6
(config-if)# alps n1 3
(config-if)# alps n2 2
(config-if)# alps servlim 20
!
(config-if)# alps ascu 51
(config-alps-ascu)# alps default-circuit CKT1
(config-alps-ascu)# alps a1-map 40 a2-map 2D
(config-alps-ascu)# alps retry-option resend
(config-alps-ascu)# alps max-msg-length 1950
(config-alps-ascu)# alps error-display 6d 78
(config-alps-ascu)# alps enable-ascu
!
(config-if)# alps ascu 44
(config-alps-ascu)# alps default-circuit CKT1
(config-alps-ascu)# alps a1-map 40 a2-map 2E
(config-alps-ascu)# alps max-msg-length 590
(config-alps-ascu)# alps error-display 6d 78
(config-alps-ascu)# alps enable-ascu
!
(config-if)# alps ascu 7E
(config-alps-ascu)# alps default-circuit CKT2
(config-alps-ascu)# alps a1-map 40 a2-map 2F
(config-alps-ascu)# alps retry-option re-send
(config-alps-ascu)# alps max-msg-length 2000
(config-alps-ascu)# alps error-display 6d 78
(config-alps-ascu)# alps enable-ascu

(config)# interface serial 3
(config-if)# encapsulation alc
(config-if)# alps t1 5
(config-if)# alps t2 6
(config-if)# alps n1 1
(config-if)# alps n2 2
(config-if)# alps servlim 20
!
(config-if)# alps ascu 4B
(config-alps-ascu)# alps default-circuit CKT3
(config-alps-ascu)# alps a1-map 63 a2-map 41
(config-alps-ascu)# alps retry-option re-send
(config-alps-ascu)# alps max-msg-length 1960
(config-alps-ascu)# alps error-display 6d 78
(config-alps-ascu)# alps enable-ascu

(config-if)# alps ascu 66
(config-alps-ascu)# alps default-circuit CKT2
(config-alps-ascu)# alps a1-map 71 a2-map 21
(config-alps-ascu)# alps max-msg-length 3800
(config-alps-ascu)# alps error-display 6d 78
(config-alps-ascu)# alps enable-ascu
```

Central CPE Configuration (Main)

AX.25 Host

```
(config)# alps local-peer 172.22.0.91 promiscuous
(config)# interface serial 0
(config-if)# encapsulation x25 ax25
(config-if)# x25 ltc 1024
(config-if)# alps host-hld 5C00 host-link 3 ax25
```

Central CPE Configuration (Backup)

AX.25 Host

```
(config)# alps local-peer 172.22.0.92 promiscuous
(config)# interface serial 0
(config-if)# encapsulation x25 ax25
(config-if)# x25 ltc 1024
(config-if)# alps host-hld 5C00 host-link 3 ax25
```

ALPS Configuration for UTS and EMTOX Example

The following configuration is an example of routing P1024C UTS data frames across the network between central and remote equipment.

Remote Router Configuration

```
(config)# hostname alps-rcpe
(config)# alps local-peer 200.100.25.2
(config)# alps keepalive interval 45 retry 5
(config)# alps remote-peer 200.100.40.2
(config)# alps enable-alarms peer 200.100.40.2
(config)# alps enable-alarms ascu

(config)# alps circuit UTS_EMTOX
(config-alps-circ)# alps primary-peer 200.100.40.2
(config-alps-circ)# alps idle-timer 90
(config-alps-circ)# alps local-hld 2525 remote-hld 5050
(config-alps-circ)# alps mpx single
(config-alps-circ)# alps hostlink 6 emtox 1100 ops 512 ips 512
(config-alps-circ)# alps service-msg-interval 2
(config-alps-circ)# alps enable-circuit

(config)# interface Loopback0
(config-if)# ip address 200.100.25.2 255.255.255.0

(config)# interface Serial0
(config-if)# ip address 200.100.50.2 255.255.255.0
(config-if)# encapsulation frame-relay IETF
(config-if)# frame-relay map ip 200.100.50.3 20


(config)# interface Serial1
(config-if)# encapsulation uts
(config-if)# alps n1 5
(config-if)# alps n3 4
(config-if)# alps poll-pause 200
(config-if)# clockrate 4800
!
(config-if)# alps ascu 21
(config-alps-ascu)# alps default-circuit UTS_EMTOX
(config-alps-ascu)# alps a1-map 22 a2-map 13
```

```
(config-alps-ascu)# alps enable-ascu
!
```

Central CPE Configuration

```
(config)# hostname alps-ccpe
(config)# alps local-peer 200.100.40.2 promiscuous
(config)# alps enable-alarms circuit
!
(config)# interface Loopback0
(config-if)# ip address 200.100.40.2 255.255.255.0
!
(config)# interface Serial0
(config-if)# ip address 200.100.50.3 255.255.255.0
(config-if)# encapsulation frame-relay IETF
(config-if)# clockrate 56000
(config-if)# frame-relay map ip 200.100.50.2 20
!
(config)# interface Serial2
(config-if)# encapsulation x25 dce
(config-if)# alps host-hld 5050 host-link 6 emtox 2222
(config-if)# alps translate 110* 200.100.25.2
(config-if)# clockrate 64000
```