Configuring and Using
Solstice™ PPP 3.0.1 Servers and Routers

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 802-5354-10
Revision A, March 1996

SunSoft
A Sun Microsystems, Inc. Business

Contents

Figures

Preface

The *Configuring and Using Solstice™ PPP Servers and Routers* describes how to configure and use Solstice PPP to create synchronous and asynchronous point-to-point communications links between computers and local area networks (LANs). Solstice PPP is a standard implementation of the Point-to-Point Protocol (PPP) and the Internet Protocol Control Protocol (IPCP).

Who Should Use This Book

This book is intended for experienced system administrators who want to implement and maintain networks using Solstice PPP. It assumes that you are familiar with workstations and servers running a Solaris™ environment.

How This Book Is Organized

Chapter 1, "Introducing Solstice PPP," describes the Solstice PPP implementation of the Point-to-Point Protocol (PPP), and tells you how to use Solstice PPP to establish IP connections across PPP links.

Chapter 2, "Configuring Solstice PPP using pppinit," tells you how to create a basic configuration using the script `pppinit(1M)`, and how to add user accounts for incoming connections using `admintool(1M)`.

Chapter 3, "Starting and Using Solstice PPP," explains how to start Solstice PPP on your machine, how to establish IP connections across a PPP link, and how to close connections. It assumes that you have installed and configured Solstice PPP on your machine.

Chapter 4, "Editing the Configuration Files," describes the format and syntax of each of the configuration files associated with Solstice PPP, and explains how to modify these files to generate more complex network configurations.

Chapter 5, "Example Configurations," includes some examples of common network configurations created using Solstice PPP, and the corresponding configuration files `ppp.conf` and `link.conf` for each one.

Chapter 6, "Troubleshooting and Diagnostics," tells you how to detect and resolve problems when running Solstice PPP. It includes instructions on how to use `ppptrace` and `pppstat` to examine the frame traffic across the PPP link, and a description of the error and status messages logged in the file `/var/adm/log/ppp.log`.

Appendix A, "PPP Link Operation," provides a brief overview of PPP link operation, and includes a phase diagram and a description of the various PPP frames.

Appendix B, "Modem and Null Modem Cables," provides the pinouts of cables that can be used for most modem and null modem configurations. Refer to the manufacturer's documentation for a detailed description of special cables you may need to connect your particular modem

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

Introducing Solstice PPP 1

This chapter describes the Solstice PPP implementation of the Point-to-Point Protocol (PPP), and describes how it is used to run IP applications across PPP links.

Overview

Solstice PPP is a standard implementation of the Point-to-Point Protocol (PPP), which defines a method for transmitting multiprotocol datagrams over synchronous and asynchronous serial point-to-point links, and the Internet Protocol Control Protocol (IPCP), which defines a method for transmitting IP datagrams over PPP.

Synchronous PPP provides permanent connections between two endpoints, and is typically used for LAN-to-LAN interconnectivity across dedicated leased-lines. Asynchronous PPP provides temporary connections between two endpoints, and is typically used for mobile communication, teleworking, and Internet access across public and private telephone networks.

Feature Summary

- Interoperates with all standard implementations of PPP.
- Integration of synchronous and asynchronous PPP in a single, homogeneous environment.
- Runs on both Solaris™ SPARC™ and Solaris x86 platforms.
- Simplified port and modem configuration.
- Dynamic IP address allocation to simplify server access.
- Enhanced security based on the PPP Password Authentication Protocol (PAP), the PPP Challenge-Handshake Authentication Protocol (CHAP).
- Load-sharing increases available bandwidth for synchronous connections.
- Improved trace and log facilities to simplify troubleshooting.
- Interactive CHAT scripts to accept user input during the connection phase.

Product Conformance

Solstice PPP is a standard implementation of the following specifications:

- **RFC 1661 (updates RFC 1331) — Point-to-Point Protocol (PPP)**
  Describes a standard method for transporting multiprotocol datagrams over serial point-to-point links.
- **RFC 1662 — PPP in HDLC-like Framing**
  Describes the use of HDLC-like framing for PPP encapsulated packets.
- **RFC 1332 — PPP Internet Protocol Control Protocol (IPCP)**
  Describes the Network Control Protocol (NCP) for establishing and configuring the Internet Protocol (IP) over PPP, and a method for negotiating the use of van jacobson TCP/IP header compression with PPP.
- **RFC 1334 — PPP Authentication Protocols**
  Describes two protocols for user authentication in the PPP domain: the Password Authentication Protocol (PAP) and the Challenge-Handshake Authentication Protocol (CHAP).
- **RFC 1144 — Compressing TCP/IP Headers for Low-Speed Serial Links**
  Describes a method to improve the performance of TCP/IP connections across low-speed links by compressing the packet headers.

Running IP Applications over Solstice PPP

Once configured and started on a host machine, Solstice PPP encapsulates standard IP packets and routes them across serial point-to-point links. This process occurs in three phases, as shown in Figure 1-1.

1. A physical connection is established between two endpoints.
2. A PPP link is established over the physical connection.
3. IP is established over the PPP link.

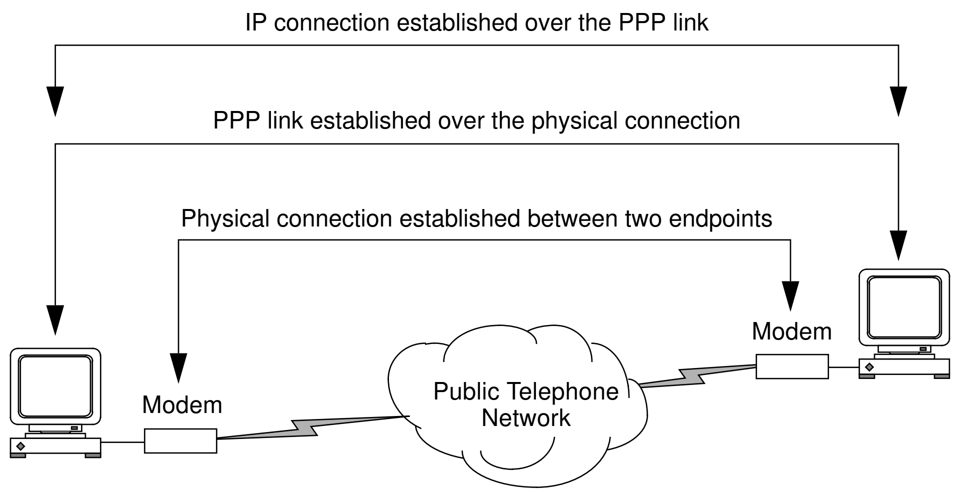


Figure 1-1 Establishing IP over PPP Links

Establishing the Physical Connection

Solstice PPP transmits data over full-duplex, bit-serial connections. It supports any of the common serial communications protocols, including EIA-232-E (formerly, RS-232-C), EIA-422, EIA-423, EIA-530, and CCITT V.24 and V.35. The physical connection for Solstice PPP may be either synchronous or asynchronous.

Synchronous Connections

Synchronous connections use independent clocking signals to synchronize the data transmission. They provide a permanent connection between two endpoints, and are typically established over dedicated leased-lines.

You pay for the synchronous connection for the duration of the lease and the cost is independent of the quantity of data transmitted. For these reasons, synchronous connections are best suited to continuous data traffic and are used typically for LAN-to-LAN interconnectivity and bulk transfers of information as shown in Figure 1-2.

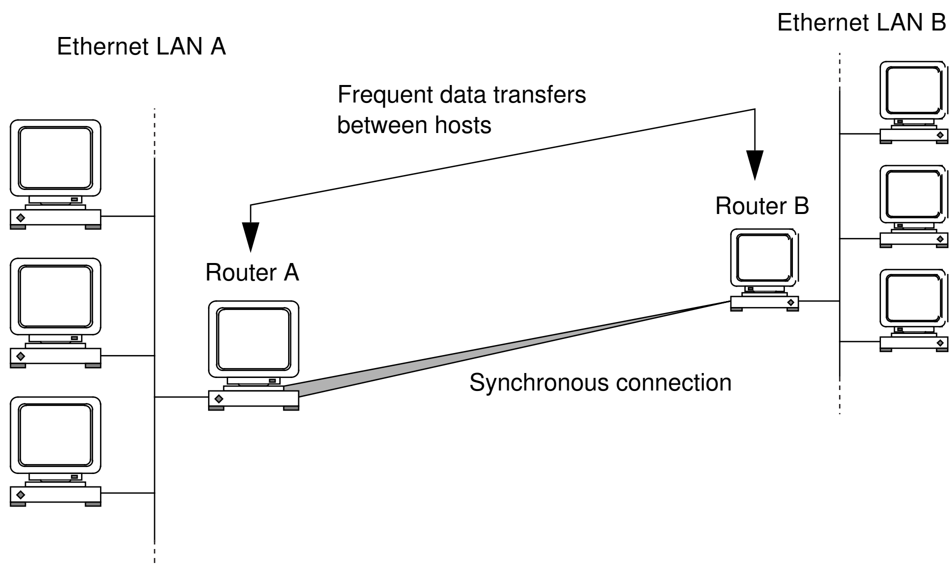


Figure 1-2 Synchronous Connection between LANs.

The synchronous connection is established through a synchronous serial interface installed in the host machine. The synchronous connection is established when Solstice PPP is started; therefore, there is no independent connection phase when a PPP link is established over a synchronous connection. Refer to the *Important Product Information for Solstice PPP* for a list of the synchronous serial interfaces with which Solstice PPP has been tested.

Asynchronous Connections

Asynchronous connections either use information carried in the data itself (software flow control), or handshake signals generated by the serial interface (hardware flow control), to control the data transmission. They provide temporary connections between two endpoints, and are typically established over private and public telephone networks or null modem links.

The cost of the connection is related to the duration of the call and the distance between endpoints. For this reason, asynchronous connections are best suited to transient network traffic, and are used typically for mobile communication, teleworking, and temporary access to Internet servers as shown in Figure 1-3.

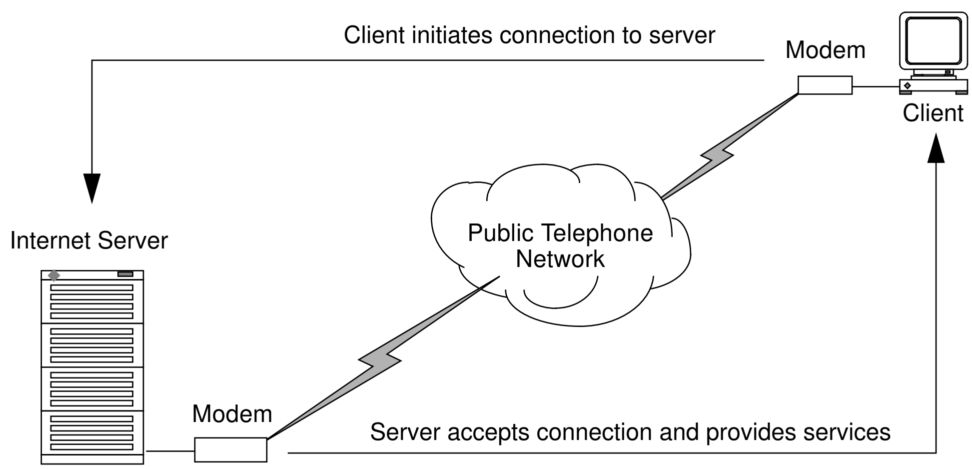


Figure 1-3 Asynchronous Connection between Client and Server

The asynchronous connection is established through an asynchronous serial interface installed in the host machine, and a modem which represents the interface between the host machine and the analog telephone network. During the connection phase, the modems at each endpoint communicate to set up the call.

Refer to the *Important Product Information for Solstice PPP* for a list of the asynchronous serial interfaces and modems with which Solstice PPP has been tested. Configuration information for these modems is contained in the modem database file `/etc/opt/SUNWconn/ppp/modems`.

You can modify the modems database file to add configuration information for additional modems, provided they support standard Hayes AT commands. Refer to the manufacturers' documentation for details of the commands used to configure your modem.

Establishing PPP over the Physical Connection

Once the physical connection is established, the two endpoints negotiate to define a common configuration for the PPP layer. For synchronous connections, the PPP layer is established each time you start Solstice PPP on your machine. For asynchronous connections, the PPP layer is established each time you initiate a call to a remote host.

During the PPP negotiation phase, the endpoints exchange Link Control Protocol (LCP) frames that contain information regarding the desired configuration and the supported protocol options. Negotiated parameters include the use of compression algorithms to improve performance over slow connections, and the use of authentication protocols to protect against unauthorized access. The policy is to converge the negotiation, if at all possible; however, if the two endpoints fail to agree on a common configuration, the link is closed automatically.

Establishing IP over PPP

Once the PPP link is established, the two endpoints negotiate to define a common configuration for the IP layer. During the IP negotiation phase, the two endpoints exchange Network Control Protocol (NCP) frames that contain information about the desired configuration. The negotiated parameters can include the IP address. This feature forms the basis of dynamic IP address allocation.

To route IP datagrams across synchronous or asynchronous PPP links, you must define the logical IP interfaces that represent the boundary between Solstice PPP and the IP layer of the Solaris operating system. These interfaces are initialized with the negotiated configuration information.

Solstice PPP supports two types of IP interface:

- point-to-point IP interfaces (`/dev/ipdptp`*n*)
- point-to-multipoint IP interfaces (`/dev/ipd`*n*)

Point-to-Point IP Interfaces (`ipdptp`*n*)

The point-to-point IP interface for Solstice PPP is `/dev/ipdptp`. This interface is used to create a direct connection for IP between exactly two hosts, as shown in Figure 1-4.

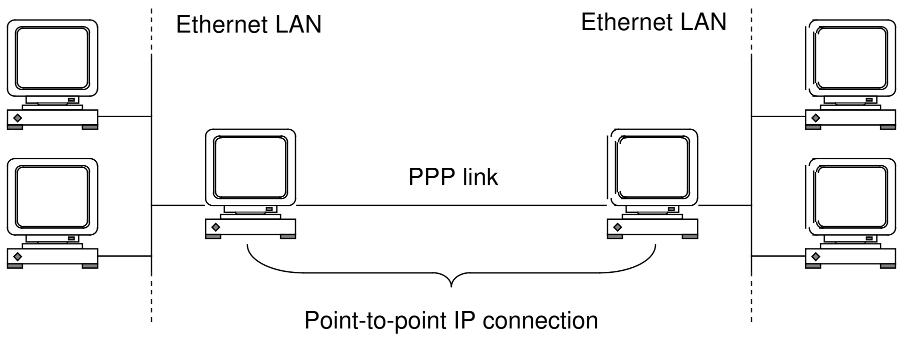


Figure 1-4 IP Point-to-Point Configuration

A point-to-point IP interface is defined by specifying a source address (or point of attachment) and a unique destination address. There is a single possible destination for the next hop, once the IP datagram has been passed to a given point-to-point interface. Therefore, the same source address can be used for multiple point-to-point connections on the same host. Multiple point-to-point connections can be used to connect to several remote hosts simultaneously.

IP point-to-point connections are supported by both synchronous and asynchronous Solstice PPP links.

Point-to-Multipoint IP Interfaces (`ipd`*n*)

The point-to-multipoint IP interface for Solstice PPP is `/dev/ipd`. A single interface is used to create IP connections between one host and several others, as shown in Figure 1-5. The effect of using point-to-multipoint interfaces is to create a virtual subnetwork, which is usually assigned its own subnetwork number.

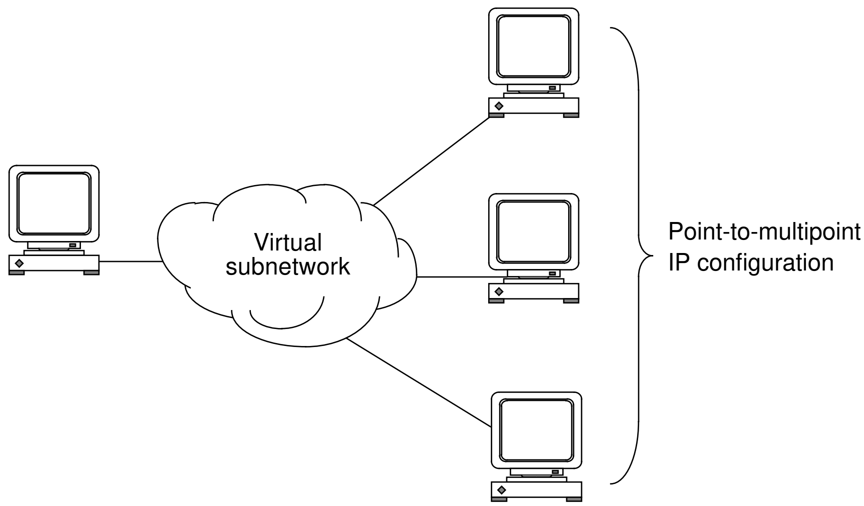


Figure 1-5 IP Point-to-Multipoint Configuration

A point-to-multipoint interface is defined by specifying a source address only. There are multiple possible destinations for the next hop, once an IP datagram has been passed to a point-to-multipoint interface. Therefore, each point-to-multipoint interface must be assigned a unique source address.

IP point-to-multipoint connections are only supported by asynchronous Solstice PPP links.

Load-Sharing for Synchronous PPP Links

Load-sharing is a Sun-specific enhancement to the standard implementation of PPP. It increases the available bandwidth by sharing the network traffic from one IP interface equally between two or more synchronous links, as shown in Figure 1-6 on page 9.

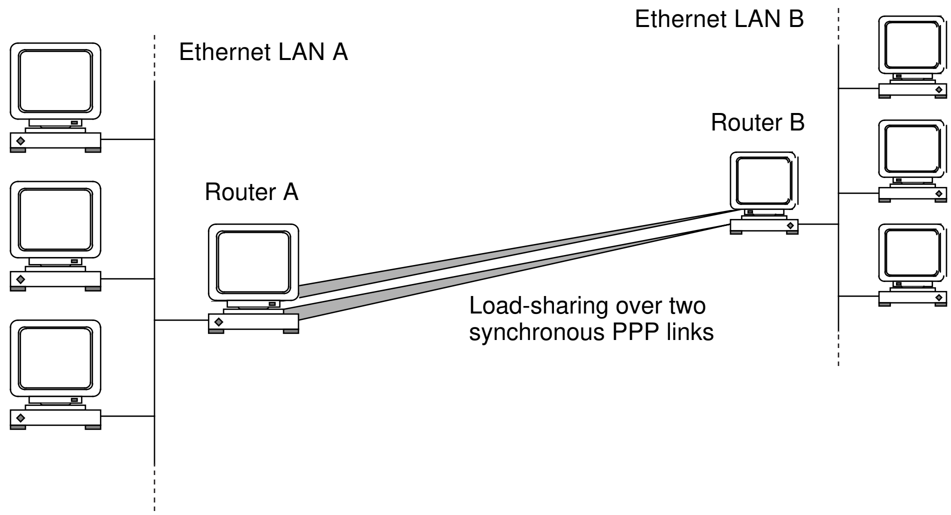


Figure 1-6 Load-Sharing over Synchronous Links

Note – For optimum performance, all of the synchronous devices used for load-sharing must be operating at the same line speed. Both ends of the link must be running Solstice PPP in order to implement load-sharing.

See “Load-Sharing over Synchronous Links” on page 82 for a detailed example that shows how to enable load-sharing using Solstice PPP.

Dynamic IP Address Allocation

Solstice PPP provides a mechanism for dynamic IP address allocation over asynchronous PPP links. This mechanism is used by clients to request IP addresses from a server at the time the PPP link is established.

Dynamic IP address allocation is enabled on the client side. When the client initiates a PPP link to the server, it requests an IP address for the connection. The server responds with its own IP address and the IP address it has allocated for the client’s local interface, and the client uses these addresses to establish a point-to-point IP connection over the existing PPP link.

To support dynamic IP address allocation on the server, you must define a pool of point-to-point IP interfaces. The server allocates an interface from this pool when it receives a request for IP addresses from a client, and the interface is returned to the pool when the PPP link is terminated.

The number of IP interfaces in the pool should be equal to the number of modems connected to the server; however, the number of clients supported by the server may be much larger. IP addresses are allocated on demand for as long as there are modems available to accept the connections.

See Chapter 4, "Editing the Configuration Files" for instructions on how to set up servers and clients to use dynamic IP address allocation.

Static and Dynamic IP Interfaces

Point-to-point IP interfaces may be *static* or *dynamic*:

- Static IP interfaces are associated with a single asynchronous device (modem). Clients that call this device are always given the same pair of IP addresses. Static IP interfaces are commonly used when a small, or predictable, number of clients connect to the server.
- Dynamic IP interfaces are associated with the available asynchronous devices (modems) on demand. Clients that connect to the server may be allocated any IP address from the pool, regardless of which device they call. Dynamic IP interfaces are commonly used when a large, or unpredictable, number of clients connect to the server.

See "Generic Internet Server Configuration" on page 94 for a detailed example of an Internet server configuration that uses dynamic IP address allocation with dynamic IP interfaces.

Peer Authentication using PAP and CHAP

Solstice PPP implements peer authentication based on the Password Authentication Protocol (PAP) and the Challenge-Handshake Authentication Protocol (CHAP) defined by RFC 1334. Peer authentication is optional, and is negotiated during the link establishment phase.

See "Editing the PPP Path Configuration File (ppp.conf)" on page 46 for instructions on how to enable and use peer authentication.

Password Authentication Protocol (PAP)

The Password Authentication Protocol (PAP) provides simple password authentication on initial link establishment. It is not a *strong* authentication method, since passwords are transmitted in clear over the link and there is no protection from repeated attacks during the life of the link.

When PAP authentication is requested by one end of the link during the link establishment phase, the other end must respond with a valid and recognized identifier and password pair. If it fails to respond, or if either the identifier or password are rejected, authentication fails and the link is closed.

PAP authentication may be requested by one end of the link only, or by both ends of the link simultaneously. If both ends request PAP authentication, they exchange identifiers and passwords. Authentication must be successful at both ends, or the link is closed.

Challenge-Handshake Authentication Protocol (CHAP)

The Challenge-Handshake Authentication Protocol (CHAP) provides password authentication on initial link establishment, based on a three-way handshake mechanism. It depends on a CHAP *secret*, known only to the authenticator and its peer, which is not transmitted over the link.

When CHAP authentication is requested by one end of the link, it generates a challenge message that includes a *challenge value*, which is calculated from the CHAP secret. The other end must respond to the challenge message with a *response value*, which is calculated from the challenge value received, and the common secret. If it fails to respond, or if the response does not correspond to that expected by the authenticator, the link is closed.

CHAP is a stronger authentication method than PAP, because the secret is not transmitted over the link, and because it provides protection against repeated attacks during the life of the link. As a result, if both PAP and CHAP authentication are enabled, CHAP authentication is always performed first.

CHAP authentication may be requested by one end of the link only, or by both ends of the link simultaneously. If both ends request CHAP authentication, they exchange challenge and response messages. Authentication must be successful at both ends, or the link is closed.

Solstice PPP Product Architecture

The primary components of Solstice PPP are shown in Figure 1-7 and are described in the following sections.

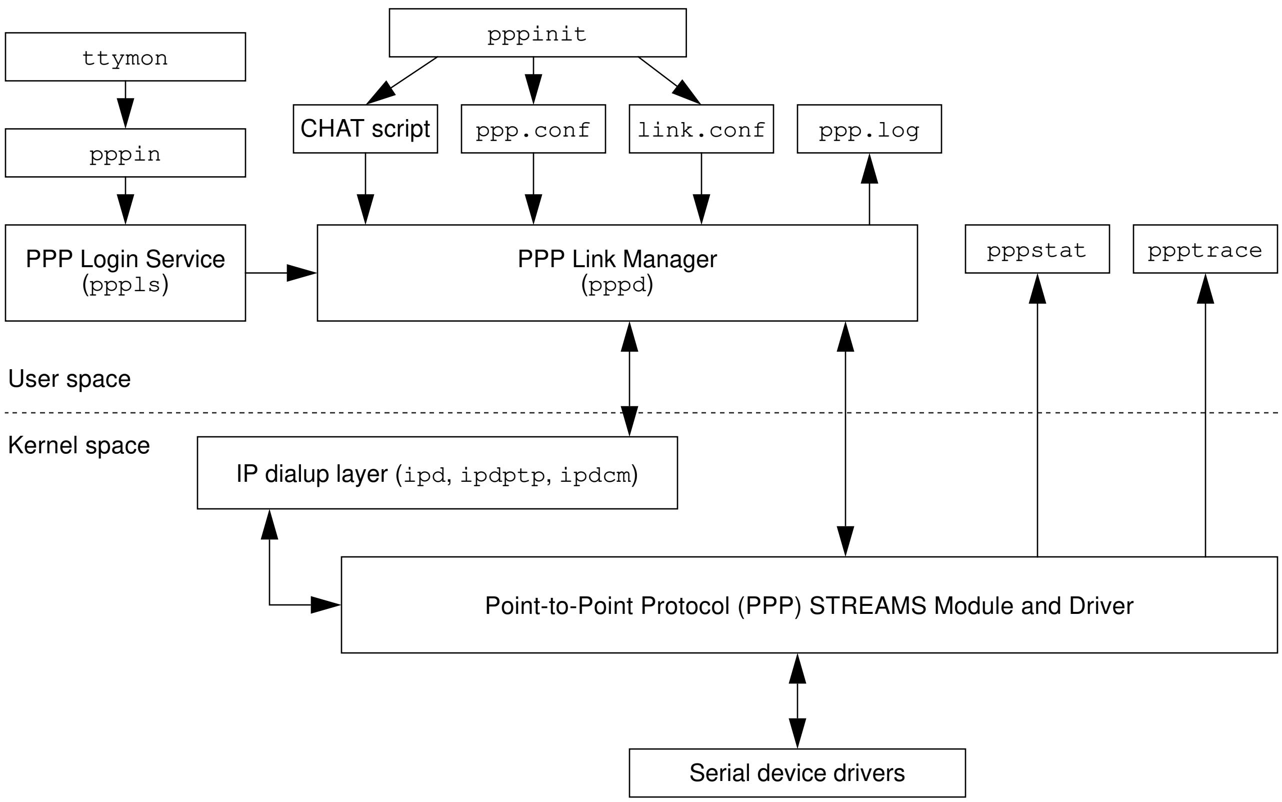


Figure 1-7 Solstice PPP Product Architecture

Point-to-Point Protocol (PPP) STREAMS Module

The PPP STREAMS module is a standard implementation of the Point-to-Point Protocol described by RFC 1661. It establishes and configures data-link level connections across serial point-to-point links, and encapsulates IP datagrams for transmission as PPP frames.

IP Dialup Layer

The IP dialup layer represents the boundary between Solstice PPP and the Solaris TCP/IP protocol suite. It includes a connection manager (`ipdcm`), which maintains and monitors the logical IP interfaces used by the IP layer to transmit datagrams across a PPP link, and the IP interfaces themselves. There are two types of IP interface associated with Solstice PPP:

- IP point-to-point interfaces (`/dev/ipdptp`*n*)
- IP point-to-multipoint interfaces (`/dev/ipd`*n*)

See "Establishing IP over PPP" on page 6 for a detailed description of these interfaces.

PPP Link Manager

The PPP link manager (`pppd`) controls all the communication links established using Solstice PPP. It reads the information in the Solstice PPP configuration files and configures the IP dialup layer and the PPP STREAMS module.

PPP Login Service

The PPP login service (`pppls`) is used to accept incoming connections from remote users. It relies on the standard UNIX login to validate the user name and password, and informs the PPP link manager of the existence of a valid PPP connection when the user is accepted.

Before a machine running Solstice PPP can accept incoming calls, the system administrator must create a user account for each remote user. This account must contain the user name and password, and must call `/usr/sbin/pppls` as the default login shell.

See "Adding User Accounts for Incoming Connections" on page 30 for a detailed description of how to create user accounts for the PPP login service.

PPP Diagnostic Utilities (`ppptrace` and `pppstat`)

Solstice PPP includes two diagnostic utilities that display information recovered from the PPP STREAMS module. The PPP trace utility `/usr/bin/ppptrace` displays PPP frame information, and the statistics collection utility `/usr/bin/pppstat` displays a cumulative record of the number and type of PPP frames sent and received since Solstice PPP was started.

PPP Initialization Script (`pppinit`*)*

The PPP initialization script, `/usr/bin/pppinit`, is used to create a basic network configuration. Use `pppinit` to configure Solstice PPP for the first time, and then modify the configuration files to create more complex PPP networks.

PPP Configuration Files (`ppp.conf` *and* `link.conf`*)*

Configuration information for Solstice PPP is contained in two configuration files. These files are created with basic configuration information using the PPP initialization script (`pppinit`). They may be modified later to add details of more complex configurations.

The file `/etc/opt/SUNWconn/ppp/ppp.conf` describes the synchronous and asynchronous paths used for IP connections over PPP. The file `/etc/opt/SUNWconn/ppp/link.conf` defines the various serial devices available for establishing PPP links, and creates a mapping between these devices and the remote hosts for outgoing connections.

See Chapter 4, "Editing the Configuration Files" for a detailed description of these files.

PPP CHAT (or Connect) Scripts

By default, the CHAT (or connect) scripts are located in the directory `/etc/opt/SUNWconn/ppp/script`. CHAT scripts contain information that defines the login dialog used to initiate asynchronous connections to remote hosts, including the login id and login password sent by the local host. You must create a unique CHAT script for each remote host to which connections will be initiated.

Some of the information contained in the CHAT script is dependent on the operating system running on the remote host. The PPP initialization script (`pppinit`) can be used to create CHAT scripts to initiate connections to hosts running a Solaris™ environment.

PPP Log File (`ppp.log`*)*

The status and error messages generated by the PPP link manager are logged in the file `/var/adm/log/ppp.log`. See "Status and Error Messages" on page 143 for a detailed description of the messages that appear in this file.

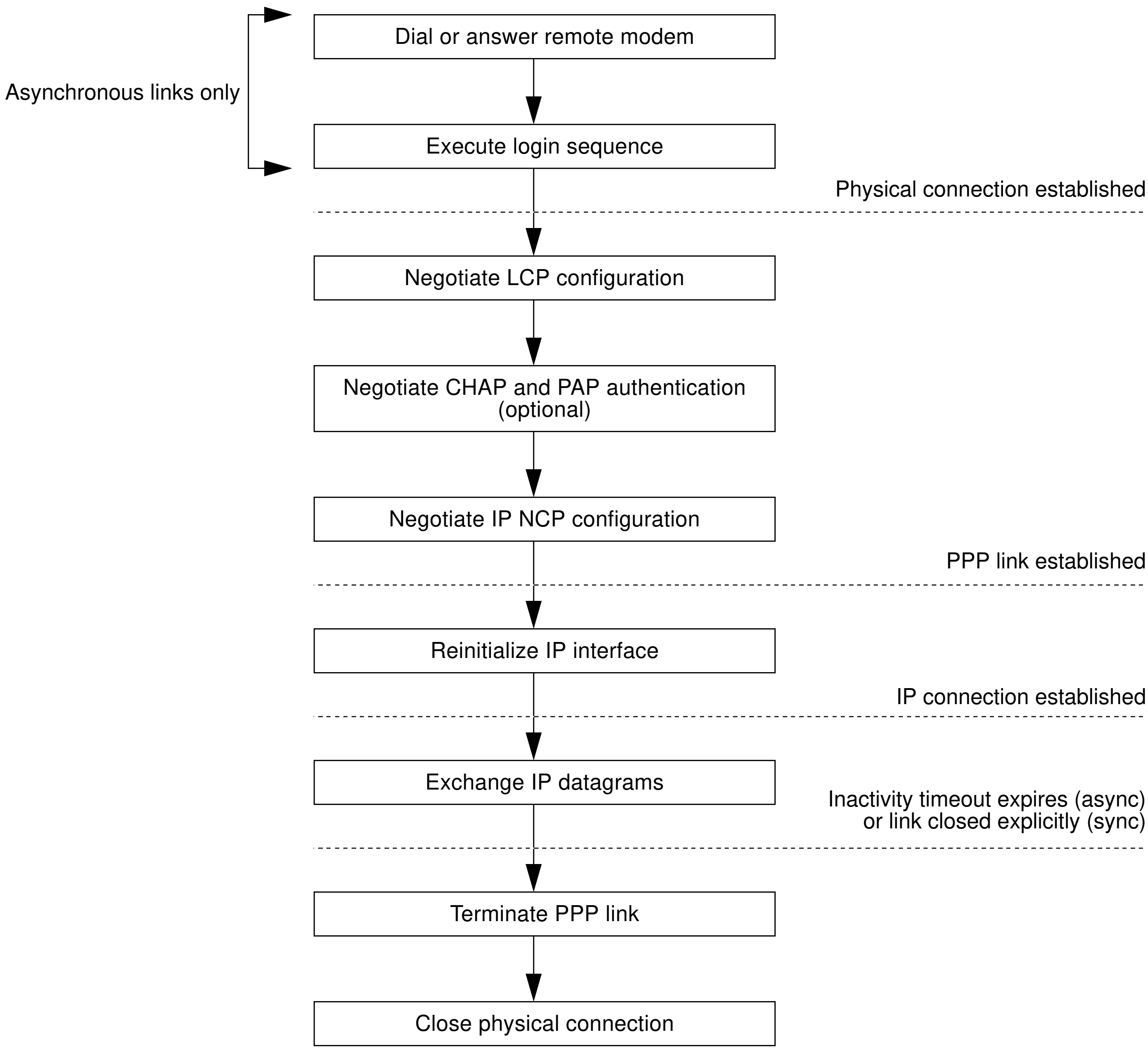


Figure 1-8 IP Connections over Synchronous and Asynchronous Links

Configuring Solstice PPP using `pppinit` 2

This chapter tells you how to create a basic configuration using the script `pppinit(1M)`, and how to add user accounts for incoming connections using `admintool(1M)`.

About `pppinit`

The script `pppinit` helps you create a basic configuration of IP point-to-point connections over synchronous and asynchronous PPP links. It requests information about your network, and enters this information in the configuration files for Solstice PPP (`ppp.conf` and `link.conf`).

You should run `pppinit` when you configure Solstice PPP for the first time. To create more complex configurations, including IP point-to-multipoint configurations, edit the configuration files to add the relevant information. See Chapter 4, "Editing the Configuration Files" for detailed instructions.

Adding Hosts for Solstice PPP

Before running pppinit, edit the file /etc/hosts to enter the source and destination addresses used for synchronous PPP links.

For example:

```
127.0.0.1       localhost
129.xxx.xxx.32  epic loghost
129.xxx.xxx.117 epic-ppp
129.xxx.xxx.119 odyssey-ppp
129.xxx.xxx.01  olympus-ppp
```

This step is necessary because Solstice PPP is started before the naming service (NIS/NIS+) when the machine is rebooted, so it reads the host information from the local files.

If you configure your machine to accept incoming asynchronous connections, you also need to create user accounts for each remote host. See "Adding User Accounts for Incoming Connections" on page 30 for detailed instructions.

Using pppinit

Caution – Running pppinit overwrites any existing configuration. Ideally, you should use pppinit to configure Solstice PPP for the first time only. See Chapter 4, "Editing the Configuration Files" for details of the keywords contained in the configuration files for Solstice PPP.

To create a configuration of IP point-to-point connections over synchronous and asynchronous PPP links:

1. **Log in as root, or become superuser.**
2. **Start** pppinit**. Note that this will overwrite any previous configurations.**

```
prompt# /usr/bin/pppinit
```

3. **Select a synchronous or an asynchronous link for configuration.**
   Type 1 or 2 to configure asynchronous links. Type 3 to configure synchronous links. Type W to exit from the script immediately,without saving any changes.

```
Welcome to the Solstice PPP 3.0 configuration script

    [1] - Asynchronous client.
    [2] - Asynchronous client/server.
    [3] - Synchronous.
    [W] - Exit Without saving

    Choice:
```

Configuring Synchronous Links

To configure synchronous links you must define one or more synchronous devices, and the IP interfaces that will be associated with them.

Configuring Synchronous Devices

1. **Enter the Unix device name of the serial port for this link.**
   The two on-board serial ports are named `zsh0` (port A) and `zsh1` (port B). High speed interface ports have names of the form `hih`*n*.

```
    Unix device for serial interface (zsh0, hih0, ...): zsh0
```

2. **Specify the type of clocking for the port.**
   Choose Internal, if the system baud rate provides clocking, or External if the clock is provided by some other device. You can also specify your own clocking parameters for the transmit (`txc`) and receive (`rxc`) clocks.

```
    Type of clocking for this port
     [1] - Internal (loopback=no txc=baud rxc=rxc)
     [2] - External (loopback=no txc=txc  rxc=rxc)
     [3] - Specify own clocking parameters

    Your choice [1]:
```

3. **Specify the line speed, if required.**
   If you chose internal clocking in the previous step, you must also specify the line speed for the device. The default value 19200 is the optimum line speed for the on-board serial interfaces (`zsh`*n*). .

```
Line speed [19200]:
```

4. **Confirm the configuration of your synchronous device.**
   The script displays the choices you selected. Type y to add this device to the configuration file.

```
    sync_device              syncdev0
    unix_device              zsh0
    line_speed               19200
    tx_clock                 baud
    rx_clock                 rxc

Ok to add this entry? [y]: y
```

Configuring IP Interfaces for Synchronous Devices

1. **Enter the local IP address.**
   Enter a hostname that appears in the file `/etc/hosts` on your machine, or enter an IP address using dot notation. Press Return to accept the default hostname.

```
Local IP address [papyrus]: 129.xxx.xxx.11
```

2. **Enter the remote IP address.**
   Enter a hostname that appears in the file `/etc/hosts` on your machine, or enter an IP address using dot notation.

```
Remote IP address: epic
```

3. **Enter the Maximum Transmission Unit (MTU).**
   By default, the MTU is set to 1500, which is the optimum value for Ethernet connections, and is sufficient in most cases. Enter a new value for the MTU (between 60 and 8232), or press Return to accept the default.

```
IP Maximum Transmission Unit (MTU) [1500]:
```

4. **Enter the names of the synchronous devices that the interface will use.**
   The script displays a list of all the synchronous devices that you defined earlier. If you associate several devices with a single synchronous interface, you enable load-sharing for this link.

```
Select synchronous interfaces for ipdptp0
Choose between:

      zsh0              zsh1

Enter a comma separated list :
      zsh0,zsh1
```

5. **Confirm the configuration of your IP interface.**
   The script displays the choices you selected. Type y to add this device to the configuration file.

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 papyrus epic mtu 1500 netmask 255.255.255.0 up

sync_path
    ip_interface          ipdptp0
    unix_device           zsh0

sync_path
    ip_interface          ipdptp0
    unix_device           zsh1

Ok to add this configuration [y]? y
```

Configuring Asynchronous Links

To configure an asynchronous link you must define one or more asynchronous devices (modems), the IP interfaces that will be associated with them, and any remote hosts to which you will be able to initiate connections.

Configuring Asynchronous Devices

1. **Specify the type of modem connected to your machine.**
   The script displays a list of the modems described in the modem database file `/etc/opt/SUNWconn/ppp/modems`. You can add other modem descriptions to this file.

```
Select one modem from your database
There are 11 modems available:
     [0]      - Null Modem
     [1]      - BocaModem V.34 DataFax
     [2]      - AT&T DataPort Express
     [3]      - Standard hayes
     [4]      - Cardinal V.34/V.FC 28.8 data/fax
     [5]      - SupraFaxModem 288
     [6]      - Hayes Accura 144B
     [7]      - Hayes Accura 288V.FC
     [8]      - Practical 14400 V32bis
     [9]      - USRobotics Sporter 14400
     [10]     - USRobotics Sporter 288

Modem type (+/- to scroll the list): 1
```

2. **Specify the Unix device for the port to which the modem is connected.**

```
List of unix devices available: [ ttya ttyb ]
Unix device for serial interface [ttya]:
```

3. **Specify the line speed for the connection between the modem and the machine.**
   For optimum performance, the line speed should be as high as possible. The default value should be used in most cases.

```
Line speed [38400]:
```

4. **Specify the type of calls that this modem will be used for.**
   You can use the modem to initiate calls to remote hosts (dial only), to accept calls from remote hosts (answer only), or to both initiate and accept calls (both).

```
What type of calls will this modem be used for:
     [1] - Dial only
     [2] - Answer only
     [3] - Both
    Choice (default 1): 3
```

5. **Confirm the configuration of your modem.**
   The script displays the choices you selected. Type y to add this device to the configuration file.

```
dialup_device pppdev0
    unix_device           ttya
    line_speed            38400
    modem                 BocaModem V.34 DataFax
    call_setup            both

Ok to add this entry [y]? y
```

6. **Repeat the previous steps until you have configured all the asynchronous devices attached to your machine.**

Configuring Modem Pools (optional)

If you define two or more asynchronous devices, you can configure a modem pool. The link manager will choose a modem from the pool to call the remote host.

1. **Indicate if you want to configure a modem pool.**

```
Do you want to set up modem pools? [n] y
```

2. **Specify the devices included in the modem pool.**
   The script displays a list of the devices you defined earlier. Enter a list of devices separated by commas or spaces.

```
Choose between:

      pppdev0       pppdev1

Enter a comma separated list to define the pool:
      pppdev0, pppdev1
```

3. **Confirm the configuration of your modem pool.**
   The script displays the choices you selected. Type y to add this device to the configuration file.

```
pool_device pool0
    pppdev0    pppdev1

Confirm to add this entry [y]: y
```

Configuring Remote Hosts for Dialing

Note – If you configured your modems to accept calls initiated by remote hosts only, you do not need to enter any remote host information.

1. **Enter the name of the remote host.**
   This is the name that will be used to identify the remote host for outgoing calls. You can choose any text string as the name of a remote host.

```
Name of the remote host: odyssey
```

2. **Enter the telephone number that will be used to initiate calls to this remote host.**

```
Phone number for this host: 1234567890
```

3. **Enter the name of file that contains the CHAT (or connect) script used to initiate calls to this remote host.**
   The CHAT script defines the dialog used to login the remote host, including the login id and the login password sent by the local host. This dialog is dependent on the operating system running on the remote host.

```
Filename of the chat script: odyssey.chat
```

   If the file you specify does not exist, the script can create a template file for making Unix connections in the directory `/etc/opt/SUNWconn/ppp/script`. You may need to edit this template if you are connecting to a remote host running another operating system.

```
The file you specified does not exist. This script can create
a template file, with default parameters for connecting
to Unix systems. Do you want to do this now? [y] y
```

Enter the login id that your machine will send to the remote host to initiate a call. There must be a corresponding user account created on the remote host. The login id must be eight characters or less.

```
Login id sent to odyssey (up to 8 characters): ppp0
```

Enter the password that your machine will send with the login id. Type it again to confirm. The password you type is not displayed on the screen.

```
Password:
Re-enter Password:
```

4. **Indicate if you want to use a specific modem for making calls to this remote host.**
   Type n if you have only one modem, or if you want to be able to use any modem to call this remote host. If you type y in response to this question, the script will list the modems and modem pools you configured earlier. Enter the name of the modem or modem pool you want to use to call this remote host.

```
Do you want to set a specific pool or modem for this host [n]:
```

5. **Confirm the configuration of your remote host.**
   The script displays the choices you selected. Type y to add this host to the configuration file.

```
remote_host epic
    phone_number           1234567890
    chat_script            odyssey.chat

Ok to add this entry [y]? y
```

6. **Repeat the previous steps until you have defined all the remote hosts to which your machine will initiate calls.**

Configuring IP Interfaces to Answer Calls

Note – If you configured your modems to initiate outgoing calls only (dial only), you cannot configure IP interfaces to answer calls.

1. **Indicate that you want to set up an IP interface to accept incoming calls.**
   By default, IP interfaces are assigned names of the form `ipdptp`*n*.

```
IP information for ipdptp1

Do you want to use this interface to accept calls
initiated by a remote host [y]? y
```

2. **Enter the local IP address for this interface.**
   Enter a hostname that appears in the file `/etc/hosts` on your machine, or enter an IP address using dot notation. Press Return to accept the default hostname.

```
Local IP address information
    Enter an IP host that is listed in your hosts map
    or an IP address with internet dot notation

    Local IP address [papyrus]: 129.xxx.xxx.11
```

3. **Enter the remote IP address for this interface.**
   Enter a hostname that appears in the file `/etc/hosts` on your machine, or enter an IP address using dot notation.

```
Remote IP address information
    Enter an IP host that is listed in your hosts map
    or an IP address with internet notation

    Remote IP address: odyssey
```

4. **Enter the IP netmask for this interface.**

```
IP netmask for this interface [255.255.255.0]:
```

5. **Enter the Maximum Transmission Unit (MTU).**
   By default, the MTU is set to 1500 bytes, which is the optimum value for Ethernet connections, and is sufficient in most cases. Enter a new value for the MTU (between 60 and 8232), or Press Return to accept the default.

```
IP Maximum Transmission Unit (MTU) [1500]:
```

6. **Enter the login id that you expect to receive from the remote host associated with this interface.**
   You must create a user account on your machine that corresponds to this login id. The login id must be eight characters or less.

```
To accept calls using this interface, you must specify the login
id you expect to receive from the remote host. You must also
create a user account that associates this login id with a valid
password.
Login id expected from remote host: od-login
```

Configuring IP Interfaces to Dial Calls

Note – If you configured your modems to accept incoming calls only (answer only), you cannot configure IP interfaces to dial calls.

1. **Indicate that you want to set up an IP interface to initiate calls to remote hosts.**

```
Do you want to use this interface to initiate
calls to a remote host [y]? y
```

2. **Enter the name of the remote host you want to be able to call.**
   The script displays a list of the remote hosts you defined earlier. Enter the name of one of these hosts.

```
Remote host for this interface: odyssey
```

3. **Specify the timeout for calls to this remote host.**
   By default, the connection will timeout after 120 seconds of inactivity.

```
inactivity_timeout[120]: 60
```

Enabling Dynamic IP Address Allocation

Note – If you configured your modems to initiate outgoing calls only (dial only), you are given an opportunity to enable dynamic IP address allocation.

1. **Indicate that you want to use dynamic IP address allocation.**

```
IP information for ipdptp0
Do you want to use dynamic IP allocation [y]? y
```

2. **Enter the IP netmask for this interface.**

```
IP netmask for this interface [255.255.255.0]:
```

3. **Enter the name of the remote host you want to be able to call.**
   The script displays a list of the remote hosts you defined earlier. Enter the name of one of these hosts.

```
Remote host for this interface: odyssey
```

4. **Specify the timeout for calls to this remote host.**
   By default, the connection will timeout after 120 seconds of inactivity.

```
inactivity_timeout[120]: 60
```

Saving your Configuration

You can either exit from `pppinit` without saving the configuration, or you can save the configuration to file.

```
Welcome to the Solstice PPP 3.0 configuration script

    [1] - Asynchronous client
    [2] - Asynchronous client/server
    [3] - Synchronous
    [W] - Exit Without saving
    [E] - Exit and Save
```

If you type E (Exit and Save), the configuration is saved to file. You are given an opportunity to send the modem configuration command contained in the file `/etc/opt/SUNWconn/ppp/modems` directly to the modem.

```
pppinit can set and save the configuration for all server
modems according the modem database
       Do you want to do it now [y]?
```

Adding User Accounts for Incoming Connections

If you configure your modem to accept incoming calls, you must create a user account for each remote host. If your machine will initiate calls only, you can omit this step.

1. **Log in as root, or become superuser.**

2. **Start** `admintool`**.**

```
prompt# /usr/bin/admintool &
```

3. **Select User Account Manager, as shown in Figure 2-1.**

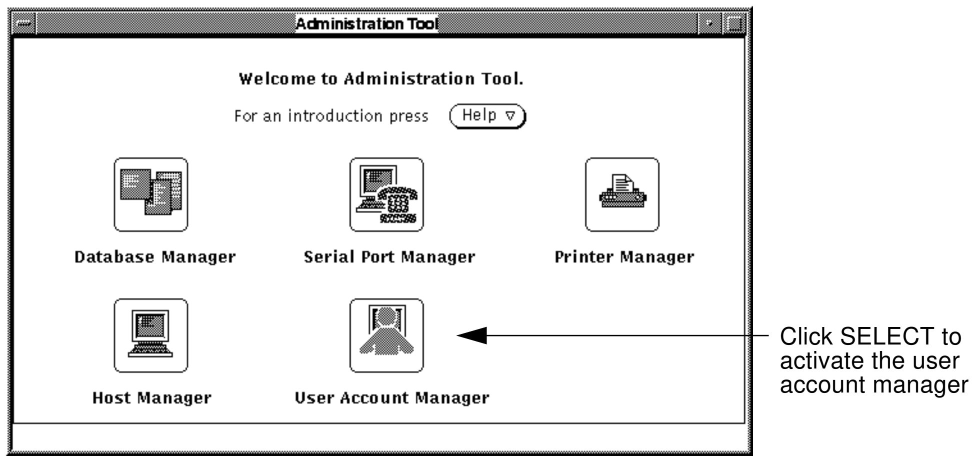


Figure 2-1 Starting the User Account Manager

4. **Select None as the Naming Service (Use** `/etc/hosts` **and** `/etc/passwd` **on local machine), as shown in Figure 2-2. Click SELECT on Apply to display a list of the current user accounts.**

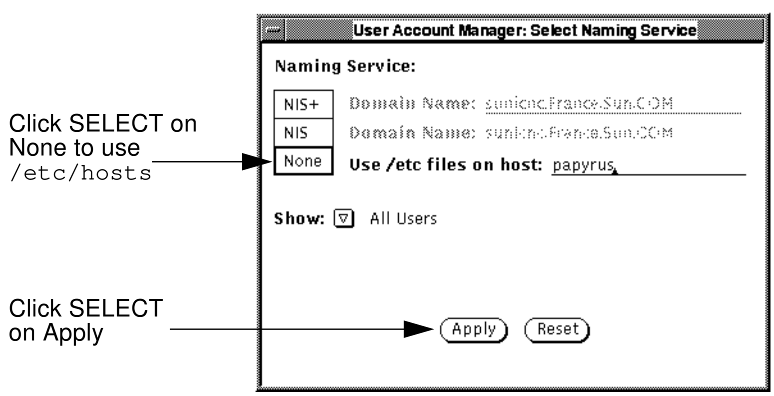


Figure 2-2 Selecting the Naming Service

5. **Choose Add user... from the Edit menu, as shown in Figure 2-3.**
   Note that a default account for the user `ppp` has been added automatically, but that you still need to set a password.

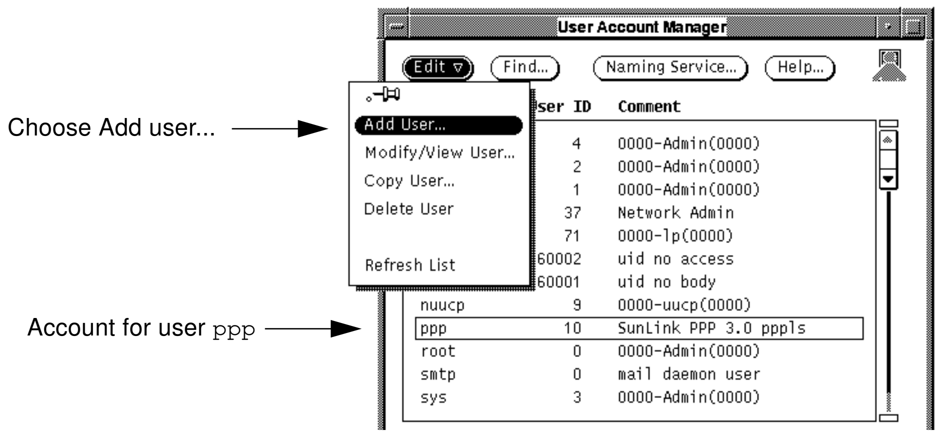


Figure 2-3 Adding a New User Account

6. **Enter a User Name, a User ID, and Login Shell for this account.**
   The User Name must be the same as the PPP login id assigned to the remote client, and the Login Shell must be `/usr/sbin/pppls`.

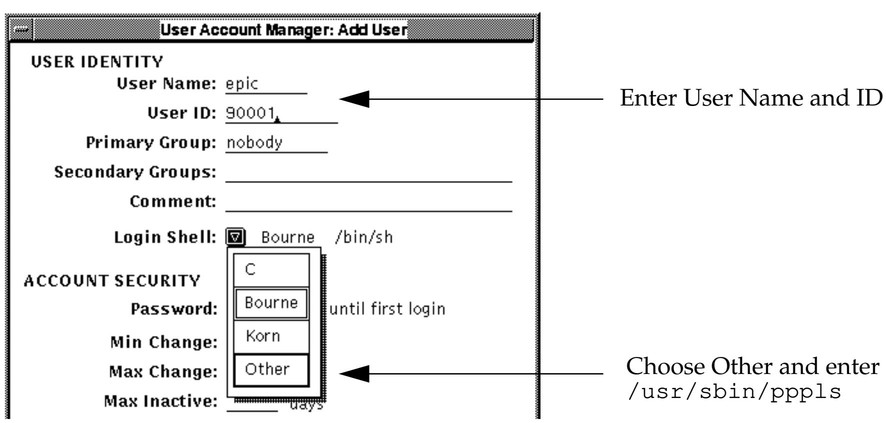


Figure 2-4 Entering Account Information

7. **Enter the password expected from the client when it logs in, as shown in Figure 2-5.**

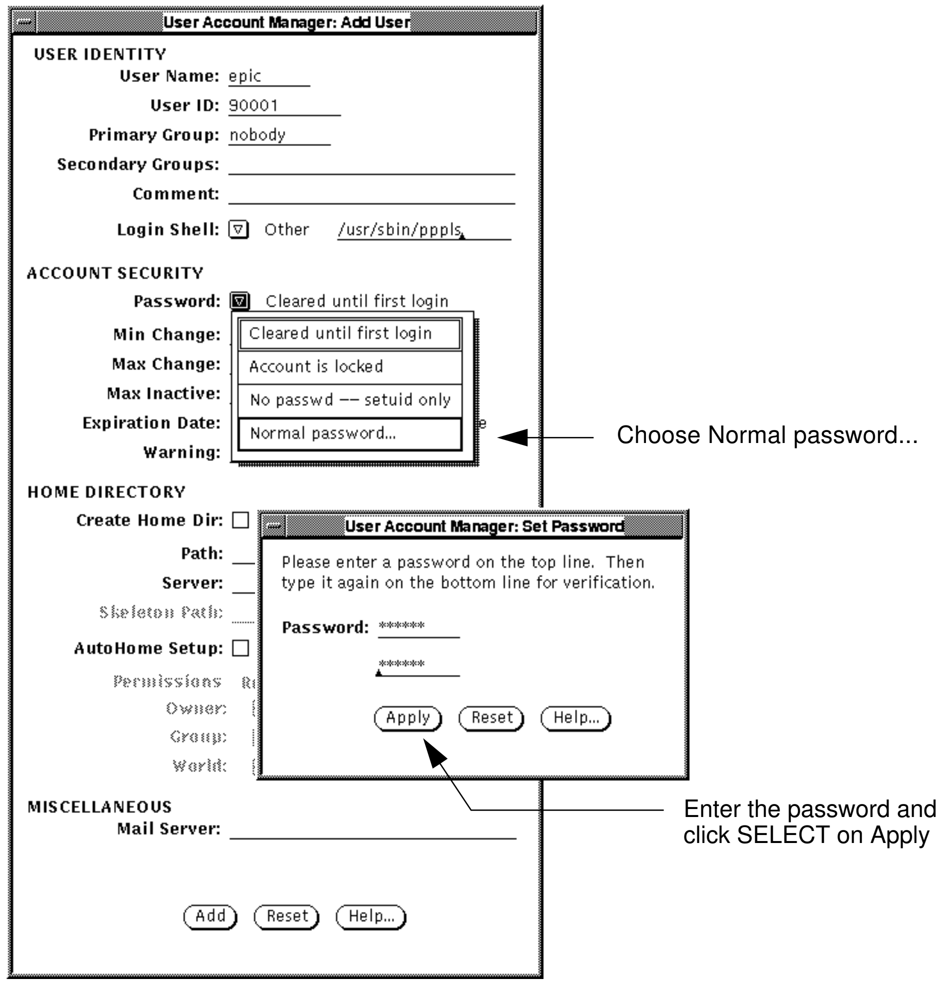


Figure 2-5 Adding a User Password

8. **Click SELECT on Add to add the user to the list of accounts.**
Continue adding accounts until there is an account for each client.

Starting and Using Solstice PPP 3

This chapter explains how to start Solstice PPP on your machine, how to establish IP connections across a PPP link, and how to close connections. It assumes that you have installed and configured Solstice PPP on your machine.

Starting Solstice PPP

Once you have modified the configuration files to describe your network configuration, you can start Solstice PPP on your machine. Starting Solstice PPP initializes the PPP link manager, PPP login service, and IP dialup layer, and establishes each of the synchronous links you defined.

Starting Solstice PPP at Boot Time

By default, Solstice PPP is started each time you reboot your machine. It is started before the naming services (NIS/NIS+) are enabled, so you must ensure that the host information required when Solstice PPP is started is contained in the local files `/etc/hosts` and `/etc/passwd`.

Starting Solstice PPP Manually

If you stop Solstice PPP, or modify the configuration files, you can restart it manually without rebooting your machine.

To execute the initialization script for Solstice PPP manually:

1. **Log in as root or become superuser.**
2. **Execute the PPP initialization script with the start option, by typing:**

```
prompt# /etc/init.d/ppp start
```

3. **If your machine is configured as a server—that is, it can accept incoming calls—run** `pppsetmod` **to configure your modem.**

Establishing PPP Links

PPP links are established automatically by the PPP link manager when it receives a request for an IP connection. This process is usually transparent; however, you may see minor differences in the response, depending on the type of link you are using.

IP Connections over Synchronous Links

Once a synchronous link is created, it is available for as long as both endpoints remain active. To establish an IP connection across a synchronous link, type a UNIX command and the hostname or IP address that corresponds to the remote end of the link. You should see an almost immediate response to your command, subject to the line speed of your link.

For example, to establish a `telnet(1)` connection to a remote host over a synchronous link, type:

```
prompt% /usr/bin/telnet hostname
Trying xxx.xxx.xxx.38 ...
Connected to hostname.
Escape character is '^]'.

UNIX(r) System V Release 4.0 (hostname)
```

IP Connections over Asynchronous Links

To establish an IP connection across an asynchronous link, type a UNIX command and the hostname or IP address that corresponds to the remote end of the link. When you establish an asynchronous link, there is always a short delay before connection. This is the time it takes to complete the PPP negotiation and to establish the asynchronous link. The delay is typically less than one minute.

You can monitor the progress of the connection by looking at the log file `/var/opt/SUNWconn/ppp.log`.

IP Connections using Dynamic IP Address Allocation

When dynamic IP address allocation is enabled, you need to invoke a connection phase during which the server provides the client with the IP addresses it needs to establish an IP connection over the PPP link.

Use `pppconn(1M)` to establish a PPP link to the server:

1. **Use `pppconn` to create the PPP link, to recover the IP information from the server, and to initialize the IP interface.**

```
prompt# /usr/bin/pppconn hostname
Connect to hostname
Connecting...
Connected over ipdptp0
Local IP address : xxx.xxx.xxx.119
Remote IP address: xxx.xxx.xxx.38
IP mtu           : 1500
```

2. **Establish an IP connection over the PPP link by typing a command and hostname. For example:**

```
prompt% /usr/bin/telnet hostname
Trying xxx.xxx.xxx.38 ...
Connected to hostname.
Escape character is '^]'.

UNIX(r) System V Release 4.0 (hostname)
```

If there is only one IP interface defined at the client side, you can invoke pppconn without any arguments:

```
prompt# /usr/bin/pppconn
Connect to hostname
Connecting...
Connected over ipdptp0
Local IP address : xxx.xxx.xxx.119
Remote IP address: xxx.xxx.xxx.01
IP mtu           : 1500
```

You can also specify a given IP interface by invoking pppconn with the –i option:

```
prompt# /usr/bin/pppconn –i ipdptpn
Connect to hostname
Connecting...
Connected over ipdptpn
Local IP address : xxx.xxx.xxx.119
Remote IP address: xxx.xxx.xxx.01
IP mtu           : 1500
```

Note – When a link is established using pppconn, it remains open until it is closed explicitly, or until the inactivity timer expires. If the link fails while it is still in use, you must stop the IP application before using pppconn to re-establish the link. This step is necessary because you cannot guarantee that the server will assign the same IP address for the new link.

Closing PPP Links

Synchronous links are established when Solstice PPP is started, and are usually closed when Solstice PPP is stopped.

Asynchronous links are closed automatically after a specified period of inactivity. By default, the inactivity timeout period is set to 120 seconds. You can alter the inactivity timeout by changing this parameter in the file /etc/opt/SUNWconn/ppp/ppp.conf. See “Defining Asynchronous Paths (dialup_path)” on page 55 for detailed instructions.

Note – There is an inactivity timeout specified at both endpoints. The connection is closed when the shortest inactivity timeout expires. If the inactivity timeout is set to zero at both ends of the link, the link stays open until closed explicitly.

Use `pppdisc(1M)` to close a connection manually, without waiting for the inactivity timeout to expire, and without stopping and starting Solstice PPP.

```
prompt# /usr/bin/pppdisc hostname
Disconnect from hostname
Disconnecting ...
Disconnected
```

If there is only one IP interface defined at the client side, you can invoke `pppdisc` without any arguments:

```
prompt# /usr/bin/pppdisc
Disconnect from hostname
Disconnecting ...
Disconnected
```

You can also specify a given IP interface by invoking `pppdisc` with the `–i` option:

```
prompt# /usr/bin/pppdisc –i ipdptpn
Disconnect from hostname
Disconnecting ...
Disconnected
```

Stopping Solstice PPP

To stop Solstice PPP manually:

1. **Log in as root or become superuser.**
2. **Execute the PPP initialization script with the stop option, by typing:**

```
prompt# /etc/init.d/ppp stop
```

Solstice PPP closes any IP connections and terminates all the PPP links, before stopping the PPP link manager.

Editing the Configuration Files 4

This chapter describes the format and syntax of each of the configuration files associated with Solstice PPP, and explains how to modify these files to generate more complex network configurations.

Configuration Files for Solstice PPP

Solstice PPP uses the information contained in the following files to establish IP connections over synchronous and asynchronous PPP links:

PPP Path Configuration File (`ppp.conf`*)*

The PPP path configuration file (`/etc/opt/SUNWconn/ppp/ppp.conf`) describes the synchronous and asynchronous (or dialup) paths used for IP connections over Solstice PPP links. It includes the `ifconfig(1M)` commands that establish the logical IP interfaces for Solstice PPP.

See "Editing the PPP Path Configuration File (ppp.conf)" on page 46 for a detailed description of the parameters set in this file.

Link Configuration File (`link.conf`)

The link configuration file (`/etc/opt/SUNWconn/ppp/link.conf`) describes the synchronous and asynchronous devices used to create the PPP link. It includes dialing information for initiating asynchronous links to remote hosts.

See "Editing the Link Configuration File (link.conf)" on page 64 for a detailed description of the parameters set in this file.

Modems Database File (`modems`)

The modems database file (`/etc/opt/SUNWconn/ppp/modems`) contains a description of the modems recognized by Solstice PPP, including the `AT` commands used by Solstice PPP to initialize the modems for incoming and outgoing connections.

See "Editing the Modem Database File (modems)" on page 70 for instructions on how to add new modem descriptions to this file.

CHAT (or Connect) Scripts

The CHAT (or connect) scripts define exchange of information between the endpoints, which occurs during the link establishment phase. By default, the CHAT scripts are located in the directory `/etc/opt/SUNWconn/ppp/script`. You can change the location of the scripts by editing the location definition in the file `link.conf` and moving the files. CHAT scripts may be non-interactive (all the information to be exchanged is contained in the script) or interactive (the script prompts for user input).

You must create one CHAT script for each remote host to which the local host will initiate calls. Some of the information contained the CHAT script is dependent on the operating system running on the remote host. The PPP initialization script (`pppinit`) can create a template file for connecting to hosts running a Solaris environment. See "Editing the CHAT (or Connect) Scripts" on page 72 for instructions modify the template file for other operating systems.

User Account Files (`/etc/passwd` and `/etc/shadow`)

To accept asynchronous PPP connections initiated by a remote host, there must be a corresponding user account created on the local machine. Use `admintool(1M)` to add the entries of the type shown in Figure 4-1 to the local user account files `/etc/passwd` and `/etc/shadow`.

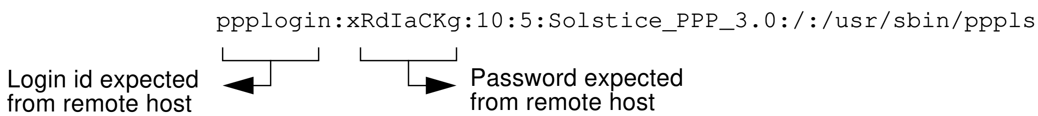


Figure 4-1 PPP Login Entry in `/etc/passwd`

Combining the Configuration Files

Figure 4-2 shows how the files `ppp.conf` and `link.conf` are combined to define a point-to-point IP connection over a synchronous PPP link.

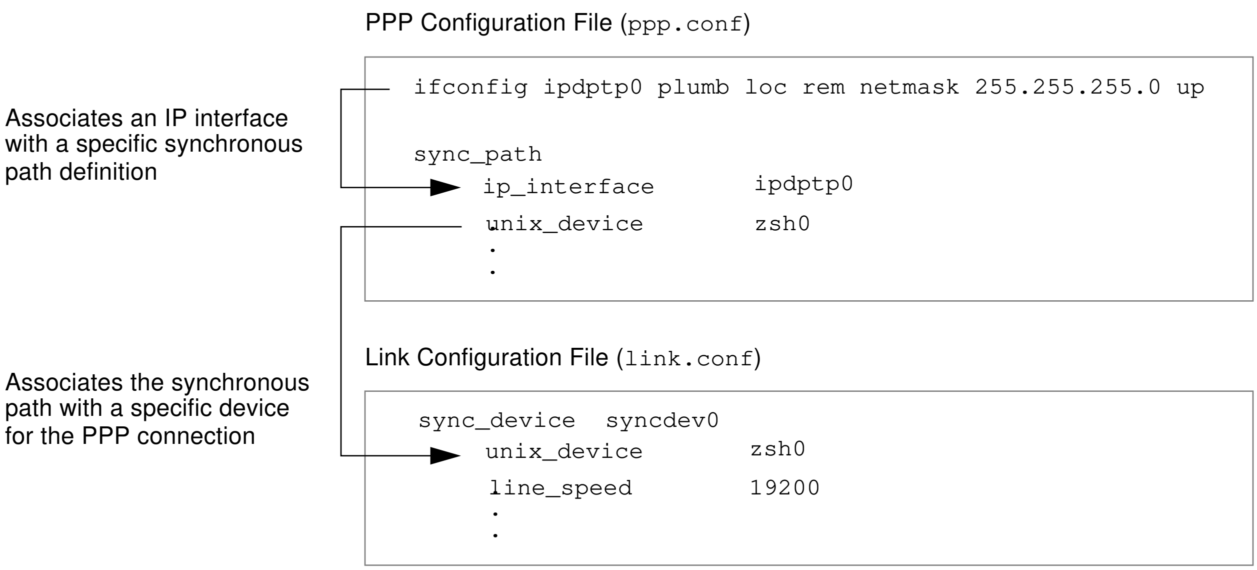


Figure 4-2 Configuration Files for Synchronous Connections

Figure 4-3 on page 44 shows the relationship between the files `ppp.conf`, `link.conf`, and the CHAT script used to initiate a point-to-point IP connection to a remote host over an asynchronous PPP link.

Figure 4-4 on page 45 shows the relationship between the files `ppp.conf`, `link.conf`, and the user account used to accept a point-to-point IP connection from a remote host over an asynchronous PPP link.

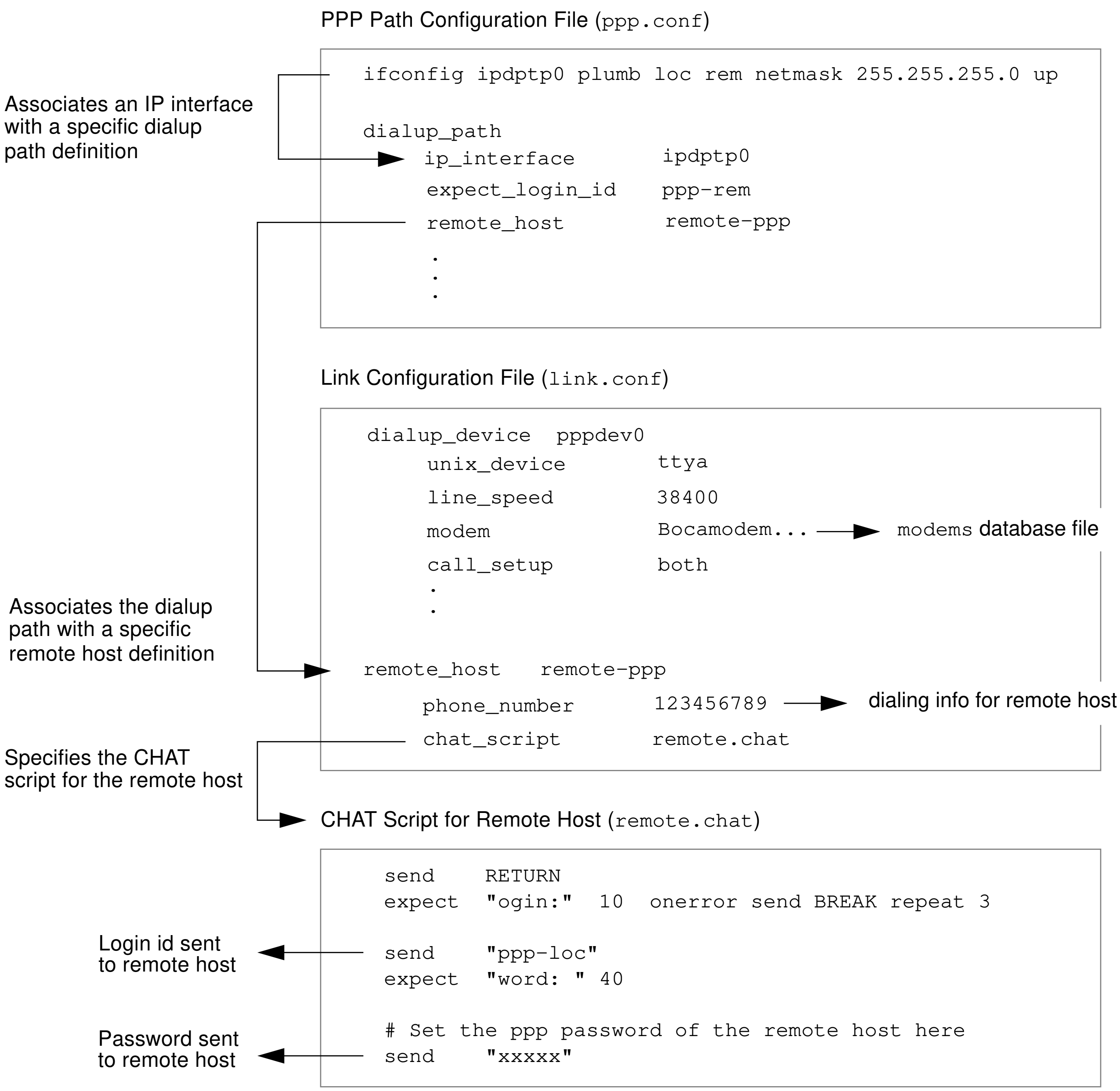


Figure 4-3 Configuration Files for Initiating Asynchronous Connections

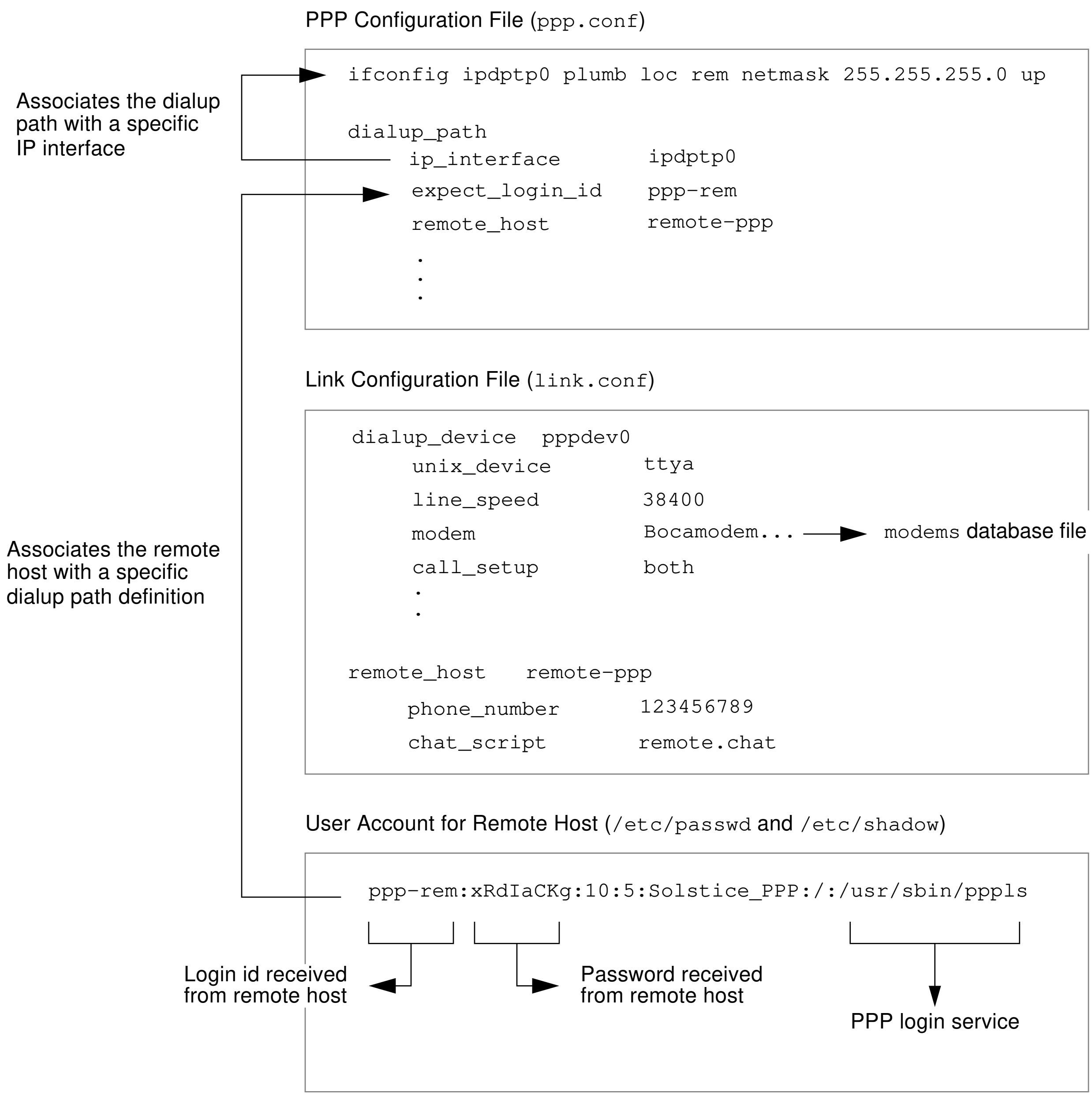


Figure 4-4 Configuration Files for Accepting Asynchronous Connections

Editing the PPP Path Configuration File (`ppp.conf`)

The PPP path configuration file (`/etc/opt/SUNWconn/ppp/ppp.conf`) describes the synchronous and asynchronous (or dialup) paths used for IP over Solstice PPP. It includes the `ifconfig(1M)` commands that establish the logical IP interfaces for Solstice PPP.

Defining IP Interfaces using `ifconfig`

The `ifconfig(1M)` commands that are contained in the PPP configuration file establish the point-to-point (`ipdptp`*n*) and point-to-multipoint (`ipd`*n*) IP interfaces for Solstice PPP. These commands are executed when Solstice PPP is started, to assign a network address to each interface and to configure the network parameters.

Synopsis

The `ifconfig` commands used to establish the IP interfaces for Solstice PPP have the general form:

```
ifconfig interface plumb source [dest] netmask mask mtu mtu up
```

Arguments

interface

The name and type of the IP interface. The IP interfaces for Solstice PPP are ipdptp*n* (point-to-point) and ipd*n* (point-to-multipoint), where *n* is a number. By convention, IP interfaces are numbered sequentially from zero.

For example, `ipdptp0`, `ipdptp1`, `ipdptp2`, or `ipd0`, `ipd1`, `ipd2`

`plumb`

Opens the device associated with the interface name, and sets up the STREAMS that enable TCP/IP to use the device.

source

An IP address (dot notation) or hostname that represents the source address, or point of attachment, for point-to-point and point-to-multipoint IP interfaces.

dest

Point-to-point interfaces only. An IP address (dot notation) or hostname that represents the destination address for a point-to-point IP interface.

`netmask` *mask*

Specifies how much of the IP address to reserve for dividing networks into subnetworks. The mask can be entered in dot notation, or in hexadecimal when preceded by `0x`.

`mtu` *mtu*

Sets the maximum transmission unit (MTU) for the interface. The MTU must be in the range 60 to 8232 bytes, and is usually set to 1500, which is the optimum value for Ethernet networks.

`up`

Marks the interface *up*—that is, active. You can disable an interface temporarily by marking it *down*. The IP interfaces associated with synchronous PPP links are usually marked *up* by default. If the IP interface associated with an asynchronous PPP link is marked *up*, the link manager will attempt to establish the link automatically when the IP layer passes an IP datagram to the interface.

Examples

To establish a point-to-multipoint IP interface for Solstice PPP, include an `ifconfig` command of the form:

```
ifconfig ipd0 plumb papyrus netmask 255.255.255.0 mtu 1500 up
```

To establish a point-to-point IP interface for Solstice PPP, include an `ifconfig` command of the form:

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 papyrus epic netmask 255.255.255.0 mtu 1500 up
```

Note that the interface can be fully defined by concatenating multiple `ifconfig` commands, as shown in the previous example.

Defining a Pool of IP Interfaces for Dynamic IP Address Allocation

To configure a server to support dynamic IP address allocation, you must define a *pool* of point-to-point IP interfaces that will be assigned to the clients as required. These interfaces are always marked *down* by default.

For example, to create a pool of *n* point-to-point IP interfaces for dynamic IP address allocation:

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 local rem1 netmask 255.255.255.0 mtu 1500 down

ifconfig ipdptp1 plumb
ifconfig ipdptp1 local rem2 netmask 255.255.255.0 mtu 1500 down

ifconfig ipdptp2 plumb
ifconfig ipdptp2 local rem3 netmask 255.255.255.0 mtu 1500 down
.
.
ifconfig ipdptpn plumb
ifconfig ipdptpn local remn netmask 255.255.255.0 mtu 1500 down
```

The number of interfaces in the pool should equal the number of asynchronous devices (modems) attached to the server, and the maximum number of interfaces in the pool is 512. The total number of clients supported by the server may be much greater.

If you have a small number of clients, or the same number of clients and modems, you can assign the interfaces *statically*. In this case, when a client requests an IP address, it is always assigned the same one from the pool.

If you have a large number of clients, or many more clients than modems, you can assign the interfaces *dynamically*. In this case, when a client requests an IP address, it is assigned one from the pool, but there is no guarantee that it will always receive the same one.

See “Defining Asynchronous Paths (dialup_path)” on page 55 for instructions on how to assign static and dynamic IP interfaces.

Defining Synchronous Paths (`sync_path`)

Synchronous paths are identified in the file `ppp.conf` by the keyword `sync_path`, which starts each definition. They are always associated with point-to-point IP interfaces.

Synopsis

Synchronous path definitions have the following general form:

```
sync_path
    ip_interface    ipdptpn
    unix_device     device_name
    .
    .
    .
```

Keywords

`sync_path`
: Mandatory parameter for synchronous paths. Indicates the start of a synchronous path definition.

`ip_interface ipdptp`*n*
: Mandatory parameter for synchronous paths. Associates the synchronous path with one of the point-to-point IP interfaces defined in the `ifconfig` section of the file. Load-sharing is enabled if two or more synchronous paths share the same IP interface.

`unix_device` *device*
: Mandatory parameter for synchronous paths. Associates the synchronous path with one of the synchronous devices defined in the file `link.conf`. The value *device* must correspond to a synchronous serial interface installed in your machine.

  For example, the device names of the form `zsh`*n* associate the path with one of the on-board serial interfaces. The device names of the form `hih`*n*, associate the path with a high-speed serial interface (HSI).

`default_route`

Optional parameter for synchronous paths. Adds the route to the routing table as the default destination. The route is removed when the IP interface is marked *down*.

`accept_any_ip_addr` *state*

Optional parameter for synchronous paths. Accepts the IP addresses provided by the remote host, even if they differ from the IP addresses assigned to the interface locally.

The value *state* can be `on` (enabled) or `off` (disabled). The default value is `off`.

`link_monitor` *state*

Optional parameter for synchronous paths. Indicates the current state of the link monitor. When enabled, the link monitor sends periodic echo requests to the remote host. If the remote host fails to respond after a specified number of requests, the link monitor assumes that the link has failed for some reason. It marks the IP interface associated with the synchronous path *down* to stop the transmission of more IP datagrams across the failed link.

The value *state* can be `on` (enabled) or `off` (disabled). The default value is `off`.

`link_monitor_timer` *seconds*

Optional parameter for synchronous paths. Specifies the number of seconds which elapse between consecutive echo requests generated by the link monitor.

The value *seconds* can be any integer greater than zero. The default value is 5 seconds.

`link_monitor_retries` *max_retries*

Optional parameter for synchronous paths. Specifies the number of unanswered echo requests generated by the link monitor before the remote host is considered unreachable and the IP interface is disabled.

The value *max_retries* can be any integer greater than zero. The default value is 12.

`ppp_link_name` *name*

Optional parameter for synchronous and asynchronous paths. Assigns a name that is used by `ppptrace` and `pppstat` to identify the link. The value *name* can be any character string.

`ipcp_compression` *state*

Optional parameter for synchronous and asynchronous paths. Indicates the current state of the header compression facility, which uses Van Jacobsen compression to improve performance over slow links.

The value *state* can be `vj` (enabled) or `off` (disabled). The default value is `vj`.

`lcp_mru` *mru*

Optional parameter for synchronous and asynchronous paths. Specifies the maximum receive unit (MRU) for the local machine. This parameter is carried in the LCP `Configure-request` frame, and sets the maximum transmission unit (MTU) for the remote host. See Appendix A, "PPP Link Operation" for more information.

By default, the value *mru* is set to 1500 bytes for Ethernet networks.

`lcp_restart_timer` *seconds*

Optional parameter for synchronous and asynchronous paths. Specifies the number of seconds which elapse between consecutive LCP `Configure-request` frames. Increasing the LCP restart timer may be necessary when connecting over long delay networks, such as satellite connections. See Appendix A, "PPP Link Operation" for more information.

The value *seconds* can be any integer greater than zero. The default value is 3 seconds.

`lcp_max_restart` *max_restart*

Optional parameter for synchronous and asynchronous paths. Specifies the number of unanswered LCP `Configure-request` frames generated before the endpoint is considered unreachable and the IP interface is marked as *down*. See Appendix A, "PPP Link Operation" for more information.

The value *max_restart* can be any integer in the range 1 to 255. The default value is 10. If the value *max_restart* is set to 255, LCP `Configure-request` frames are generated periodically until the remote host finally responds.

`expect_authentication` *mode*

Optional parameter for synchronous and asynchronous paths. Indicates that the local host will request authentication from remote hosts, and the authentication protocol to be used. If authentication is enabled, remote hosts must authenticate themselves successfully, or the connection is closed.

The value *mode* can be `off` (no authentication), `pap` (authentication using PAP), `chap` (authentication using CHAP), or `pap|chap` (authentication using both PAP and CHAP). The default value is `off`.

If both PAP and CHAP are enabled, CHAP authentication is performed first. If the remote host does not support CHAP authentication, it is allowed to participate in PAP authentication only.

`expect_pap_id` *pap_id*

Mandatory parameter, if the local host requests PAP authentication. Specifies the PAP identifier expected from a remote host. The value *pap_id* can be any string between 0 and 255 characters in length. A zero length value is represented by: `expect_pap_id ""`

`expect_pap_passwd` *pap_passwd*

Mandatory parameter, if the local host requests PAP authentication. Specifies the PAP password expected from a remote host. The value *pap_passwd* can be any string, between 0 and 255 characters in length. A zero length value is represented by: `expect_pap_passwd ""`

`expect_chap_name` *chap_name*

Mandatory parameter, if the local host requests CHAP authentication. Specifies the CHAP name expected from a remote host. The value *chap_name* can be any string, between 1 and 255 characters in length.

`chap_peer_secret` *chap_secret*

Mandatory parameter, if the local host requests CHAP authentication. Specifies the CHAP secret that is used with the challenge value to generate the response expected from the remote host. The value *chap_secret* can be any string, between 1 and 255 characters in length.

`send_authentication` *mode*

Optional parameter. Indicates whether the local host will participate in authentication negotiation requested by remote hosts, and the authentication protocol used.

The value *mode* can be `off` (no authentication), `pap` (authentication using PAP), `chap` (authentication using CHAP), or `pap|chap` (authentication using both PAP and CHAP). The default value is `off`.

`send_pap_id` *pap_id*

Mandatory parameter, if the remote host requests PAP authentication. Specifies the PAP identifier sent to a remote host when it requests authentication. The value *pap_id* can be any string, between 0 and 255 characters in length.
A zero length value is represented by: `expect_pap_id ""`

`send_pap_passwd` *pap_passwd*

Mandatory parameter, if the remote host requests PAP authentication. Specifies the PAP password sent to a remote host when it requests authentication. The value *pap_passwd* can be any string, between 0 and 255 characters in length.
A zero length value is represented by: `expect_pap_passwd ""`

`send_chap_name` *chap_name*

Mandatory parameter, if the remote host requests CHAP authentication. Specifies the CHAP name sent to a remote host when it requests authentication. The value *chap_name* can be any string, between 1 and 255 characters in length.

`chap_own_secret` *chap_secret*

Mandatory parameter, if the remote host requests CHAP authentication. Specifies the CHAP secret that is used with the challenge value to generate the response sent to the remote host. The value *chap_secret* can be any string, between 1 and 255 characters in length.

Examples

The following synchronous path definition shows that the local host will request both PAP and CHAP authentication from remote hosts, but will only participate in PAP negotiation when authentication is requested by a remote host:

```
sync_path
    ip_interface           ipdptp0
    unix_device            zsh0
    expect_authentication  pap|chap
    expect_pap_id          epic_id
    expect_pap_passwd      epic_passwd
    expect_chap_name       epic_name
    chap_peer_secret       epic_secret
    send_authentication    pap
    send_pap_id            papyrus_id
    send_pap_passwd        papyrus_passwd
```

The following synchronous path definitions show load-sharing enabled between two synchronous paths that use the same IP interface:

```
sync_path
    ip_interface    ipdptp2
    unix_device     hih0

sync_path
    ip_interface    ipdptp2
    unix_device     hih1
```

Defining Asynchronous Paths (`dialup_path`)

Asynchronous paths are identified in the file `ppp.conf` by the keyword `dialup_path`, which starts each definition. They can be associated with point-to-point and point-to-multipoint IP interfaces. Dynamic IP address allocation is supported over asynchronous paths only.

Synopsis

Asynchronous path definitions have the following general forms:

Dialup path using static point-to-point IP interface →

Dialup path using dynamic point-to-point IP interface →

Dialup path using point-to-multipoint IP interface →

```
dialup_path
    ip_interface          ipdptpn
    expect_login_id       user_name
    .
    .
dialup_path
    ip_interface          ipdptp*
    expect_login_id       user_name
    .
    .
dialup_path
    ip_interface          ipdn
    expect_login_id       user_name
    remote_ip_addr        ip_addr
```

Keywords

`dialup_path`
: Mandatory parameter for asynchronous paths. Indicates the start of an asynchronous (or dialup) path definition.

`ip_interface` *interface*
: Mandatory parameter for asynchronous paths. Associates the asynchronous path with one of the point-to-point (`ipdptp`*n*) or point-to-multipoint (`ipd`*n*) IP interfaces defined in the `ifconfig` section of the file.

Point-to-point IP interfaces may be *static* or *dynamic*. Static point-to-point IP interfaces are identified by a number (ipdptp0, ipdptp1, ..., ipdptp*n*), and associate the dialup path with exactly one pair of source and destination IP addresses. For example:

```
dialup_path
    ip_interface          ipdptp0
```

Dynamic IP interfaces are used for dynamic IP address allocation on the server side, and are identified by an asterisk (ipdptp*). An interface is assigned on demand, for as long as there are interfaces available in the pool. For example:

```
dialup_path
    ip_interface          ipdptp*
```

remote_host *name*

Mandatory parameter for asynchronous paths used to initiate calls. Associates the asynchronous path with the name of one of the remote hosts defined in the file link.conf. The value *name* can be any character string.

remote_ip_addr *ip_addr*

Mandatory parameter for point-to-multipoint connections. Not required for point-to-point connections. Specifies the IP address of the remote host associated with the asynchronous path. The value *ip_addr* can be an IP address (expressed using dot notation) or a hostname that appears in the file /etc/hosts.

expect_login_id *login*

Mandatory parameter for asynchronous paths used to accept incoming calls. Specifies the login id expected from the remote host. This parameter is used to associate an incoming call with a specific asynchronous path; therefore each remote host must have a unique login id.

The value *login* can be any lowercase string, between 1 and 8 characters in length. It must correspond to the login id which appears in the relevant connect script on the remote host.

You must also create a user account with this login id, using `admintool(1M)`. See Section , "Adding User Accounts for Incoming Connections," on page 30 for detailed instructions.

`default_route`
: Optional parameter for point-to-point IP interfaces. When the IP interface is marked *up*, the route is added to the routing table automatically as the default destination. It is removed from the routing table when the IP interface is marked *down*. This parameter is most commonly used in client configurations—that is, links configured for outgoing calls only. It should never be used in conjunction with a routing daemon running on the machine, because this generates unnecessary network traffic.

`inactivity_timeout` *seconds*
: Optional parameter for asynchronous paths. Specifies the number of seconds of inactivity that elapse before an asynchronous connection is closed automatically.

The value *seconds* can be any integer. The default value is 120 seconds (2 minutes). If the value *seconds* is set to zero, the connection remains open until closed explicitly.

`request_ip_addr` *state*
: Optional parameter for asynchronous paths. Enables dynamic IP address allocation *at the client side only*. When the value *state* is set to on, the client requests an IP address from a pool of interfaces assigned at the server side.

The value *state* can be `on` (enabled) or `off` (disabled). The default value is `off`.

`private` *ip_interface*
: Optional parameter for asynchronous paths. Hides the specified IP interface from the interface pool defined for dynamic IP address allocation on the server side. Can be used to reserve point-to-point IP interfaces so they can be used for synchronous connections.

`accept_any_ip_addr` *state*
: Optional parameter for asynchronous paths. Accepts the IP addresses provided by the remote host, even if they differ from the IP addresses assigned to the interface locally.

The value *state* can be `on` (enabled) or `off` (disabled). The default value is `off`.

`lcp_async_map` *mask*

Optional parameter for asynchronous paths. Specifies the LCP asynchronous map used by the remote host. The LCP asynchronous map is a negotiated parameter that defines which control characters are transposed for transmission in PPP frames.

Control characters in the range `0x00` to `0x1f`, such as `CTRL-S` and `CTRL-Q`, are used by some devices to implement software flow control. These devices may interpret the control characters transmitted in PPP frames, and close the link as a result. To avoid problems interoperating with these devices, all 32 control characters are automatically transposed for transmission, so that they appear outside of the significant range. Encoding and decoding the control characters incurs a processing overhead at both ends of the link.

The LCP asynchronous map defines which of the control characters is transposed by the remote host. A bit set to `1` in the value *mask* tells the remote host to transpose the corresponding control character; a bit set to zero tells the remote host to leave the control character unchanged. Provided you can predict how each device in the link will respond to the control characters it receives in PPP frames, you can tell the remote host to transpose a subset of the control characters, by specifying a different *mask* value. For example, a *mask* value of `0x0000ffff` tells the remote host to transpose the first 16 control characters only.

By default, the value *mask* is set to `0xffffffff,` which tells the remote host to transpose all 32 control characters. A value of `0x0` leaves all control characters unchanged.

`lcp_compression` *state*

Optional parameter for asynchronous paths. Indicates whether the Address and Protocol fields in the PPP frame are compressed. See Appendix A, "PPP Link Operation" for more information.

The value *state* can be `on` (enabled) or `off` (disabled). The default value is `on`.

`lcp_mru` *mru*

Optional parameter for synchronous and asynchronous paths. Specifies the maximum receive unit (MRU) for the local machine. This parameter is carried in the LCP `Configure-request` frame, and sets the maximum transmission unit (MTU) for the remote host. See Appendix A, "PPP Link Operation" for more information.

By default, the value *mru* is set to 1500 bytes for Ethernet networks.

`lcp_restart_timer` *seconds*

Optional parameter for synchronous and asynchronous paths. Specifies the number of seconds which elapse between consecutive LCP `Configure-Request` frames. Increasing the LCP restart timer may be necessary when connecting over long delay networks, such as satellite connections. See Appendix A, "PPP Link Operation" for more information.

The value *seconds* can be any integer greater than zero. The default value is 3 seconds.

`lcp_max_restart` *max_restart*

Optional parameter for synchronous and asynchronous paths. Specifies the number of unanswered LCP `Configure-Request` frames generated before the endpoint is considered unreachable and the IP interface is marked as *down*. See Appendix A, "PPP Link Operation" for more information.

The value *max_restart* can be any integer in the range 1 to 255. The default value is 10. If the value *max_restart* is set to 255, LCP `Configure-Request` frames are generated periodically until the remote host finally responds.

`ppp_link_name` *name*

Optional parameter for synchronous and asynchronous paths. Assigns a name to the link, which is used by `ppptrace` and `pppstat`. The value *name* can be any character string.

`ipcp_compression` *state*
Optional parameter for synchronous and asynchronous paths. Indicates the current state of the header compression facility, which uses Van Jacobsen compression to improve performance over slow links. See Appendix A, "PPP Link Operation" for more information.

The value *state* can be `vj` (enabled) or `off` (disabled). The default value is `vj`.

`expect_authentication` *mode*
Optional parameter for synchronous and asynchronous paths. Indicates that the local host will request authentication from remote hosts, and the authentication protocol to be used. If authentication is enabled, remote hosts must authenticate themselves successfully, or the connection is closed.

The value *mode* can be `off` (no authentication), `pap` (authentication using PAP), `chap` (authentication using CHAP), or `pap|chap` (authentication using both PAP and CHAP). The default value is `off`.

If both PAP and CHAP are enabled, CHAP authentication is performed first. If the remote host does not support CHAP authentication, it is allowed to participate in PAP authentication only.

`expect_pap_id` *pap_id*
Mandatory parameter, if the local host requests PAP authentication. Specifies the PAP identifier expected from a remote host. The value *pap_id* can be any string between 0 and 255 characters in length. A zero length value is represented by: `expect_pap_id ""`

`expect_pap_passwd` *pap_passwd*
Mandatory parameter, if the local host requests PAP authentication. Specifies the PAP password expected from a remote host. The value *pap_passwd* can be any string, between 0 and 255 characters in length. A zero length value is represented by: `expect_pap_passwd ""`

`expect_chap_name` *chap_name*
Mandatory parameter, if the local host requests CHAP authentication. Specifies the CHAP name expected from a remote host. The value *chap_name* can be any string, between 1 and 255 characters in length.

`chap_peer_secret` *chap_secret*

Mandatory parameter, if the local host requests CHAP authentication. Specifies the CHAP secret that is used with the challenge value to generate the response expected from the remote host. The value *chap_secret* can be any string, between 1 and 255 characters in length.

`send_authentication` *mode*

Optional parameter. Indicates whether the local host will participate in authentication negotiation requested by remote hosts, and the authentication protocol used.

The value *mode* can be `off` (no authentication), `pap` (authentication using PAP), `chap` (authentication using CHAP), or `pap|chap` (authentication using both PAP and CHAP). The default value is `off`.

`send_pap_id` *pap_id*

Mandatory parameter, if the remote host requests PAP authentication. Specifies the PAP identifier sent to a remote host when it requests authentication. The value *pap_id* can be any string, between 0 and 255 characters in length.
A zero length value is represented by: `expect_pap_id ""`

`send_pap_passwd` *pap_passwd*

Mandatory parameter, if the remote host requests PAP authentication. Specifies the PAP password sent to a remote host when it requests authentication. The value *pap_passwd* can be any string, between 0 and 255 characters in length.
A zero length value is represented by: `expect_pap_passwd ""`

`send_chap_name` *chap_name*

Mandatory parameter, if the remote host requests CHAP authentication. Specifies the CHAP name sent to a remote host when it requests authentication. The value *chap_name* can be any string, between 1 and 255 characters in length.

`chap_own_secret` *chap_secret*

Mandatory parameter, if the remote host requests CHAP authentication. Specifies the CHAP secret that is used with the challenge value to generate the response sent to the remote host. The value *chap_secret* can be any string, between 1 and 255 characters in length.

Examples

The following asynchronous path definition shows a point-to-multipoint IP interface, and that the local host will request CHAP authentication from the remote host `odyssey`:

```
dialup_path
    ip_interface           ipd0
    expect_login_id        odyssey-login
    remote_host            odyssey
    remote_ip_addr         129.xxx.xxx.119
    inactivity_timeout     120
    expect_authentication  chap
    expect_chap_name       odyssey_name
    chap_peer_secret       odyssey_secret
```

The following asynchronous path definition shows dynamic IP address allocation enabled at the client side:

```
dialup_path
    ip_interface           ipdptp0
    remote_host            odyssey
    request_ip_addr        on
```

The following asynchronous path definitions show dynamic IP interfaces assigned to three dialup paths:

```
dialup_path
    ip_interface           ipdptp*
    expect_login_id        remote1

dialup_path
    ip_interface           ipdptp*
    expect_login_id        remote2

dialup_path
    ip_interface           ipdptp*
    expect_login_id        remote3
```

Assigning Path Defaults

The keyword `defaults` is used to define a list of default parameters that are applied to all subsequent synchronous and asynchronous path definitions. Any *optional* parameter may appear in the list of defaults. Mandatory parameters such as `ip_interface` or `unix_device`, or parameters used to create associations between files, such as `remote_host` or `expect_login_id`, *must not* be used as defaults.

Take care when combining defaults for both synchronous and asynchronous paths. In particular, do not attempt to enable dynamic IP addressing for synchronous paths. To avoid errors, it is better to define separate defaults for each type of path.

For example, the following path definitions show defaults set independently for both synchronous and asynchronous paths:

```
defaults
    link_monitor            on
    lcp_mru                 4352

sync_path
    ip_interface            ipdptp0
    unix_device             hih0

sync_path
    ip_interface            ipdptp1
    unix_device             hih1

defaults
    inactivity_timeout      180
    request_ip_addr         on

dialup_path
    ip_interface            ipdptp2
    remote_host             server0

dialup_path
    ip_interface            ipdptp3
    remote_host             server1
```

Editing the Link Configuration File (link.conf)

The link configuration file (/etc/opt/SUNWconn/ppp/link.conf) describes the synchronous and asynchronous devices used to establish the PPP link, and includes dialing information for asynchronous links.

The first section in the link configuration file is used to define the location of the modem configuration file and the directory that contains the connect (or CHAT) scripts. By default, the modem configuration file is /etc/opt/SUNWconn/ppp/modems and the connect (or CHAT) scripts are contained in the directory /etc/opt/SUNWconn/ppp/script.

Defining Synchronous Devices

Synchronous devices are identified in the file link.conf by the keyword sync_device, which starts each definition.

Synopsis

Synchronous device definitions have the following general form:

```
sync_device  syncdevn
    unix_device             device_name
    line_speed              line_speed
    tx_clock                txc
    rx_clock                rxc
```

Keywords

sync_device syncdev*n*

Mandatory parameter for synchronous devices. Indicates the start of a synchronous device definition, and assigns a name to the device.

unix_device *device_name*

Mandatory parameter for synchronous devices. Identifies the serial port used for synchronous communication, and associates the synchronous device with one of the synchronous paths defined in the file ppp.conf. The value *device_name* must be the device name of a synchronous serial interface installed in your machine.

For example, device names of the form `zsh`*n* associate the path with one of the on-board serial interfaces. The device names of the form `hih`*n*, associate the path with a high-speed serial interface (HSI).

`line_speed` *line_speed*
: Mandatory parameter for synchronous devices when internal clocking is selected. The optimum line speed is dependent on the serial device selected. The default value proposed is the optimum line speed for one of the onboard serial interfaces (`zsh`). Refer to the manufacturer's documentation, if you are using a different serial device.

`tx_clock` *txc*
: Mandatory parameter for synchronous devices. Specifies the source of the transmit clock signal. The value *txc* can be `baud` (internal clocking using system baud rate), `txc` (external clocking using the signal on the transmit pin), or `rxc` (external clocking using the signal on the receive pin). The transmit clock is most commonly set to `baud` (internal clocking), or `txc` (external clocking).

`rx_clock` *rxc*
: Mandatory parameter for synchronous devices. Specifies the source of the receive clock signal. The value *rxc* can be `baud` (internal clocking using system baud rate), `txc` (external clocking using the signal on the transmit pin), or `rxc` (external clocking using the signal on the receive pin). The receive clock is most commonly set to `rxc`, for both internal and external clocking.

Examples

The following synchronous device definition uses a high-speed serial interface (`hih`) and external clocking:

```
sync_device                 syncdev0
    unix_device             hih0
    line_speed              0
    tx_clock                txc
    rx_clock                rxc
```

The following synchronous device definition uses one of the onboard serial interfaces (`zsh`) and internal clocking:

```
sync_device                 syncdev0
    unix_device             zsh0
    line_speed              19200
    tx_clock                baud
    rx_clock                rxc
```

Defining Asynchronous Devices

Asynchronous devices are identified in the file `link.conf` by the keyword `dialup_device`, which starts each definition.

Synopsis

Asynchronous device definitions have the following general form:

```
dialup_device               pppdevn
    unix_device             device_name
    line_speed              speed
    modem                   modem_id
    calls                   call_type
```

Keywords

`dialup_device pppdev`*n*

Mandatory parameter for asynchronous devices. Indicates the start of an asynchronous device definition, and assigns a name to the device.

`unix_device` *device_name*

Mandatory parameter for asynchronous devices. Specifies the serial port used to connect between the host and the modem.

`line_speed` *speed* `[ctsrts]`

Mandatory parameter for asynchronous devices. Specifies the line speed for the connection between the host and the modem. For optimum performance, the line speed should be equal to, or greater than, the baud rate of the modem. The option `ctsrts` enables software flow control using Clear to Send and Ready to Send signals.

`modem` *modem_id*

Mandatory parameter for asynchronous devices. Specifies the type of modem connected to the serial port, and associates the asynchronous device with one of the definitions in the modem database, which is the file `/etc/opt/SUNWconn/ppp/modems` by default. This parameter is set to `none` for a null modem configuration.

`call_setup` *call_type*

Mandatory parameter for asynchronous devices. Specifies the behavior of the device during the call setup phase. The value *call_type* can be `answer` (device accepts calls only), `dial` (device makes initiates calls only), or `both` (device accepts and initiates calls).

Examples

The following asynchronous device definitions use a BocaModem V.34 DataFax modems, and a line speed of 38400. The first device accepts and initiates calls; the second device initiates calls only:

```
dialup_device pppdev0
    unix_device            ttya
    line_speed             38400
    modem                  BocaModem V.34 DataFax
    call_setup             both

dialup_device  pppdev1
    unix_device            ttyb
    line_speed             38400
    modem                  BocaModem V.34 DataFax
    call_setup             dial
```

The following asynchronous device definition accepts calls in a null modem configuration:

```
dialup_device pppdev0
    unix_device            ttya
    line_speed             38400
    modem                  none
    call_setup             answer
```

Defining a Pool of Asynchronous Devices

If you create more than one asynchronous device definition in the file `link.conf`, you can define a *pool* of asynchronous devices.

A pool of devices is associated with one or more remote hosts, in exactly the same way as an individual device. When an incoming or outgoing connection request for one of these hosts is processed, an asynchronous device is selected from the pool automatically, for as long as there are devices available in the pool.

Synopsis

A device pool definition has the following general form:

```
pool_device  pooln
    pppdevn pppdevn+1 ...
```

Keywords

`pool_device pooln`
: Mandatory parameter to enable device pooling. Indicates the start of a device pool definition, and assigns a name to the device.

`pppdevn pppdevn+1 ...`
: Mandatory parameters to enable device pooling. Specifies a list of asynchronous devices included in the device pool. Device names must be separated by spaces or commas.

Examples

The following example shows a device pool that consists of four asynchronous devices:

```
pool_device  pool0
    pppdev0 pppdev1 pppdev2 pppdev3
```

Defining Remote Hosts

The link configuration file (`link.conf`) also contains a list of the remote hosts to which the local host can initiate asynchronous connections. All remote host definitions must be entered *after* the synchronous and asynchronous device definitions.

Synopsis

A remote host definition has the following general form:

```
remote_host name
    phone_number          number
    chat_script           filename
    use_device            pppdevn
```

Keywords

`remote_host` *name*
: Mandatory parameter for remote host definitions. Indicates the start of a remote host definition, and associates it with one of the asynchronous paths defined in the PPP path configuration file (`ppp.conf`).

`phone_number` *number*
: Mandatory parameter for remote host definitions. Specifies the telephone number used to call the remote host. This can be an extension number if the call is within the same private branch exchange. Assign a "dummy" telephone number for null modem configurations. Telephone numbers can include both digits and characters, including special characters such as `#` and `*`. A comma (`,`) is used to insert a pause. The duration of the pause is usually 2 seconds, but is programmable on most modems. Telephone numbers that are prefixed by the letter `P` indicate that pulse dialing mode should be used.

`chat_script` *filename*
: Mandatory parameter for remote host definitions. Specifies the name of the file that contains the CHAT script, which defines the dialog used to set up the connection.

`use_device` `pppdev`*n* *or* `pool`*n*
: Optional parameter for remote hosts. Specifies a device, or pool of devices, to be used for this remote host.

Editing the Modem Database File (`modems`)

The modem database file (by default, `/etc/opt/SUNWconn/ppp/modems`) contains a description of the modems recognized by Solstice PPP, including the `AT` commands used by Solstice PPP and the `pppsetmod(1M)` command to initialize the modems for incoming and outgoing connections.

A list of the modems with which the current version of Solstice PPP has been tested is contained in the *Important Product Information for Solstice PPP*, which accompanies this product. Configuration information for these modems is contained in the modem database file `/etc/opt/SUNWconn/ppp/modems`.

The `pppsetmod(1M)` command scans the contents of the modem configuration file and sends the appropriate AT command to initialize the modems for a server configuration—that is, to wait for incoming calls. Modems used in a client configuration are initialized by the link manager each time a call is initiated.

By default, the modem is reset to a default mode each time it is switched off and back on again. You can also save the configuration to the modem, so that it is reinitialized in server mode when it is switched back on.

To initialize the modem, without saving the configuration, type:

```
prompt# /usr/bin/pppsetmod
```

To initialize a modem, and save the configuration, type:

```
prompt# /usr/bin/pppsetmod -s
```

You can modify the modems database file to add configuration information for your own modems, if they are not supported already. Refer to the documentation that you received from the manufacturer for details of the AT commands used to configure your modem.

For *outgoing* (client) connections, you must enable:

- hardware flow control; and
- verbose mode.

For *incoming* (server) connections, you must enable:

- hardware flow control;
- verbose mode; and
- standard management for device carrier detect (DCD).

To add a new modem definition to the modem configuration file:

1. Edit the standard modem list to add a pointer to the command definition for your modem.

Modem name with pointer to command definition

```
[modems]
BocaModem V.34 DataFax = boca
AT&T DataPort Express  = att
Standard hayes = hayes
Cardinal V.34/V.FC 28.8 data/fax = cardinal
SupraFaxModem 288 = supra
Hayes Accura 144B = accura
Hayes Accura 288V.FC = accura
Practical 14400 V32bis = practical
USRobotics Sporter 14400 = usr
USRobotics Sporter 288 = usr_v34
My Third-party Modem = mymodem
other=hayes
```

2. Add a command definition for your modem, including an `init` command (for outgoing connections), a `reset` command, and a `server` command (for incoming connections).

Sent by link manager each time call is initiated

Sent by `pppsetmod`

```
[mymodem]
init   = AT <command string  to configure for  outgoing calls>
reset  = AT Z
server = AT <command string  to configure to wait  for incoming calls>
```

Note that some *short buffer* modems, such as the USRobotics Sporter 288, accept a restricted number of characters in each AT command. The commands can be concatenated as follows:

```
server = AT <command1> AT <command1> AT <command1> AT <command1> ...
```

Editing the CHAT (or Connect) Scripts

The CHAT (or connect) scripts define the exchange of information between the endpoints, which occurs during the link establishment phase. By default, the CHAT scripts are located in the directory `/etc/opt/SUNWconn/ppp/script`. You can change the location of the scripts by editing the location definition in the file `link.conf`, and moving the files.

You must create one CHAT script for each remote host to which the local host will initiate calls. It must contain a unique login id, which must correspond to a user account on the remote machine. If the remote host is running Solstice PPP, the login id must also appear as the value of the parameter `expect_login_id` associated with one of the dialup paths in the file `ppp.conf` on this machine.

CHAT scripts may be non-interactive (all the information to be exchanged is contained in the script) or interactive (the script prompts for user input).

The following example shows a simple CHAT script used to log in to remote hosts running a Solaris operating system. It is based on the template CHAT script delivered with Solatice PPP.

Start dialog
Response expected from remote host
Login id sent to remote host
Response expected from remote host
Login password sent to remote host

```
#
# Chat script for ppp login
#

#
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setline cs7 parodd

send    RETURN
expect  "ogin:"  10  onerror send BREAK repeat 3

send    "ppp1"
expect  "word: " 40
#
# Set the ppp password of the remote host here
#
send    "okppp1"
```

Changing the Login Id and Password

The login id and login password are used for the login sequence at the remote host. This sequence and the restrictions it imposes on the login dialog is operating system dependent.

If the remote host is running a Solaris environment, the login id must be all lowercase, and between 1 and 8 characters in length. The login password can contain both uppercase and lowercase characters, and must be equal to, or greater than, 6 characters in length; however, only the first 6 characters are significant.

Note – To accept the call, there must be a user account with this login id and password on the remote host. If the remote host is running Solstice PPP the login id must also appear as the value of the `expect_login_id` parameter for one of the dialup paths in the file `ppp.conf`.

Edit the default CHAT script to replace the login id and the login password with the strings expected by the remote host. For example, the following CHAT script is used to establish a link with a login id `mylogin` and login password `mypasswd`.

```
#
# Chat script for ppp login
#

#
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setline cs7 parodd

send    RETURN
expect  "ogin:"  10  onerror send BREAK repeat 3

send    "mylogin"
expect  "word: " 40
#
# Set the ppp password of the remote host here
#
send    "mypasswd"
```

Login id sent to to remote host

Login password sent to remote host

Changing the Number of Call Initiation Attempts

When the call is first initiated, the local host waits for a response from the remote host to begin the login sequence. The length of time the local host waits and the number of times the local host attempts to initiate the call are defined by the following entry:

```
expect  "ogin:"  10  onerror send BREAK repeat 3
```

The first figure (10) defines the wait period, and the second figure (3) defines the number of call initiation attempts. You can modify both of these parameters.

For example, to retry the call initiation once every 5 seconds for a total of 10 attempts, change the line in the file to:

```
expect  "ogin:"  5  onerror send BREAK repeat 10
```

Changing the Login Dialog

The default CHAT script assumes that a link is to be established between two Solaris systems. If you are connecting to a machine running operating system you may need to change the expected login dialog.

For example, to connect to an OpenVMS system running PPP, modify the connect script as follows:

Start dialog

Response expected from VMS host

Response expected from OpenVMS host

```
send    RETURN
expect  "name:"  10  onerror send BREAK repeat 3

send    "ppp"
expect  "word:" 40
#
# Set the ppp password of the remote host here
#
send    "xxxxx"
```

The template CHAT script supports a simple method of logging in to a remote host. You can also modify and expand the CHAT script to create a more complex login dialog. Note that to initiate connections to a remote host running Solstice PPP, you must include a login id that is recognized by the remote host, and provide a way to start the Solstice PPP login service.

For example, the following CHAT script could be used to log in to a remote host running a Solaris environment, invoke `mailx(1)` to send an email message to indicate that the operation was successful, and start the Solstice PPP login service. Note that, in this case, the user account on the remote host that corresponds to the login id sent by the script must use `/usr/bin/csh` as the default shell.

```
#
# Chat script for ppp login
#

#
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setline cs7 parodd

send    RETURN
expect  "ogin:"  10  onerror send BREAK repeat 3

send    "mylogin"
expect  "word: " 40
#
# Set the ppp password of the remote host here
#
send    "mypasswd"

expect  "%"

send  "echo Login to remote successful | /usr/bin/mailx alias"

expect  "%"

send    "/usr/sbin/pppls"
```

Login id sent to to remote host

Login password sent to remote host

C-shell prompt returned by remote host

Command sent to invoke `mailx`

C-shell prompt returned by remote host

Start PPP login service (`pppls`)

The following CHAT script could be used to log in to a telnet terminal server:

```
#
# Chat script for ppp login
#

#
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setline cs7 parodd

send    RETURN
expect  "sername:" 10 onerror send RETURN repeat 5

send    "<your username>"
expect  "assword:" 35

send    "<your password>"
expect  "pc>" 10 onerror send RETURN repeat 2

send    "terminal flowcontrol hardware"
expect  "pc>" 10 onerror send RETURN repeat 2

send    "terminal databits 8"
expect  "pc>" 10 onerror send RETURN repeat 2

send    "terminal parity none"
expect  "pc>" 10 onerror send RETURN repeat 2

send    "terminal escape-character 0"
expect  "pc>" 10 onerror send RETURN repeat 2

send    "terminal telnet-transparent"
expect  "pc>" 10 onerror send RETURN repeat 2

send    "telnet <your own machine>"
expect  "ogin:" onerror send RETURN repeat 5

send    "<your login for PPP>"
expect  "assword:" 35

send    "<your password  for PPP>"
```

Interactive CHAT Scripts

Interactive CHAT scripts use `echo` and `read` keywords to display prompts and to acquire user input. The user input is stored as variables, which are identified by the `$` prefix. For example, an interactive version of the previous script could be:

Prompt for input from user
Read input from user
Prompt for input from user
Read input from user

```
send   RETURN
expect  "ogin:"  10  onerror send BREAK repeat 3

echo   "Enter your PPP login id: "
read   $login
send   "$login"

expect  "word: " 40

echo   "Enter your PPP password: "
read   $password
send   "$password"
```

A more complex example shows a CHAT script used to manage the interaction between the user and a dynamic challenge-response authentication system:

Prompt for input from user
Read input from user
Wait for response from server
Prompt for input from user
Wait for response from server

```
send   RETURN
expect "ID:"  10  onerror send BREAK repeat 3

echo   "Enter your user ID number: "
read   $id
send   "$id"

expect "Challenge: ${challenge,6}" 10
echo   "Enter the response for Challenge ${challenge}: "
read   $response
send   "$response"

expect "${host}:" 20
echo   "Connected to ${host}\n"
send   "ppp"
```

In this example, the script reads a user id and sends it to the server. It waits for 10 seconds for a response from the server that starts with the string `Challenge:` and then reads the next six characters which represent the challenge value.

The script then displays the challenge value, waits for the user to enter the corresponding response, and sends this to the server. If the response is accepted, the connection is completed.

The `{}` brackets are used to delimit a variable when it appears with other characters as part of a character string.

Example Configurations 5

This chapter provides examples of common network configurations created using Solstice PPP, including the corresponding configuration files.

To configure Solstice PPP on your machine:

1. Create the files `ppp.conf` and `link.conf`, which specify the configuration of the synchronous and asynchronous PPP links on your machine.

2. To initiate asynchronous connections, create a CHAT script that specifies the login id and password sent during the link establishment phase.

3. To accept asynchronous connections, create a user account that specifies the login id and login password expected during the link establishment phase.

Synchronous LAN to LAN Configuration

Figure 5-1 shows a single synchronous PPP link used to connect two local area networks (LANs). Two hosts (`epic` and `odyssey`) running Solstice PPP act as routing gateways between the two networks. Both hosts request authentication using the Password Authentication Protocol (PAP).

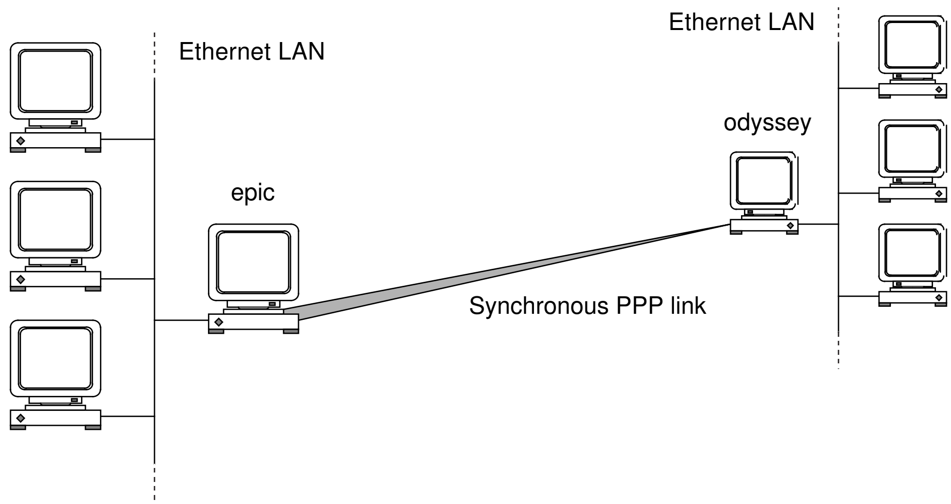


Figure 5-1 Synchronous LAN to LAN Configuration

The simplest, and most efficient, way to implement this configuration establishes a point-to-point IP connection between the gateways `epic` and `odyssey`. Each host can use its primary IP address as the source address of the point-to-point IP interface.

The two hosts act as IP routers in this configuration; therefore, the file `/etc/gateways` must exist on each host so that IP datagrams are routed correctly.

In this example, both hosts use one of the onboard serial ports (`zsh`) to make the link; however, the same link could be established using some other serial device.

PPP Configuration File (`ppp.conf`) for `epic`:

Create point-to-point interface
Use on-board serial port (zsh) driver
Request authentication
Respond to authentication request

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 epic odyssey netmask 255.255.255.0 mtu 1500 up

sync_path
    ip_interface              ipdptp0
    unix_device               zsh0
    expect_authentication     pap
    expect_pap_id             odyssey_id
    expect_pap_passwd         odyssey_passwd
    send_authentication       pap
    send_pap_id               epic_id
    send_pap_passwd           epic_passwd
```

PPP Configuration File (`ppp.conf`) for `odyssey`:

Create point-to-point interface
Exchange PAP parameters

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 odyssey epic netmask 255.255.255.0 mtu 1500 up

sync_path
    ip_interface              ipdptp0
    unix_device               zsh0
    expect_authentication     pap
    expect_pap_id             epic_id
    expect_pap_passwd         epic_passwd
    send_authentication       pap
    send_pap_id               odyssey_id
    send_pap_passwd           odyssey_passwd
```

Link Configuration Files (`link.conf`) for `epic` and `odyssey`:

Use on-board serial port (`zsh`) driver
Use internal clock

```
sync_device syncdev0
    unix_device               zsh0
    line_speed                19200
    tx_clock                  baud
    rx_clock                  rxc
```

Load-Sharing over Synchronous Links

Figure 5-2 shows two serial connections used to implement a single synchronous PPP link between two local area networks (LANs). Two hosts (`epic` and `odyssey`) running Solstice PPP act as routing gateways between the two networks. Neither host requests peer authentication.

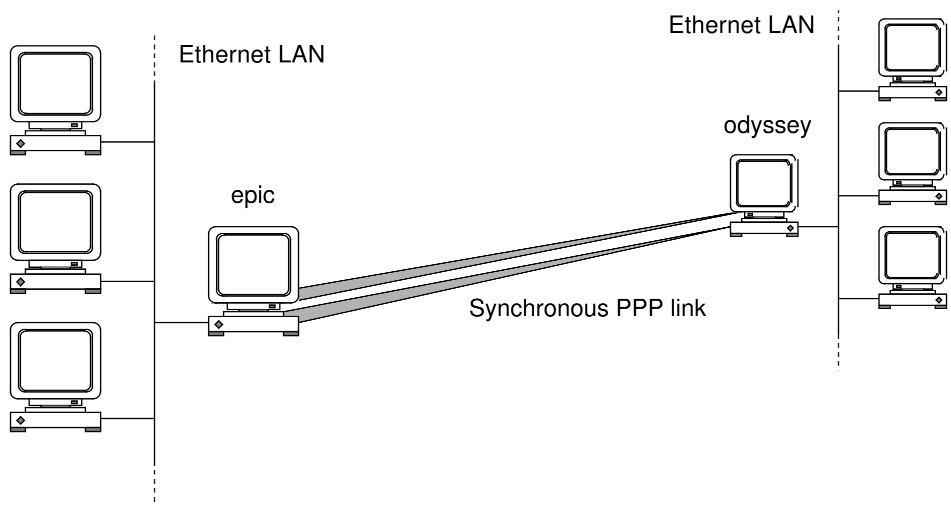


Figure 5-2 Load-sharing over Synchronous Links

This example is similar to the basic synchronous link described on page 80; however, both on-board serial ports are used in a load-sharing configuration to double the available bandwidth.

Load-sharing is a Sun-specific enhancement to the standard Point-to-Point Protocol (PPP). Both hosts must be running Solstice PPP, and, for optimum performance, both serial devices must be operating with the same line speed.

The two hosts act as IP routers in this configuration; therefore, the file `/etc/gateways` must exist on each host so that IP datagrams are routed correctly.

PPP Configuration File (`ppp.conf`) for `epic`:

Create point-to-point interface

Two synchronous devices share the same IP interface

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 epic odyssey netmask 255.255.255.0 mtu 1500 up

sync_path
    ip_interface            ipdptp0
    unix_device             zsh0

sync_path
    ip_interface            ipdptp0
    unix_device             zsh1
```

PPP Configuration File (`ppp.conf`) for `odyssey`:

Create point-to-point interface

Two synchronous devices share the same IP interface

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 odyssey epic netmask 255.255.255.0 mtu 1500 up

sync_path
    ip_interface            ipdptp0
    unix_device             zsh0

sync_path
    ip_interface            ipdptp0
    unix_device             zsh1
```

Link Configuration Files (`link.conf`) for `epic` and `odyssey`:

Two synchronous devices use the same line speed for load-sharing

```
sync_device  syncdev0
    unix_device             zsh0
    line_speed              19200
    tx_clock                baud
    rx_clock                rxc

sync_device  syncdev1
    unix_device             zsh1
    line_speed              19200
    tx_clock                baud
    rx_clock                rxc
```

Virtual Subnetwork Configuration

Figure 5-3 shows asynchronous links used to create a virtual subnetwork between four hosts. Each host can establish one link at a time with any other host in the network.

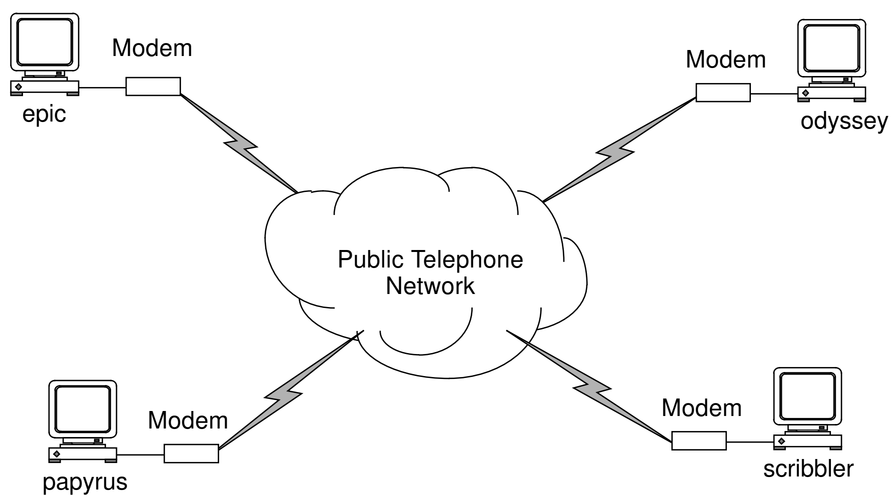


Figure 5-3 Virtual Subnetwork Configuration

Point-to-multipoint IP interfaces are used to create a virtual subnetwork over asynchronous links. A unique IP address must be assigned to each interface.

The PPP configuration file (`ppp.conf`) on each machine contains a list of dialup path definitions, which are associated with a single point-to-multipoint IP interface. The link configuration file (`link.conf`) contains corresponding definitions for each of the remote hosts in the network, which include the phone numbers and the names of the CHAT (or connect) scripts used to initiate connections.

To initiate connections, each host uses CHAT (or connect) scripts that specify a unique login id and login password sent during the link establishment phase.

To accept connections, each host must have a corresponding user account that specifies the unique login id and login password expected during the link establishment phase.

PPP Configuration File (`ppp.conf`) for `epic`:

Create point-to-multipoint interface

```
ifconfig ipd0 plumb
ifconfig ipd0 epic-ppp netmask 255.255.255.0 mtu 1500 up
```

Need to create user account with login id `ppp1`

```
dialup_path
    ip_interface          ipd0
    remote_host           odyssey
    remote_ip_addr        odyssey-ppp
    expect_login_id       ppp1
```

Need to create user account with login id `ppp2`

```
dialup_path
    ip_interface          ipd0
    remote_host           papyrus
    remote_ip_addr        papyrus-ppp
    expect_login_id       ppp2
```

Need to create user account with login id `ppp3`

```
dialup_path
    ip_interface          ipd0
    remote_host           scribbler
    remote_ip_addr        scribbler-ppp
    expect_login_id       ppp3
```

Link Configuration File (`link.conf`) for `epic`:

Device can initiate and accept connections

```
dialup_device  pppdev0
    unix_device           ttya
    line_speed            38400
    modem                 BocaModem V.34 DataFax
    call_setup            both
```

Remote hosts to which `epic` can initiate connections

```
remote_host odyssey
    phone_number          1234561235
    chat_script           odyssey.script
remote_host papyrus
    phone_number          1234561236
    chat_script           papyrus.script
remote_host scribbler
    phone_number          1234561237
    chat_script           scribbler.script
```

PPP Configuration File (`ppp.conf`) for `odyssey`:

Create point-to-multipoint interface

Need to create user account with login id `ppp0`

Need to create user account with login id `ppp2`

Need to create user account with login id `ppp3`

```
ifconfig ipd0 plumb
ifconfig ipd0 odyssey-ppp netmask 255.255.255.0 mtu 1500 up

dialup_path
    ip_interface            ipd0
    remote_host             epic
    remote_ip_addr          epic-ppp
    expect_login_id         ppp0

dialup_path
    ip_interface            ipd0
    remote_host             papyrus
    remote_ip_addr          papyrus-ppp
    expect_login_id         ppp2

dialup_path
    ip_interface            ipd0
    remote_host             scribbler
    remote_ip_addr          scribbler-ppp
    expect_login_id         ppp3
```

Link Configuration File (`link.conf`) for `odyssey`:

Device can initiate and accept connections

Remote hosts to which `odyssey` can initiate connections

```
dialup_device pppdev0
    unix_device             ttya
    line_speed              38400
    modem                   Cardinal V.34/V.FC 28.8 data/fax
    call_setup              both

remote_host epic
    phone_number            1234561234
    chat_script             epic.script
remote_host papyrus
    phone_number            1234561236
    chat_script             papyrus.script
remote_host scribbler
    phone_number            1234561237
    chat_script             scribbler.script
```

CHAT script (`odyssey.script`) used by `epic` to call `odyssey`:

Login id sent by `epic` to `odyssey` →

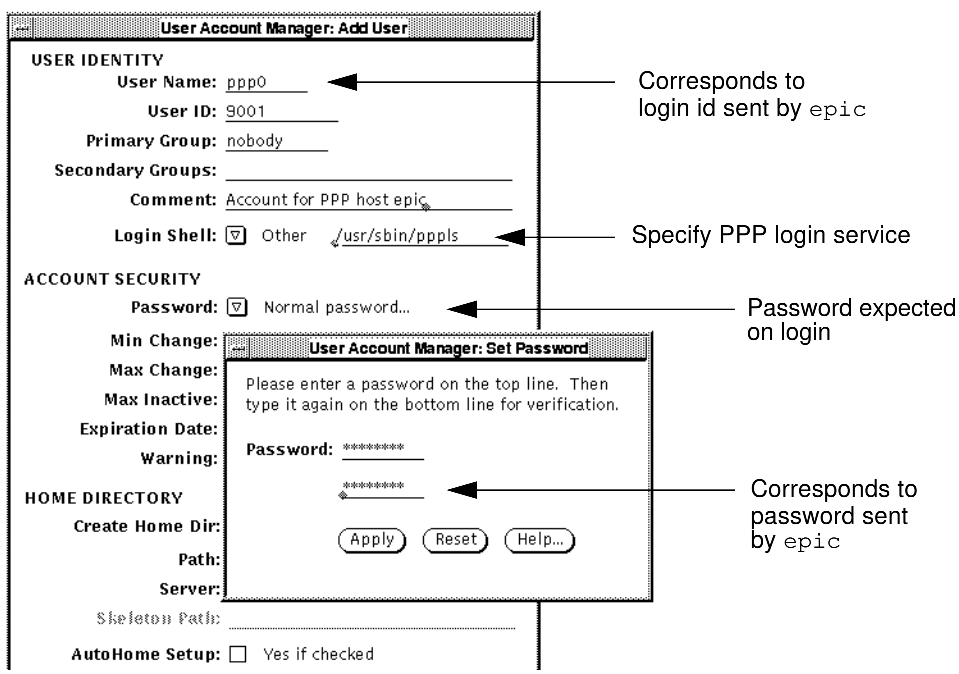
Password sent by `epic` to `odyssey` →

```
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setlinecs7 parodd

send   RETURN
expect "ogin:"  10  onerror send BREAK repeat 3

send   "ppp0"
expect "word: " 40
#
# Set the ppp password of the remote host here
#
send  "epic-pass"
```

User account on `odyssey` used to accept calls from `epic`:



CHAT script (`epic.script`) used by `odyssey` to call `epic`:

Login id sent by `odyssey` to `epic`

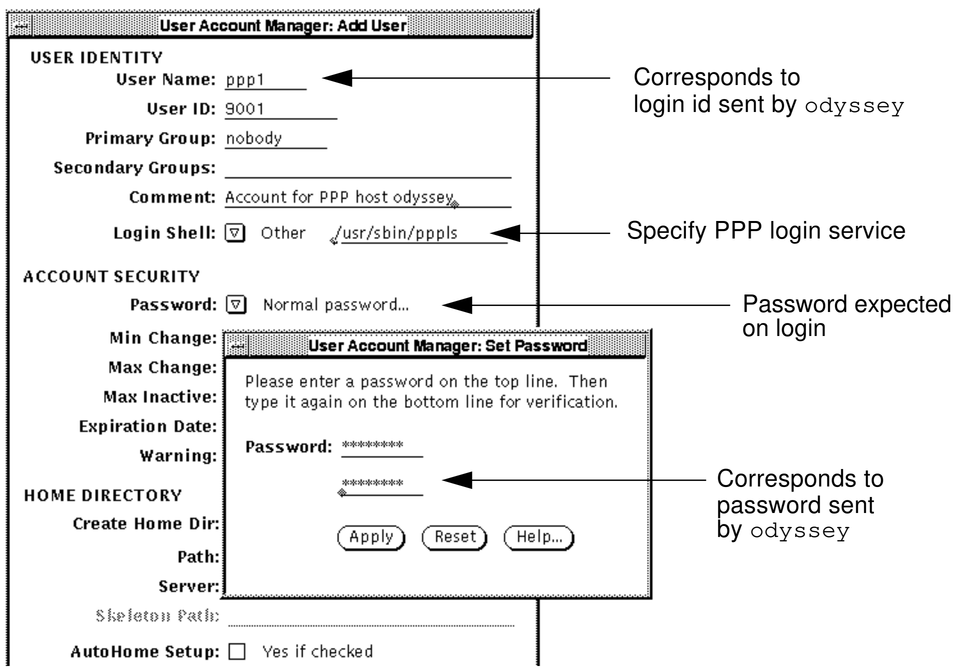
Password sent by `odyssey` to `epic`

```
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setlinecs7 parodd

send   RETURN
expect "ogin:"  10  onerror send BREAK repeat 3

send   "ppp1"
expect "word: " 40
#
# Set the ppp password of the remote host here
#
send  "odyssey-pass"
```

User account on `epic` used to accept calls from `odyssey`:



Asynchronous Client/Server Configuration

Figure 5-4 shows a small number of clients connected to a server using asynchronous links. Clients initiate calls to the server, which requests authentication using the Challenge-Handshake Authentication Protocol (CHAP).

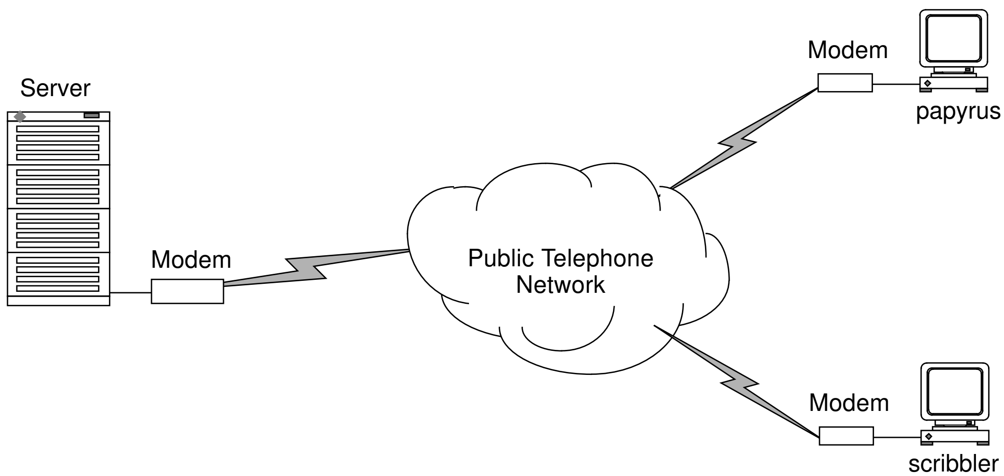


Figure 5-4 Asynchronous Client/Server Configuration

The server uses a single point-to-multipoint IP interface to create a virtual subnetwork of clients. Each client establishes a single point-to-point IP connection to the server.

To initiate calls to the server, each client uses a CHAT (or connect) script that specifies the login id and password it sends during the link establishment phase.

To accept calls from a client, the server must have a corresponding user account that specifies the login id and password it expects to receive. In this simple example, the server has only one modem and only one IP interface; therefore it can only accept one incoming call at a time.

PPP Configuration File (ppp.conf) for papyrus:

Create point-to-point interface

Respond to CHAP authentication request

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 papyrus server netmask 255.255.255.0 mtu 1500 up

dialup_path
    ip_interface            ipdptp0
    remote_host             server
    send_authentication     chap
    send_chap_name          chap-papyrus
    chap_own_secret         Secret*1
```

Link Configuration File (link.conf) for papyrus:

Initiate connections

Information used to initiate connection to server

```
dialup_device  pppdev0
    unix_device             ttya
    line_speed              38400
    modem                   BocaModem V.34 DataFax
    call_setup              dial

remote_host server
    phone_number            1234561000
    chat_script             server.script
```

CHAT script (server.script) used by papyrus to call server:

Login id sent by papyrus to server

Password sent by papyrus to server

```
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setlinecs7 parodd

send   RETURN
expect "ogin:"  10  onerror send BREAK repeat 3

send   "ppp_log1"
expect "word: " 40
#
# Set the ppp password of the remote host here
#
send  "papyrus-pass"
```

PPP Configuration File (`ppp.conf`) for `scribbler`:

Create point-to-point interface

Respond to CHAP authentication request

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 scribbler server netmask 255.255.255.0 mtu 1500
up

dialup_path
    ip_interface            ipdptp0
    remote_host             server
    send_authentication     chap
    send_chap_name          chap-scribbler
    chap_own_secret         Secret*2
```

Link Configuration File (`link.conf`) for `scribbler`:

Initiate connections

Information used to initiate connection to server

```
dialup_device  pppdev0
    unix_device             ttya
    line_speed              38400
    modem                   Practical 14400 V32bis
    call_setup              dial

remote_host server
    phone_number            1234561000
    chat_script             server.script
```

CHAT script (`server.script`) used by `scribbler` to call `server`:

Login id sent by `scribbler` to `server`

Password sent by `scribbler` to `server`

```
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setlinecs7 parodd

send   RETURN
expect "ogin:"  10  onerror send BREAK repeat 3

send   "ppp_log2"
expect "word: " 40
#
# Set the ppp password of the remote host here
#
send  "scribbler-pass"
```

PPP Configuration File (`ppp.conf`) for `server`:

Create point-to-multipoint interface

Need to create user account with login id `ppp_log1`

Request CHAP authentication

Need to create user account with login id `ppp_log2`

Request CHAP authentication

```
ifconfig ipd0 plumb
ifconfig ipd0 papyrus netmask 255.255.255.0 mtu 1500 up

dialup_path
    ip_interface            ipd0
    remote_ip_addr          papyrus-ppp
    expect_login_id         ppp_log1
    expect_authentication   chap
    expect_chap_name        chap-papyrus
    chap_peer_secret        Secret*1

dialup_path
    ip_interface            ipd0
    remote_ip_addr          scribbler-ppp
    expect_login_id         ppp_log2
    expect_authentication   chap
    expect_chap_name        chap-scribbler
    chap_peer_secret        Secret*2
```

Link Configuration File (`link.conf`) for `server`:

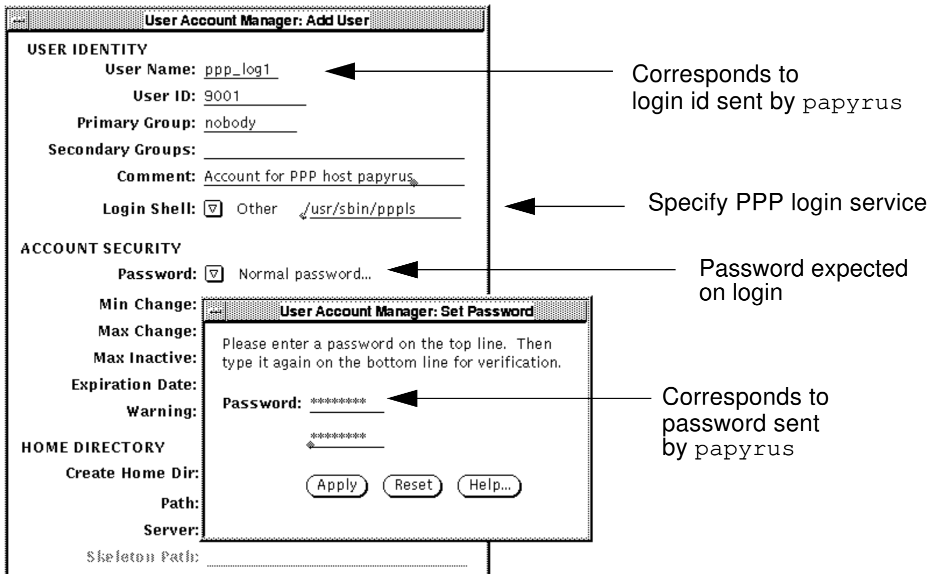
Accept connections

Accept connections

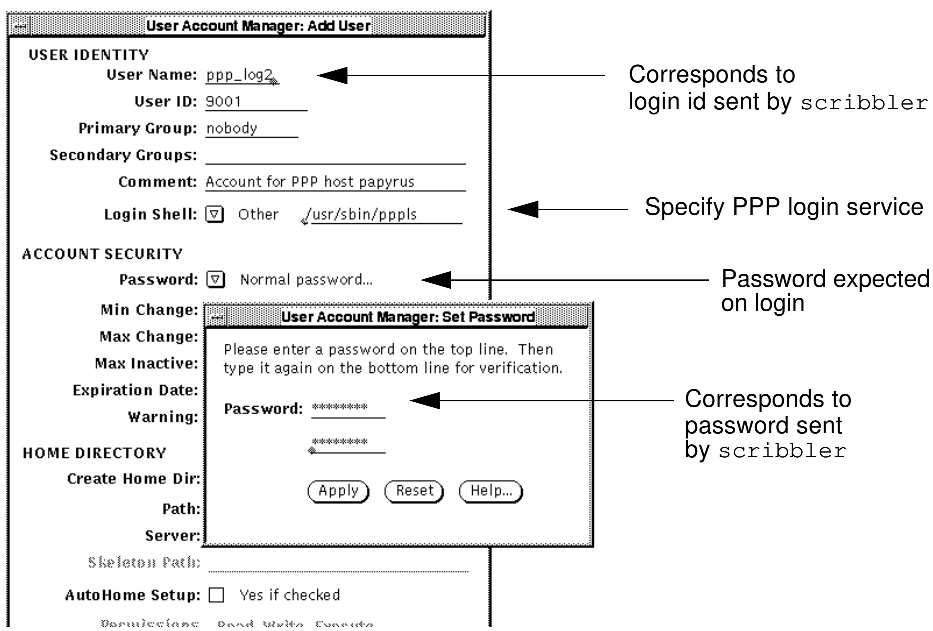
```
dialup_device  pppdev0
    unix_device             ttya
    line_speed              38400
    modem                   Cardinal V.34/V.FC 28.8 data/fax
    call_setup              answer

dialup_device  ppdev1
    unix_device             ttyb
    line_speed              38400
    modem                   Cardinal V.34/V.FC 28.8 data/fax
    call_setup              answer
```

User account on `server` used to accept calls from `papyrus`:



User account on `server` used to accept calls from `scribbler`:



Generic Internet Server Configuration

Figure 5-5 shows a large number of clients connected to a server using asynchronous links. Clients initiate calls to the server, and request an IP address for the connection. The server has a pool of IP addresses, and a pool of modems, which it allocates to clients as required. In this example, the server requires authentication using the Password Authentication Protocol (PAP).

This configuration is typical of connections between clients and a generic Internet server.

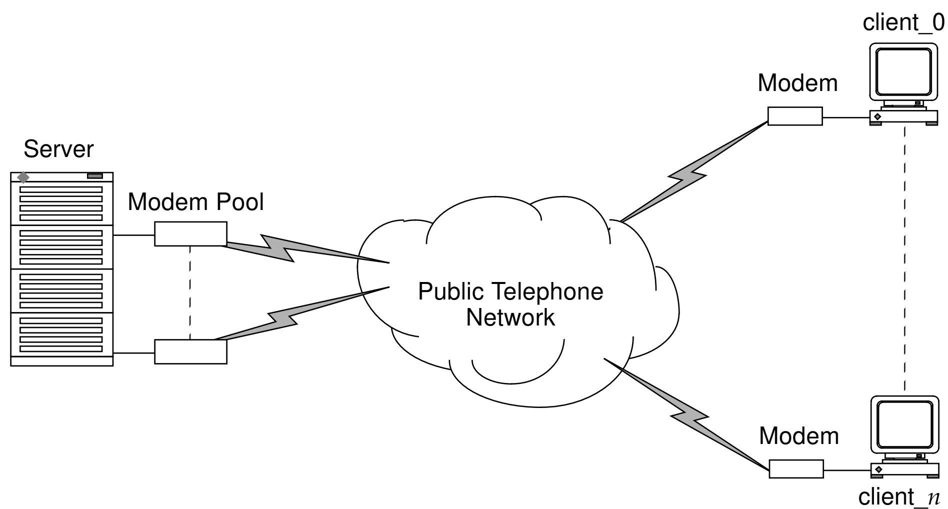


Figure 5-5 Dynamic IP Address Allocation

The server has a pool of IP addresses equal to the number of modems it has connected to it. It assigns these addresses to remote clients on request, for as long as there are modems available. Clients use `pppconn(1M)` to establish a PPP link to the server, and to recover an IP address for the IP connection.

```
prompt# /usr/bin/pppconn server
```

In the following example, a total of *n* clients can make connections to a server with a pool of eight modems; therefore, eight clients can be connected simultaneously.

PPP Configuration File (`ppp.conf`) for `client_0`:

Create point-to-point interface without IP address

Request IP addresses from server

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 netmask 255.255.255.0 down

dialup_path
    ip_interface            ipdptp0
    request_ip_addr         on
    remote_host             server
    send_authentication     pap
    send_pap_id             pap_id0
    send_pap_passwd         pap_passwd0
```

Link Configuration File (`link.conf`) for `client_0`:

Initiate connections

Information used to initiate connection to server

```
dialup_device  pppdev0
    unix_device             ttya
    line_speed              38400
    modem                   BocaModem V.34 DataFax
    call_setup              dial

remote_host server
    phone                   1234561000
    chat_script             server.script
```

CHAT script (`server.script`) used by `client_0` to call `server`:

Login id sent by client_0 to `server`

Password sent by client_0 to `server`

```
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setlinecs7 parodd

send   RETURN
expect "ogin:"  10  onerror send BREAK repeat 3

send   "clnt_0"
expect "word: " 40
#
# Set the ppp password of the remote host here
#
send  "clnt_0-pass"
```

PPP Configuration File (`ppp.conf`) for `server`:

Create pool of eight point-to-point interfaces

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 local remote0 netmask 255.255.255.0 down

ifconfig ipdptp1 plumb
ifconfig ipdptp1 local remote1 netmask 255.255.255.0 down

ifconfig ipdptp2 plumb
ifconfig ipdptp2 local remote2 netmask 255.255.255.0 down
    .
    .
ifconfig ipdptp8 plumb
ifconfig ipdptp8 local remote8 netmask 255.255.255.0 down
```

Create pool of *n* dialup path definitions, using dynamically allocated IP interfaces

```
dialup_path
    ip_interface            ipdptp*
    expect_login_id         clnt_0
    expect_authentication   pap
    expect_pap_id           pap-id0
    expect_pap_passwd       pap_passwd0

dialup_path
    ip_interface            ipdptp*
    expect_login_id         clnt_1
    expect_authentication   pap
    expect_pap_id           pap-id1
    expect_pap_passwd       pap_passwd1

dialup_path
    ip_interface            ipdptp*
    expect_login_id         clnt_2
    expect_authentication   pap
    expect_pap_id           pap-id2
    expect_pap_passwd       pap_passwd2
    .
    .
dialup_path
    ip_interface            ipdptp*
    expect_login_id         clnt_n
    expect_authentication   pap
    expect_pap_id           pap-idn
    expect_pap_passwd       pap_passwdn
```

Link Configuration File (`link.conf`) for `server`:

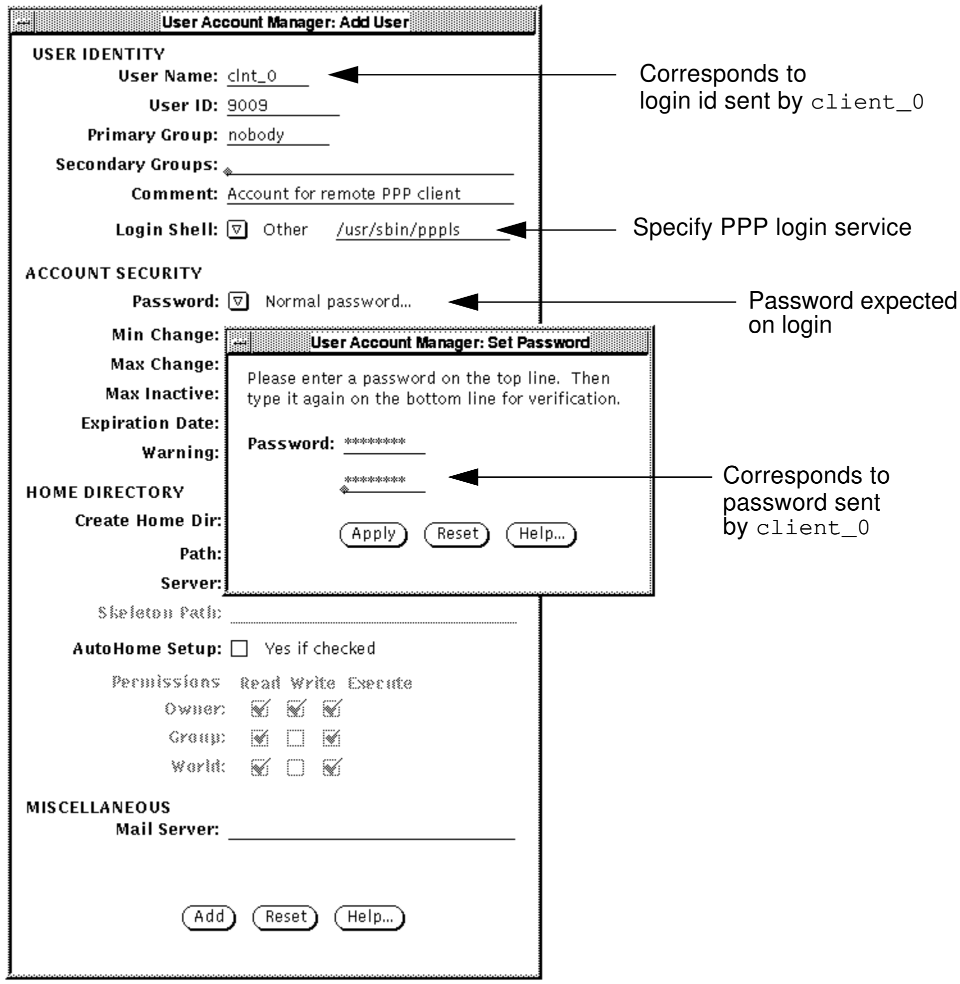
Create pool of eight asynchronous devices

```
dialup_device               pppdev0
    unix_device             tty0
    line_speed              38400
    modem                   Cardinal V.34/V.FC 28.8 data/fax
    call_setup              answer

dialup_device               pppdev1
    unix_device             tty1
    line_speed              38400
    modem                   Cardinal V.34/V.FC 28.8 data/fax
    call_setup              answer

dialup_device               pppdev2
    unix_device             tty2
    line_speed              38400
    modem                   Cardinal V.34/V.FC 28.8 data/fax
    call_setup              answer
    .
    .
    .
    .
dialup_device               pppdev10
    unix_device             tty8
    line_speed              38400
    modem                   Cardinal V.34/V.FC 28.8 data/fax
    call_setup              answer
```

User account on `server` used to accept calls from `client_0`:



Null Modem Configuration

The null modem configuration is used to connect two hosts together directly. There are no modems required. See Appendix B, "Modem and Null Modem Cables" for a description of the null modem cables that can be used for this configuration.

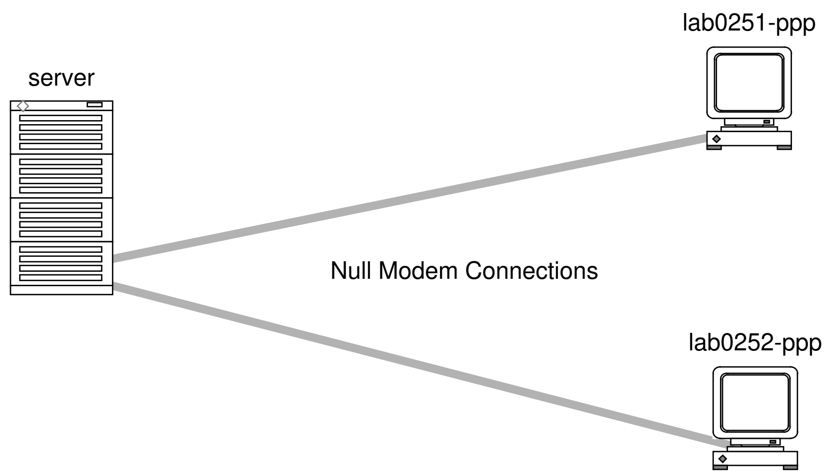


Figure 5-6 Null Modem Configuration

In this example, the server has two asynchronous null modem connections to the clients `lab0251-ppp` and `lab0252-ppp`. The `modem` keyword in the `link.conf` file is always set to `none`, and a dummy telephone number is assigned to the server.

Note – There is a limitation in this revision of the product that means that you must always specify a telephone number to initiate an asynchronous connection, even if it is never actually used.

PPP Configuration File (`ppp.conf`) for `lab0251-ppp`:

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 lab0251-ppp server netmask 255.255.255.0 down

dialup_path
    ip_interface           ipdptp0
    remote_host            server
    inactivity_timeout     120
```

Link Configuration File (`link.conf`) for `lab0251-ppp`:

Modem set to none

Initiate connections

Dummy telephone number

```
dialup_device  pppdev0
    unix_device            ttya
    line_speed             38400
    modem                  none
    call_setup             dial

remote_host server
    phone                  1001
    chat_script            server.script
```

CHAT script (`server.script`) used by `lab0251-ppp` to call `server`:

Login id sent by lab0251-ppp to `server`

Password sent by lab0251-ppp to `server`

```
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setlinecs7 parodd

send   RETURN
expect "ogin:"  10  onerror send BREAK repeat 3

send   "lab0251"
expect "word: " 40
#
# Set the ppp password of the remote host here
#
send  "lab0251-pass"
```

PPP Configuration File (`ppp.conf`) for `lab0252-ppp`:

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 lab0252-ppp server netmask 255.255.255.0 down

dialup_path
    ip_interface            ipdptp0
    remote_host             server
    inactivity_timeout      120
```

Link Configuration File (`link.conf`) for `lab0252-ppp`:

Modem set to none

Initiate connections

Dummy telephone number

```
dialup_device  pppdev0
    unix_device             ttya
    line_speed              38400
    modem                   none
    call_setup              dial

remote_host server
    phone                   1001
    chat_script             server.script
```

CHAT script (`server.script`) used by `lab0252-ppp` to call `server`:

Login id sent by lab0251-ppp to `server`

Password sent by lab0251-ppp to `server`

```
# Set the line regarding the remote site configuration
# Due to UUCP limitations some systems only accept cs7
#
# setlinecs7 parodd

send   RETURN
expect "ogin:"  10  onerror send BREAK repeat 3

send   "lab0252"
expect "word: " 40
#
# Set the ppp password of the remote host here
#
send  "lab0252-pass"
```

PPP Configuration File (`ppp.conf`) for `server`:

Inactivity timeout set to zero, so link stays open until closed by remote

Inactivity timeout set to zero, so link stays open until closed by remote

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 server lab0251-ppp netmask 255.255.255.0 down

ifconfig ipdptp1 plumb
ifconfig ipdptp1 server lab0252-ppp netmask 255.255.255.0 down

dialup_path
    ip_interface            ipdptp0
    expect_login_id         lab0251
    inactivity_timeout      0

dialup_path
    ip_interface            ipdptp1
    expect_login_id         lab0252
    inactivity_timeout      0
```

Link Configuration File (`link.conf`) for `server`:

Modem set to none

Accept connections

Modem set to none

Accept connections

```
dialup_device               pppdev0
    unix_device             tty0
    line_speed              38400
    modem                   none
    call_setup              answer

dialup_device               pppdev1
    unix_device             tty1
    line_speed              38400
    modem                   none
    call_setup              answer
```

Advanced IP Routing Configuration

A single synchronous PPP link is used to connect a single remote host to a local area network (LAN), as shown in Figure 5-7. Ideally, the remote host should appear as though it is connected directly to the same network.

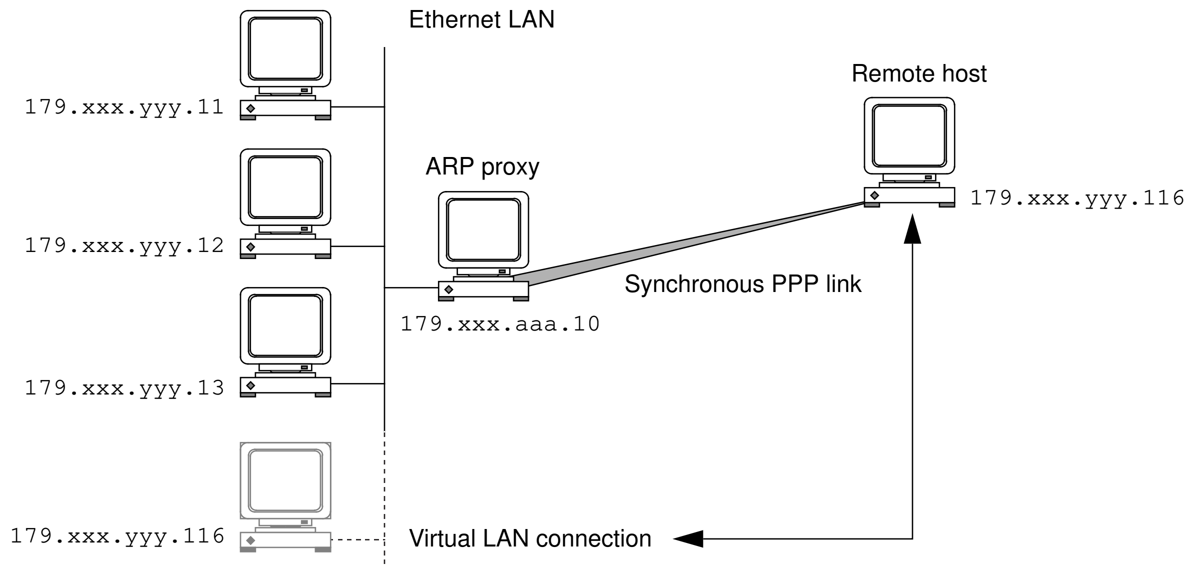


Figure 5-7 Advanced IP Routing to Create Virtual LAN Connection

There are a number of ways to create this configuration, but one of the simplest uses the `arp(1M)` command to create an ARP entry on the router so that it acts as an ARP proxy, and responds to ARP requests on behalf of the remote host. If the IP interface associated with the PPP link is configured *up*, the ARP proxy will pass IP datagrams to the remote host automatically.

Use the following `arp(1M)` command, specifying the hostname (or IP address) of the remote host and the Ethernet address of the ARP proxy, to enable this configuration:

```
prompt# arp -s hostname ether pub
```

The change is not saved, and you must run the arp(1M) command each time the router is rebooted. To ensure that the router will always respond as an ARP proxy on behalf of the remote host, create the file /etc/rc2.d/S99arp with the following contents::

```
#!/bin/sh
#

mode=$1
case "$mode" in
'start')
        if [ -f /etc/arp.cf ]; then
                 echo "Setting proxy arp entries."
                  /usr/sbin/arp -f /etc/arp.cf
                  ndd -set /dev/ip ip_forwarding 1
        fi

        exit 0
        ;;
'stop')
        exit 0
         ;;
esac
```

Create a file called /etc/arp.cf that contains the hostname (or IP address) of the remote host and the Ethernet address of the router, as follows:

```
hostname ether pub
```

The contents of the file /etc/arp.cf is read each time the router is rebooted, and the router is configured as an ARP proxy for the remote host automatically. You can configure the router as an ARP proxy for multiple hosts by making multiple entries in the file /etc/arp.cf.

Configuration using the Domain Name Service (DNS)

The domain name service (DNS) is the accepted Internet standard naming service for the resolution of hostnames and IP addresses. To use DNS in combination with Solstice PPP, you must specify your name servers and enable DNS as your naming service. Consider the previous example configuration, as shown in Figure 5-8.

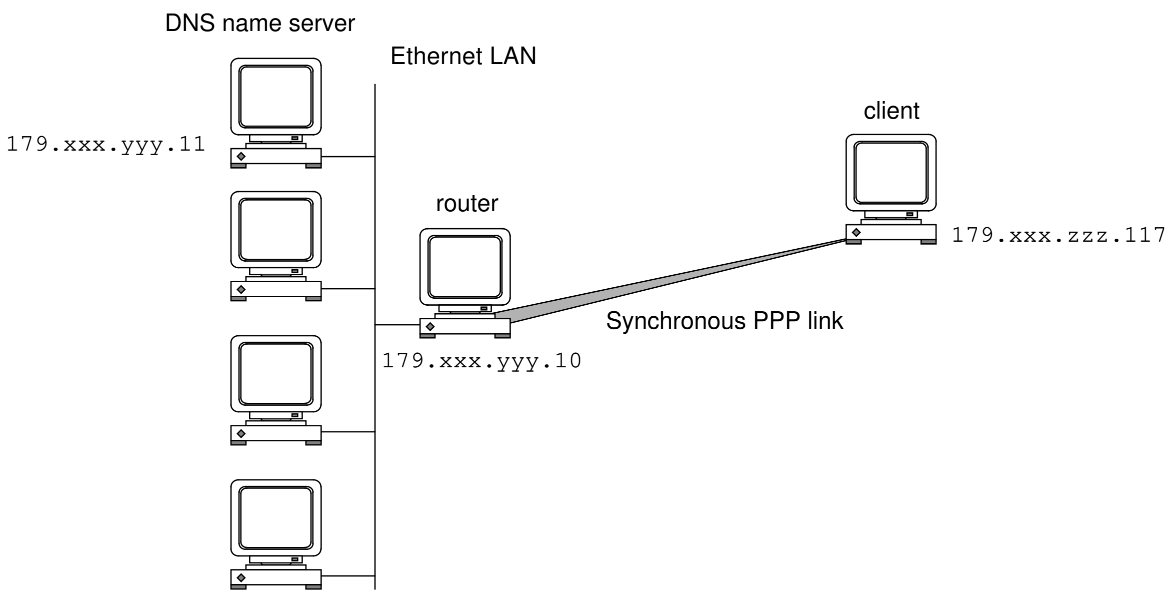


Figure 5-8 Using the Domain Name Service (DNS)

Configuring the Client

On the client, create the file `/etc/resolv.conf` that contains the domain name and the IP addresses of all the name servers. For example:

```
domain        xyz.Company.COM
nameserver    179.xxx.zzz.11
nameserver    ...
nameserver    ...
```

To use DNS as the default naming service, edit the file `/etc/nsswitch.conf` and edit the `hosts` entry as follows:

```
hosts:   files dns
```

The changes are implemented automatically. You do not need to reboot the machine.

To be certain that you can resolve all IP addresses over the PPP link, use the `default_route` keyword in the dialup path definition for the connection to the router. This adds the route to the route table as the default destination. The route is removed when the IP interface is marked *down*.

Configuring the Router

The router does not have to use DNS as its naming service, but it may do so. On the router, the IP interface for the PPP link may be configured *up* or *down*.

If the interface is *up*, the IP routing is configured when the machine is rebooted. The file `/etc/gateways` must exist on the router, so that it advertises itself to the rest of the network as a gateway.

If the interface is *down*, you must configure the routing manually, as follows:

1. **Set the `ip_forwarding` flag.**

```
prompt# ndd -set /dev/ip_forwarding 1
```

2. **Add a passive route to the remote host in the routing table.**

```
prompt# route add host remote_host_addr router_addr 1
```

   To add the route for the remote host in the example, type:

```
prompt# route add host 179.xxx.zzz.117 179.xxx.yyy.10 1
```

Alternatively, you can use a different routing mechanism, such as `gated`, to configure the IP routing for Solstice PPP.

Running PPP over PAD over X.25

The packet assembler disassembler (PAD) enables a machine to communicate over an X.25 packet switched data network (PSDN). It assembles outgoing packets so they can be forwarded over the PSDN, and disassembles incoming packets so they can be read by the remote machine.

A Solstice PPP server can be configured to accept incoming PAD calls. In the following example, a server running Solstice PPP and SunLink X.25 9.0 or later, uses the PAD emulator to receive packets sent by a PPP client.

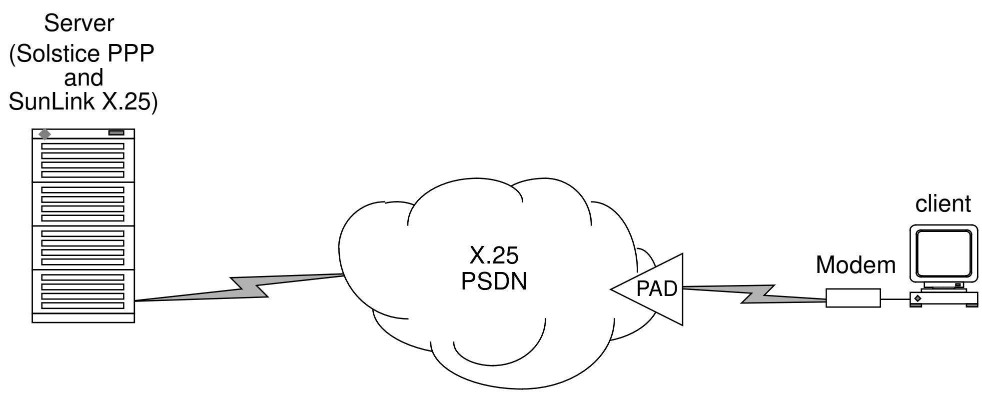


Figure 5-9 Running PPP over PAD over X.25

There is no special configuration needed for SunLink X.25. It must be configured to accept incoming PAD calls, and, in this example, the default X.3 profile for SunLink X.25 was used.

Solstice PPP sees the PAD as a normal serial line. You do not need to configure the serial line for Solstice PPP, since the PAD configuration takes care of this; therefore, you do not need a corresponding entry in the link configuration file (`link.conf`). This file must exist, however, if Solstice PPP is to start correctly.

The standard maximum receive unit (MRU) for IP running over X.25 is 576. For optimum performance, set this value using the `lcp_mru` keyword in the `dialup_path` definition in the PPP path configuration file (`ppp.conf`).

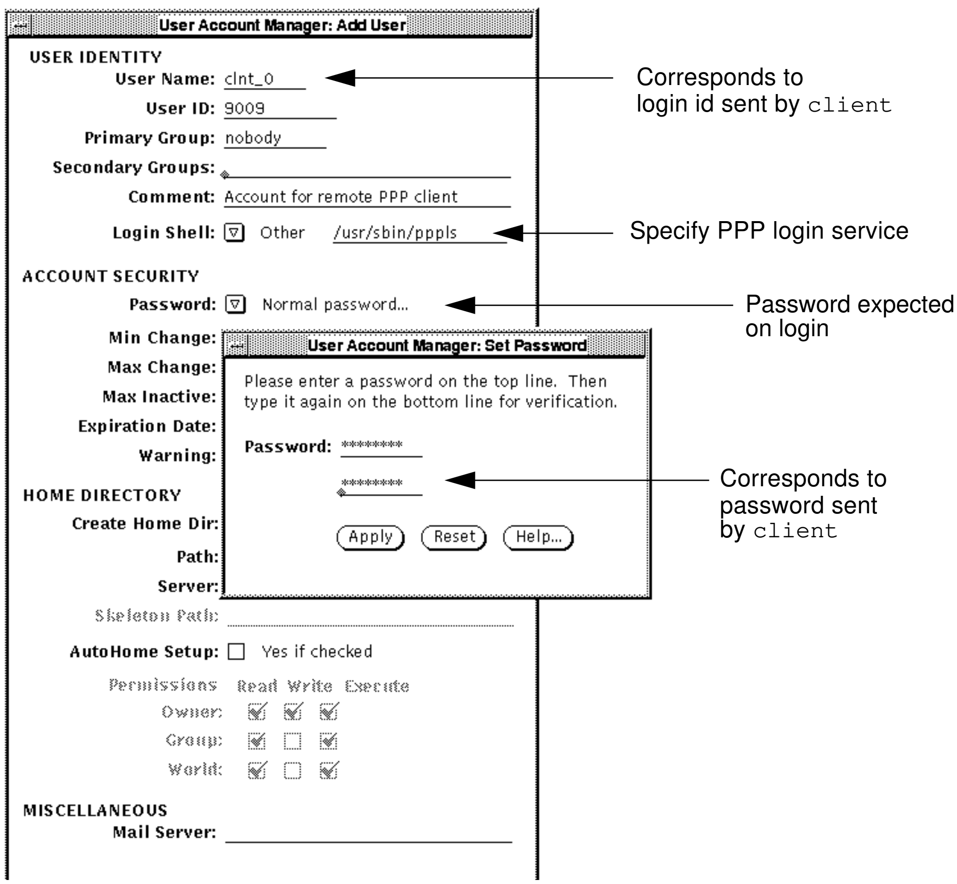
As for all Solstice PPP server configurations, you must also create a user account for the incoming connection.

PPP Configuration File (`ppp.conf`) for `server`:

```
ifconfig ipdptp0 plumb
ifconfig ipdptp0 server client netmask 255.255.255.0 down


dialup_path
    ip_interface            ipdptp0
    expect_login_id         clnt_0
    lcp_mru                 576
```

User account on `server` used to accept calls from `client`:



Troubleshooting and Diagnostics 6

This chapter tells you how to detect and resolve problems with Solstice PPP. It includes a description of how PPP links operate, instructions on how to use `ppptrace` and `pppstat` to examine the frame traffic across the PPP link, and a description of the error and status messages logged in the file `/var/opt/SUNWconn/ppp.log`.

First Steps in Troubleshooting

If you cannot establish an IP connection across a Solstice PPP link, first check the following:

1. Check the physical connections between your machine and the synchronous device or modem. If you are using a modem, check that the modem is switched on and configured correctly.

2. Check that Solstice PPP is configured and running on your machine, by typing:

```
prompt% ps -ef | grep ppp
root pid timestamp 0:00 /usr/sbin/pppd -d 1
root pid timestamp 0:00 /usr/sbin/pppd -d 1
```

3. Check that the IP interfaces for Solstice PPP are configured, by typing:

Point-to-point IP interface configured, but not up → ipdptp0

Point-to-point IP interface configured, and up → ipdptp1

```
prompt# ifconfig —a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
    inet 127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 129.x.x.117 netmask ffffff00 broadcast 129.x.x.255
    ether 8:0:20:10:c2:b1
ipdptp0: flags=8d0<POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 129.x.x.117 --> 129.x.x.253 netmask ffffff00
    ether 0:0:0:0:0:0
ipdptp1: flags=28d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>
    inet 129.x.x.117 --> 129.x.x.38 netmask ffffff00
    ether 0:0:0:0:0:0
```

4. Check for error and status messages in the Solstice PPP log file (`/var/opt/SUNWconn/ppp.log`), by typing:

Outgoing call detected → 22:55:02 line

Login sequence failed → 22:56:02 line

```
prompt# tail -f /var/opt/SUNWconn/ppp.log
05/31/95 22:52:48 - Link manager (17302) has started 05/31/95
05/31/95 22:52:48 - Successful configuration
05/31/95 22:54:49 - Solstice PPP 3.0 has found a valid license
05/31/95 22:55:02 - Connection requested to remote_host
05/31/95 22:55:03 - Dialing number 389 ...
05/31/95 22:55:21 - Got modem connection
05/31/95 22:56:02 - fail at line 12 in chat script chat_script
```

Solving Common Problems

The following sections describe common problems you may encounter when installing, configuring, or using Solstice PPP.

Problems Installing Solstice PPP

Problem: Cannot start `pkgadd(1M)`.

Solution: You must log in as `root` or become `superuser` before you can run the `pkgadd`.

Problem: Cannot find the packages for Solstice PPP.

Solution: Check that you typed the source directory correctly. If the Volume Manager (`vold`) is running on your machine, the Solstice PPP packages are located in `/cdrom/ppp_3.0`. If you are not running the Volume Manager (`vold`) on your machine, you need to mount the CD-ROM manually. See *Installing and Licensing Solstice PPP* for detailed instructions.

Problems with Licensing

Problem: Cannot start the `lit`. Command fails with error message: `permission denied`.

Solution: You must log in as `root` or become `superuser` before you can start the `lit`.

Problem: Cannot start the `lit`. Command fails with error message: `Cannot open connection to window server`.

Solution: You must enable `root` access to your Xserver.

```
prompt% $OPENWINHOME/bin/xhost + root@local_host
```

Problem: The Select Product pull-down menu does not show a license for Solstice PPP.

Solution: Each time you install the `lit` package (`SUNWlit`) associated with a product, it overwrites any previously installed version. If you are installing several licensed products you should either ensure that you install each license before installing the next product, or you should backup the license file `LIC_CONFIG_FILE.combined` to safeguard the product-specific information it contains.

Problem: The Server Options button is disabled (grayed out).

Solution: You cannot change the license server configuration once you have installed a license server on your machine. The Server Options button is disabled and the name and `hostid` of the current server(s) are displayed.

Problem: Cannot start Solstice PPP. Command fails with error:
`Cannot find license file`

Solution: You must install a license server and run the license installation script (`LIC_CONFIG_SCRIPT`) on each machine running Solstice PPP.

Problem: Cannot start SunLink PPP. Command fails with error:
`Feature has expired`

Solution: Indicates that the license for Solstice PPP has expired. If the license obtained was for a demonstration copy, it is no longer valid. You must purchase a new license and obtain a new license password from the license distribution center.

Problem: Cannot start Solstice PPP. Command fails with error:
`Encryption code in license file is inconsistent`

Solution: Any incorrect or missing license information (for example, number of RTUs, license password, expiration date) will disable your license when SunLink PPP attempts to establish a session. Restart the `lit` and re-enter the license information correctly.

Problem: Cannot start Solstice PPP. Command fails with error:
`Cannot connect to licence server`

Solution: This indicates a severe problem with the license server. Check that the licence server machine is up and that the license daemon `lmgrd.ste` is running.

Try restarting the license daemon:

```
prompt# /etc/rc2.d/S85lmgrd start
```

Problem: Cannot start Solstice PPP following a system crash. Right to Use license is reported to be in use.

Solution: If the system crashes while a licensed product is in use, the license may not be released correctly.

To check the current status of a license server:

```
prompt# cd /etc/opt/licenses
prompt# ./lmstat -c licenses_combined
lmstat - Copyright (C) 1993 Globetrotter Software Inc
Flexible License Manager status on datestamp

License server status:

    hostname: license server UP (MASTER)

Vendor daemon status:

    lic.SUNW (version): UP

Feature usage info:

    Users of PPP: (Total of <rtu> licenses available)
```

To recover a lost license for SunLink PPP:

```
prompt# cd /etc/opt/licenses
prompt# ./lmremove -c licenses_combined -v PPP
```

Problems Configuring Solstice PPP

Problem: Cannot configure modem using `pppinit` or `pppsetmod`. Operation fails with the error message “`Warning: unable to set server mode on dialup_device pppdev`*n* `for` *modem_name*”

Solution: You have selected a modem configuration that does not correspond to the modem connected to your machine. Select another modem from the modem database, or add a new modem configuration to the database, based on the information contained in the manufacturer’s documentation for your modem. See “Editing the Modem Database File (modems)” on page 70 for help.

Problems Using Solstice PPP

Problem: Solstice PPP stops working after upgrading to Solaris 2.5.

Solution: Solaris 2.5 creates a default file `/etc/ttydefs`, which overwrites some changes made by Solstice PPP. Safeguard the configuration files `ppp.conf`, `ppp.link`, and the CHAT scripts. Reinstall Solstice PPP to correct the problem.

Problem: Cannot establish PPP link. Operation fails with the status message:

```
"PPP error on ip_interface: Maximum number of
 configure requests exceeded"
```

Solution: PPP `Configure-request` frames are generated to start the link establishment phase. After a certain number of frames (defined by the keyword `lcp_max_restart`) are generated without a valid response, the connection initiator assumes that the remote host is unreachable. This may indicate one of the following:

There is a problem with the physical connection between the two hosts. Check the cables to and from the modems at each end of the link.

PPP is not running on the remote host. Check that PPP is configured and started at both ends of the link.

The link establishment phase is not completed, because the configuration negotiation does not converge. Check for configuration problems at both ends of the link.

If you are trying to establish a link over a long-delay network, such as a satellite connection, or over a congested line, the maximum number of configure requests may be exceeded before the negotiation is completed. Increase the maximum number of configure requests sent (`lcp_max_restart`) and the time between retries (`lcp_restart_timer`). See "Defining Asynchronous Paths (dialup_path)" on page 55 for detailed instructions.

Problem: Cannot establish PPP link. Operation fails with the status message: `"Authentication failed"`

Solution: The peer authentication phase failed. Check that the PAP and CHAP parameters set on the two hosts are coherent. If one host requests authentication using either PAP or CHAP, the other host must participate in the authentication phase, or the link is closed.

Problem: Cannot establish PPP link. Operation fails with the status message: `"Loop back detected"`

Solution: The PPP frames generated by the local host are being reflected by the remote host. The *magic numbers* contained in the PPP frames indicate a loop back condition. This may indicate one of the following:

There is a problem with the physical connection between the two hosts. Check the cables to and from the modems at each end of the link.

The UNIX login sequence is not completed successfully. Check that the login id and password set in the CHAT script are correct, and that a corresponding user account exists on the remote host. Check that the rest of the login dialog defined in the CHAT script is correct. If the remote host is not running a Solaris environment, you may need to modify the login sequence provided in the template file created by `pppinit(1M)`.

The remote host fails to respond quickly enough, and the maximum number of configure requests is exceeded before the negotiation is completed. Increase the maximum number of configure requests sent (`lcp_max_restart`) and the time between retries (`lcp_restart_timer`). See "Defining Asynchronous Paths (dialup_path)" on page 55 for detailed instructions.

Problem: Modem dials unexpectedly, or when the machine is rebooted.

Solution: The IP interfaces associated with asynchronous PPP links are usually marked *up*, by default. In this way, the PPP link manager initiates the PPP link automatically, when an IP datagram is passed to the interface by the IP layer.

Some applications and processes broadcast requests occasionally. For example, when searching for a license daemon, or when the machine is rebooted. The PPP link manager responds to the broadcast and tries to dial the remote host.

To prevent this behavior, mark the relevant IP interface *down*, and use `pppconn(1M)` to initiate connections as required. See "Establishing PPP Links" on page 36 for detailed instructions.

Problem: IP datagrams are not routed correctly across a synchronous PPP link used to connect to local area networks (LANs).

Solution: The two hosts at the endpoints of the PPP link act as IP routers in this configuration. Ensure that the file `/etc/gateways` exists on each host.

Problem: LCP negotiation fails with the error message: "`PPP error on ipdptp1: Negotiation of mandatory options failed`"

Solution: Check that the IP interface used to initiate the connection, and the IP interface associated with the dialup path used to accept the connection have coherent IP addresses. The source address on host must match the destination address on the other.

Problem: Cannot make `rsh(1)` or `rlogin(1)` connection to the remote host. Operation fails with the message "`Permission denied`"

Solution: Check for the hostname, or IP address, of your local host in the files `/etc/.rhosts` and `/etc/hosts.equiv` on the remote host. A + character in these files enables access for all hosts.

Checking the Physical Layer for Synchronous Links

There are four standard utilities, delivered either as part of the Solaris environment or in conjunction with the driver for the high-speed serial interface (`hsi`), that are used to test the serial devices used for synchronous links:

- `syncloop(1M)` and `syncstat(1M)`
  These utilities test the onboard serial interfaces (`zsh0` and `zsh1`)
- `hsi_loop(1M)` and `hsi_stat(1M)`
  These utilities test the high speed serial interfaces (`hih`*n*)

Caution – Do not run `syncloop` or `hsi_loop` on a interfaces that are in use. Interfaces are in loopback mode while the tests are running, and cannot communicate with remote hosts.

There are four test options associated with `syncloop` and `hsi_loop`. Table 6-1 lists the equipment you require to run the different the test options.

Table 6-1 Equipment for `syncloop` and `hsi_loop` Test Options

Test Options	Equipment
Option 1	None
Option 2	Loopback plug
Option 3	Modem(s)
Option 4	Previous setup

Using ppptrace

The ppptrace command traces the PPP frames as they are sent and received across the network. This information is the primary tool used to detect problems at the PPP link level.

See Appendix A, "PPP Link Operation" for a detailed description of the various PPP frames and their significance.

The ppptrace command has the following general form:

```
prompt# ppptrace [-x] [ppp_link_name]
```

To interpret the output from ppptrace completely, you need to be familiar with the operation of the Point-to-Point Protocol and the format of the various PPP frames. See Appendix A, "PPP Link Operation" for more information.

Tracing PPP Frames

When the ppptrace command is executed without arguments it displays a trace of all the active PPP links. You can restrict the trace by specifying an IP interface, or the link name you assigned to the link using the ppp_link_name parameter in the file ppp.conf.

To display a trace of the link associated with the IP interface ipdptp0, type:

```
prompt# ppptrace ipdptp0
```

To display a trace of the link associated with the link name ppplink0, type:

```
prompt# ppptrace ppplink0
```

The basic output from a typical ppptrace has the following components:

```
timestamp: link_id type: description [id: xx] [length: yyyy]
```

timestamp
The time at which the trace was recorded.

link_id
Identifies the link from which the trace was recorded. This may be an IP interface name (`ipdptp`*n* or `ipd`*n*) or a link name assigned using the `ppp_link_name` parameter in the file `ppp.conf`.

type
Identifies the source of the frame, which may be either *sent* by the local host (outgoing), or *received* from the remote host (incoming).

description
Describes the PPP frame type. See Appendix A, "PPP Link Operation" for more information.

id: *xx*
The contents of the id field of the PPP frame, which carries an identifier that is used to match associated requests and replies. See Appendix A, "PPP Link Operation" for more information.

length: *yyyy*
The contents of the length field of the PPP frame, which carries the total length of the information field. The length must be less than the maximum receive unit (MRU). See Appendix A, "PPP Link Operation" for more information.

For example, the following trace shows an LCP `Configure-request` packet sent by the local host. The basic trace is followed by information that describes the frame in more detail, including the *magic number*, which identifies the initial source of the frame and is used to detect looped back conditions. See Appendix A, "PPP Link Operation" for more information.

```
18:10:32: ipdptp0 Send: LCP Packet Config Request  [id: 15] [length: 0012]
          Maximun Receive unit: 1500
          Magic Number: 4e0d20fb
          Protocol Field Compression
          Address and Control Field Compression
```

Displaying PPP Frame Contents

Executing the `ppptrace` command with the `–x` option dumps the contents of the PPP frames to display the hexadecimal codes.

To dump the contents of the PPP frames, type:

Description of frame (IP packet) sent by local host

Dump of packet contents (21 indicates IP packet)

```
prompt# ppptrace –x
18:38:37: ipdptp0 Send: Internet Protocol Packet

21 45 00 00 54 86 e1 40 00 ff 01 58 9d 81 9d cc  "!E..T..@...X...."
3d 81 9d cc b1 08 00 f9 4c 02 51 00 00 2f ba 26  "=.......L.Q../.&"
8d 00 01 bb 16 08 09 0a 0b 0c 0d 0e 0f 10 11 12  "................"
13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22  ".............!""
23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32  "#$%&'()*+,-./012"
33 34 35 36 37                                    "34567           "
```

PPP Trace Examples

The following examples show how `ppptrace` can be used to detect and diagnose problems with a PPP link.

Trace Example 1—Successful Link Establishment

The following example trace shows an exchange of LPC and IPNCP packets that results in a PPP link established successfully between two hosts:

```
18:10:32: ipdptp0 Send: LCP Packet Config Request  [id: 15]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 4e0d20fb
      Protocol Field Compression
      Address and Control Field Compression
18:10:32: ipdptp0 Recv: LCP Packet Config Request  [id: 15]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 4e0d20fb
      Protocol Field Compression
      Address and Control Field Compression
```

```
18:10:32: ipdptp0 Recv: LCP Packet Config Request  [id: 1c]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 17d1cf32
      Protocol Field Compression
      Address and Control Field Compression
18:10:32: ipdptp0 Send: LCP Packet Config Request  [id: 16]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 4e0d20fb
      Protocol Field Compression
      Address and Control Field Compression
18:10:32: ipdptp0 Send: LCP Packet Config Ack      [id: 1c]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 17d1cf32
      Protocol Field Compression
      Address and Control Field Compression
18:10:33: ipdptp0 Recv: LCP Packet Config Request  [id: 1d]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 17d1cf32
      Protocol Field Compression
      Address and Control Field Compression
18:10:33: ipdptp0 Send: LCP Packet Config Ack      [id: 1d]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 17d1cf32
      Protocol Field Compression
      Address and Control Field Compression
18:10:33: ipdptp0 Recv: LCP Packet Config Ack      [id: 16]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 4e0d20fb
      Protocol Field Compression
      Address and Control Field Compression
18:10:33: ipdptp0 Send: NCP Packet Internet Protocol Config
Request  [id: 17] [length: 0010]
      IP Compression Protocol Van Jacobson
      IP Address: 129.157.204.61
18:10:33: ipdptp0 Recv: NCP Packet Internet Protocol Config
Request  [id: 1e] [length: 0010]
      IP Compression Protocol Van Jacobson
```

```
      IP Address: 129.157.204.177
18:10:33: ipdptp0 Send: NCP Packet Internet Protocol Config Ack
[id: 1e] [length: 0010]
      IP Compression Protocol Van Jacobson
      IP Address: 129.157.204.177
18:10:33: ipdptp0 Recv: NCP Packet Internet Protocol Config Ack
[id: 17] [length: 0010]
      IP Compression Protocol Van Jacobson
      IP Address: 129.157.204.61
```

Trace Example 2 — Failed CHAP Negotiation

The following example trace shows the sequence of events that occur at the caller side, when the caller requests CHAP authentication from a peer that is not configured to support CHAP:

```
11:17:41: ipdptp0 Send: LCP Packet Config Request  [id: 55]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: ae36559a
      Protocol Field Compression
      Address and Control Field Compression
11:17:42: ipdptp0 Recv: LCP Packet Config Request  [id: 55]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: ae36559a
      Protocol Field Compression
      Address and Control Field Compression
11:17:42: ipdptp0 Recv: LCP Packet Config Request  [id: 0e]
[length: 0017]
      Maximun Receive unit: 1500
      Authentication protocol: CHAP using MD5 algorithm
      Magic Number: 3ca14856
      Protocol Field Compression
      Address and Control Field Compression
11:17:42: ipdptp0 Send: LCP Packet Config Reject   [id: 0e]
[length: 0009]
      Authentication protocol: CHAP using MD5 algorithm
11:17:42: ipdptp0 Recv: LCP Packet Terminate Request [id: 0f]
[length: 0004]
```

```
11:17:42: ipdptp0 Send: LCP Packet Terminate Ack     [id: 0f]
[length: 0004]
11:17:45: ipdptp0 Send: LCP Packet Config Request  [id: 56]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: ae36559a
      Protocol Field Compression
      Address and Control Field Compression
11:17:48: ipdptp0 Send: LCP Packet Config Request  [id: 57]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: ae36559a
      Protocol Field Compression
      Address and Control Field Compression
      .
      .
      .
11:18:09: ipdptp0 Send: LCP Packet Config Request  [id: 5f]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: ae36559a
      Protocol Field Compression
      Address and Control Field Compression
```

The following example trace shows the corresponding sequence of events that occur at the answering side:

```
18:28:21: ipdptp0 Send: LCP Packet Config Request  [id: 20]
[length: 0017]
          Maximun Receive unit: 1500
          Authentication protocol: CHAP using MD5 algorithm
          Magic Number: f16105df
          Protocol Field Compression
          Address and Control Field Compression
18:28:21: ipdptp0 Recv: LCP Packet Config Reject   [id: 20]
[length: 0009]
          Authentication protocol: CHAP using MD5 algorithm
18:28:21: ipdptp0 Send: LCP Packet Terminate Request [id: 21]
[length: 0004]
18:28:21: ipdptp0 Recv: LCP Packet Terminate Ack     [id: 21]
[length: 0004]
```

Trace Example 3 — Successful CHAP Negotiation

The following example trace shows the sequence of events that occur at the caller side, when CHAP authentication is negotiated successfully:

```
13:49:58: ipdptp0 Send: LCP Packet Config Request  [id: 62]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 7bc9c0bb
      Protocol Field Compression
      Address and Control Field Compression
13:49:59: ipdptp0 Recv: LCP Packet Config Request  [id: 62]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 7bc9c0bb
      Protocol Field Compression
      Address and Control Field Compression
13:49:59: ipdptp0 Recv: LCP Packet Config Request  [id: 1b]
[length: 0016]
      Maximun Receive unit: 1500
      Authentication protocol: PAP
      Magic Number: 406bdebf
      Protocol Field Compression
      Address and Control Field Compression
13:49:59: ipdptp0 Send: LCP Packet Config Request  [id: 63]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 7bc9c0bb
      Protocol Field Compression
      Address and Control Field Compression
13:49:59: ipdptp0 Send: LCP Packet Config Ack      [id: 1b]
[length: 0016]
      Maximun Receive unit: 1500
      Authentication protocol: PAP
      Magic Number: 406bdebf
      Protocol Field Compression
      Address and Control Field Compression
13:49:59: ipdptp0 Recv: LCP Packet Config Request  [id: 1c]
[length: 0016]
      Maximun Receive unit: 1500
      Authentication protocol: PAP
      Magic Number: 406bdebf
      Protocol Field Compression
```

```
      Address and Control Field Compression
13:49:59: ipdptp0 Send: LCP Packet Config Ack      [id: 1c]
[length: 0016]
      Maximun Receive unit: 1500
      Authentication protocol: PAP
      Magic Number: 406bdebf
      Protocol Field Compression
      Address and Control Field Compression
13:49:59: ipdptp0 Recv: LCP Packet Config Ack      [id: 63]
[length: 0012]
      Maximun Receive unit: 1500
      Magic Number: 7bc9c0bb
      Protocol Field Compression
      Address and Control Field Compression
13:49:59: ipdptp0 Send: PAP Packet Authenticate Request [id: 01]
[length: 0010]
      Peer ID: 4e 41 4d 45  (NAME)
      Password: 50 41 53 53 57 44  (PASSWD)
13:49:59: ipdptp0 Recv: PAP Packet Authenticate Ack [id: 01]
[length: 0005]
      No message
13:49:59: ipdptp0 Send: NCP Packet Internet Protocol Config
Request  [id: 64] [length: 0010]
      IP Compression Protocol Van Jacobson
      IP Address: 129.157.204.61
13:49:59: ipdptp0 Recv: NCP Packet Internet Protocol Config
Request  [id: 1d] [length: 0010]
      IP Compression Protocol Van Jacobson
      IP Address: 129.157.204.177
13:49:59: ipdptp0 Send: NCP Packet Internet Protocol Config Ack
[id: 1d] [length: 0010]
      IP Compression Protocol Van Jacobson
      IP Address: 129.157.204.177
13:49:59: ipdptp0 Recv: NCP Packet Internet Protocol Config Ack
[id: 64] [length: 0010]
      IP Compression Protocol Van Jacobson
      IP Address: 129.157.204.61
```

Trace Example 4 — Dump of Successful `ping`

The following example trace shows a dump of the PPP frames exchanged during the successful execution of a `ping(1M)` command over an established PPP link:

```
18:38:37: ipdptp0 Send: Internet Protocol Packet

21 45 00 00 54 86 e1 40 00 ff 01 58 9d 81 9d cc  "!E..T..@...X...."
3d 81 9d cc b1 08 00 f9 4c 02 51 00 00 2f ba 26  "=.......L.Q../.&"
8d 00 01 bb 16 08 09 0a 0b 0c 0d 0e 0f 10 11 12  "................"
13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22  "..............!""
23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32  "#$%&'()*+,-./012"
33 34 35 36 37                                    "34567           "

18:38:37: ipdptp0 Recv: Internet Protocol Packet

ff 03 00 21 45 00 00 54 ec 77 40 00 ff 01 f3 06  "...!E..T.w@....."
81 9d cc b1 81 9d cc 3d 00 00 01 4d 02 51 00 00  ".......=...M.Q.."
2f ba 26 8d 00 01 bb 16 08 09 0a 0b 0c 0d 0e 0f  "/.&............."
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f  "................"
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f  ".!"#$%&'()*+,-./"
30 31 32 33 34 35 36 37                           "01234567        "

18:38:38: ipdptp0 Send: Internet Protocol Packet

21 45 00 00 54 86 e2 40 00 ff 01 58 9c 81 9d cc  "!E..T..@...X...."
3d 81 9d cc b1 08 00 3f 98 02 51 00 01 2f ba 26  "=......?..Q../.&"
8e 00 02 74 c8 08 09 0a 0b 0c 0d 0e 0f 10 11 12  "...t............"
13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22  "..............!""
23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32  "#$%&'()*+,-./012"
33 34 35 36 37                                    "34567           "
```

Trace Example 5 — Dump of Successful `telnet` *Session*

The following example trace shows a dump of the PPP frames exchanged during the successful execution of a `telnet(1)` command over an established PPP link:

```
18:40:44: ipdptp0 Send: Internet Protocol Packet

21 45 00 00 2c 86 ee 40 00 ff 06 58 b3 81 9d cc  "!E..,..@...X...."
3d 81 9d cc b1 80 35 00 17 94 b9 9a 00 00 00 00  "=.....5........."
00 60 02 22 38 2a be 00 00 02 04 05 b4           ".`."8*.......   "

18:40:44: ipdptp0 Recv: Internet Protocol Packet

ff 03 00 21 45 00 00 2c ec 84 40 00 ff 06 f3 1c  "...!E..,..@....."
81 9d cc b1 81 9d cc 3d 00 17 80 35 93 e9 98 00  ".......=...5...."
94 b9 9a 01 60 12 22 38 fe c2 00 00 02 04 05 b4  "....`."8........"

18:40:44: ipdptp0 Send: VJ Uncompressed TCP/IP Packet

2f 45 00 00 28 86 ef 40 00 ff 00 58 b6 81 9d cc  "/E..(..@...X...."
3d 81 9d cc b1 80 35 00 17 94 b9 9a 01 93 e9 98  "=.....5........."
01 50 10 22 38 16 80 00 00                        ".P."8....       "

18:40:44: ipdptp0 Send: VJ Compressed   TCP/IP Packet

      2d 10 17 5c ff fd 03 ff fb 18               "-..\......      "

18:40:44: ipdptp0 Recv: VJ Uncompressed TCP/IP Packet

ff 03 00 2f 45 00 00 28 ec 85 40 00 ff 00 f3 1f  ".../E..(..@....."
81 9d cc b1 81 9d cc 3d 00 17 80 35 93 e9 98 01  ".......=...5...."
94 b9 9a 07 50 10 22 32 16 80 00 00              "....P."2....    "

18:40:45: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 12 fe 70 06 ff fd 18                 "...-..p....     "

18:40:45: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 0c 16 77 03 06                                 "-..w..          "

18:40:45: ipdptp0 Recv: VJ Compressed   TCP/IP Packet
```

```
ff 03 00 2d 1f 26 52 ff fb 03 ff fa 18 01 ff f0  "...-.&R........."

18:40:45: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 14 e4 8d 09 ff fa 18 00 53 55 4e 2d 43 4d 44  "-........SUN-CMD"
ff f0                                             "..              "

18:40:45: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 0c 16 61 0d 09                        "...-..a..       "

18:40:46: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 10 7f a0 ff fb 01 ff fd 01 0d 0a 0d  "...-............"
0a 55 4e 49 58 28 72 29 20 53 79 73 74 65 6d 20  ".UNIX(r).System."
56 20 52 65 6c 65 61 73 65 20 34 2e 30 20 28 62  "V.Release.4.0.(b"
6c 6f 6e 64 65 29 0d 0a 0d 00 0d 0a 0d 00        "londe)........  "

18:40:46: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 1c 18 1d 37 0d ff fd 01 ff fc 01              "-...7.......    "

18:40:46: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 1c b4 01 06 37 6c 6f 67 69 6e 3a 20  "...-....7login:."

18:40:46: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 0c 16 1d 07 06                                 "-.....          "

18:40:46: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 1f 15 13 ff fe 01                     "...-......      "

18:40:46: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 04 16 1a 03                                    "-....           "

18:40:53: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 10 a4 10 72                                    "-...r           "

18:40:53: ipdptp0 Recv: VJ Compressed   TCP/IP Packet
```

```
ff 03 00 2d 1c a4 0f 01 03 72                    "...-.....r     "
18:40:53: ipdptp0 Send: VJ Compressed   TCP/IP Packet
2d 0b 16 18                                      "-...           "
18:40:54: ipdptp0 Send: VJ Compressed   TCP/IP Packet
2d 10 a7 0e 6f                                   "-...o          "
18:40:54: ipdptp0 Recv: VJ Compressed   TCP/IP Packet
ff 03 00 2d 1b a7 0d 6f                          "...-...o       "
18:40:54: ipdptp0 Send: VJ Compressed   TCP/IP Packet
2d 0b 16 16                                      "-...           "
18:40:54: ipdptp0 Send: VJ Compressed   TCP/IP Packet
2d 10 a7 0c 6f                                   "-...o          "
18:40:54: ipdptp0 Recv: VJ Compressed   TCP/IP Packet
ff 03 00 2d 1b a7 0b 6f                          "...-...o       "
18:40:55: ipdptp0 Send: VJ Compressed   TCP/IP Packet
2d 0b 16 14                                      "-...           "
18:40:55: ipdptp0 Send: VJ Compressed   TCP/IP Packet
2d 10 a2 0a 74                                   "-...t          "
18:40:56: ipdptp0 Recv: VJ Compressed   TCP/IP Packet
ff 03 00 2d 1b a2 09 74                          "...-...t       "
18:40:56: ipdptp0 Send: VJ Compressed   TCP/IP Packet
2d 0b 16 12                                      "-...           "
18:40:59: ipdptp0 Send: VJ Compressed   TCP/IP Packet
```

```
2d 10 09 08 0d 00                                  "-.....          "

18:40:59: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 1c 08 fc 02 01 0d 0a                   "...-.......     "

18:40:59: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 0b 16 0e                                        "-...            "

18:40:59: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 1f 2e 33 50 61 73 73 77 6f 72 64 3a  "...-..3Password:"
20                                                 ".               "

18:40:59: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 04 16 04 0a                                     "-....           "

18:45:46: ipdptp0 Recv: Internet Protocol Packet

ff 03 00 21 45 00 00 28 ec 92 40 00 ff 06 f3 12  "...!E..(..@....."
81 9d cc b1 81 9d cc 3d 00 17 80 35 93 e9 98 5e  ".......=...5...^"
94 b9 9a 20 50 11 22 38 16 03 00 00              "....P."8....    "

18:45:46: ipdptp0 Send: VJ Compressed   TCP/IP Packet

2d 04 16 03 01                                     "-....           "

18:45:46: ipdptp0 Send: Internet Protocol Packet

21 45 00 00 28 87 02 40 00 ff 06 58 a3 81 9d cc  "!E..(..@...X...."
3d 81 9d cc b1 80 35 00 17 94 b9 9a 20 93 e9 98  "=.....5........."
5f 50 11 22 38 16 02 00 00                         "_P."8....       "

18:45:46: ipdptp0 Recv: VJ Compressed   TCP/IP Packet

ff 03 00 2d 2c 16 02 01 0b 02                      "...-,.....      "
```

Using pppstat

The pppstat command displays information about the number and type of IP packets and PPP frames sent or received since Solstice PPP was started. This information provides a record of link usage and can be used to highlight problems with the network.

See Appendix A, “PPP Link Operation" for a detailed description of the various PPP frames and their significance.

The pppstat command has the following general form:

```
prompt# pppstat [−i] [−e] [−l] [−n] [ppp_link_name]
```

Displaying IP Statistics

Executing the pppstat command with the −i option displays information about the IP packets sent and received since Solstice PPP was started. A single IP datagram is encapsulated in each PPP frame.

```
prompt# pppstat −i

IP status for PPP MIB link number 1

33 IP   packets in (1500 octets total)
38 IP   packets out (366 octets total)

2       packets in  TYPE IP
3       packets in  TYPE VJ UNCOMPRESSED
28      packets in  TYPE VJ COMPRESSED
2       packets out TYPE IP
3       packets out TYPE VJ UNCOMPRESSED
33      packets out TYPE VJ COMPRESSED
```

Displaying PPP Error Statistics

Executing the `pppstat` command with the `-e` option displays information about the PPP error frames sent and received since Solstice PPP was started.

```
prompt# pppstat -e

Errors status for PPP MIB link number 1

0 bad HDLC address fields, last was 0x00
0 bad HDLC control fields, last was 0x00
0 frames received with invalid protocol id, last was 0x0000
0 packets exceeded local maximum receive unit
0 packets were too short to be valid
0 packets had headers which were too short
0 packets received with bad CRC
1 Configure Request packets retransmitted due to timeout
0 Terminate Request packets retransmitted due to timeout
```

Displaying PPP LCP Statistics

Executing the `pppstat` command with the `-l` option displays information about the PPP LCP (link control protocol) frames sent and received since Solstice PPP was started.

```
prompt# pppstat -l

LCP status for PPP MIB link number 1

4 LCP packets in (72 octets)   4 LCP packets out (72 octets)
0 LCP protocol rejects
2 LCP Configure Requests sent  3 LCP Configure Requests received
2 LCP Configure Acks sent      1 LCP Configure Acks received
0 LCP Configure Naks sent      0 LCP Configure Naks received
0 LCP Configure Rejs sent      0 LCP Configure Rejs received
0 LCP Terminate Requests sent  0 LCP Terminate Requests received
0 LCP Terminate Acks sent      0 LCP Terminate Acks received
0 LCP Code Rejects sent        0 LCP Code Rejects received
0 LCP Echo Requests sent       0 LCP Echo Requests received
0 LCP Echo Responses sent      0 LCP Echo Responses received
0 LCP Disconnect Reqs sent     0 LCP Disconnect Reqs received
```

Displaying IP NCP Statistics

Executing the `pppstat` command with the `−n` option displays information about the IP NCP (network control protocol) frames sent and received since Solstice PPP was started.

```
prompt# pppstat −n

IPNCP status for PPP MIB link number 1

3 IPNCP packets in (42 octets)     3 IPNCP packets out (48 octets)
0 IPNCP protocol rejects
2 IPNCP Configure Requests sent 1 IPNCP Configure Reqs received
1 IPNCP Configure Acks sent     1 IPNCP Configure Acks received
0 IPNCP Configure Naks sent     1 IPNCP Configure Naks received
0 IPNCP Configure Rejs sent     0 IPNCP Configure Rejs received
0 IPNCP Terminate Requests sent 0 IPNCP Terminate Reqs received
0 IPNCP Terminate Acks sent     0 IPNCP Terminate Acks received
0 IPNCP Code Rejects sent       0 IPNCP Code Rejects received
0 IPNCP Echo Requests sent      0 IPNCP Echo Requests received
0 IPNCP Echo Responses sent     0 IPNCP Echo Responses received
0 IPNCP Disconnect Reqs sent    0 IPNCP Disconnect Reqs received
```

Checking the IP Routing

IP routing is a common source of problems with products such as Solstice PPP, which implement IP connections over PPP links. To detect problems with IP routing:

1. Check the protocol status, using `ifconfig(1M)`
2. Check the IP connectivity, using `ping(1M)`
3. Check the routing tables, using `netstat(1M)`
4. Check the packet flow, using `snoop(1M)`

Checking the Protocol Status (`ifconfig`)

Use `ifconfig(1M)` to check that the IP interfaces for Solstice PPP are configured, by typing:

Point-to-point IP interface configured, but not up

Point-to-point IP interface configured, and up

```
prompt# /usr/sbin/ifconfig —a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
    inet 127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 129.x.x.117 netmask ffffff00 broadcast 129.x.x.255
    ether 8:0:20:10:c2:b1
ipdptp0: flags=8d0<POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 129.x.x.117 --> 129.x.x.253 netmask ffffff00
    ether 0:0:0:0:0:0
ipdptp1: flags=28d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>
    inet 129.x.x.117 --> 129.x.x.38 netmask ffffff00
    ether 0:0:0:0:0:0
```

Checking the IP Connectivity (`ping`)

Use `ping(1M)` to check that you can reach a remote host, by typing:

```
prompt# /usr/sbin/ping —r remote_host
```

Follow the status of the connection in the file `/var/opt/SUNWconn/ppp.log`. If the LCP negotiation fails with the message "`PPP error on ipdptp1: Negotiation of mandatory options failed`", the IP addresses used for the interfaces at each end of the link may not be coherent.

Checking the Routing Tables (`netstat`)

Use `netstat(1M)` to check the network addresses of the local and remote hosts, by typing:

```
prompt# /usr/sbin/netstat —i
lo0     8232  loopback  localhost 4631890 0  4631890   0    0      0
le0     1500  lab       local     5370160 15 2153627   273  430451 0
ipdptp0 1500  remote    local     0       0  0         0    0      0
```

Check that the local and remote addresses are correct. Hostnames must be resolved as IP addresses in either the local file `/etc/hosts`, or by a naming service (NIS/NIS+ or DNS). IP interfaces for synchronous links are initialized by Solstice PPP before the naming services are available; therefore the hostname for these files *must* be resolved in `/etc/hosts`.

Use `netstat(1M)` to check the routing table, by typing:

```
prompt# /usr/sbin/netstat —r
```

If the routing table does not include the expected routes, check that the routing daemon `in.routed` is running on your machine, by typing:

```
prompt# ps —ef | grep in.routed
owner 12481  3812  7 23:06:08 pts/6    0:00 grep in.routed
```

Checking the Packet Flow (`snoop`)

Use `snoop(1M)` to check that packets are being transmitted. For example:

```
prompt# /usr/sbin/snoop —d ipdptp0
local -> remote         RSHELL C port=1017   >0   'H   ![    ?
local -> remote         RSHELL C port=1017   ,(!\nM/
local -> remote         RSHELL C port=1017   Y@#K /  ]0#Z    ] 'Z
local -> remote         RSHELL C port=1017   .?;-)P3O89, "  2 H8"
local -> remote         RSHELL C port=1017   @         \#\
.
.
.
```

Understanding the Log File

Error and status messages for Solstice PPP are written to the log file `/var/opt/SUNWconn/ppp.log`. See "Status and Error Messages" on page 143 for a complete list of the messages which may appear in the log file.

Note – The log file can grow very large and should be truncated regularly.

Successful Connection Attempt

The following log file extract shows the messages logged when Solstice PPP is started successfully on a client, and a connection to a remote server is opened and then disconnected:

Solstice PPP started successfully →

Modem connection complete →

PPP negotiation complete and connection open →

Disconnect request received and connection closed →

```
11/14/95 15:32:55 - Link manager (904) has started 11/14/95
11/14/95 15:32:55 - Successful configuration
11/14/95 15:33:06 - Connection requested to miles
11/14/95 15:33:07 - Dialing number P365 ...
11/14/95 15:33:40 - Got modem connection
11/14/95 15:33:42 - LCP up on ipdptp0
11/14/95 15:33:42 - IP_NCP up on ipdptp0
11/14/95 15:33:42 - IP up on interface ipdptp0, with timeout set
to 120 seconds
11/14/95 15:35:41 - Disconnect indication on ipdptp0
11/14/95 15:35:41 - IP_NCP down on ipdptp0
11/14/95 15:35:42 - LCP down on ipdptp0
```

The corresponding log file on the server shows the messages logged when the incoming connection is accepted and then disconnected:

Call accepted from user with login id ppp2 →

PPP negotiation complete and connection open →

Disconnect request received and connection closed →

```
11/14/95 15:28:58 - Received a call on pppdev0
11/14/95 15:28:59 - Using dialup_path with expect_login_id ppp2
11/14/95 15:28:59 - LCP up on ipdptp0
11/14/95 15:28:59 - IP_NCP up on ipdptp0
11/14/95 15:29:00 - IP up on interface ipdptp0, with timeout set
to 120 seconds
11/14/95 15:30:59 - interface ipdptp0 has been disconnected
11/14/95 15:30:59 - Device pppdev0 has been disconnected
```

Problems Configuring Solstice PPP

When Solstice PPP is started, it reads the configuration files that you created using the configuration script `pppinit`. If these files do not exist, Solstice PPP displays the following message:

```
starting ppp (not configured)
```

The following log file extract shows the log messages that are generated when one of the configuration files contains an error:

```
11/13/95 18:53:22 - Link manager (460) has started 11/13/95
11/13/95 18:53:22 - parse_config_file: unrecognized symbol
nactivity_timeout
11/13/95 18:53:22 - parse_config_file: unrecognized symbol 180
11/13/95 18:53:22 parse_config_file: Errors in configuration file
/etc/opt/SUNWconn/ppp/ppp.conf
```

In this example, there is an unrecognised keyword contained in the file `ppp.conf`, which is rejected when the file is parsed. Check the configuration files for typographical errors and illegal values assigned to keywords. Restart Solstice PPP to read the configuration.

Problems Getting the Modem Connection

The first step in the connection phase is the modem connection. The client dials the telephone number of the modem connected to the server, and the two modems communicate to set up the connection.

The following log file extract shows the log messages that are generated when a client fails in an attempt to establish the modem connection:

```
11/13/95 19:57:44 - Connection requested to miles
11/13/95 19:57:45 - Dialing number P365 ...
11/13/95 19:58:09 - remote host is busy
```

The connection request never reaches the server; therefore, there are no corresponding messages in the log file on the server.

The message *remote host is busy* is displayed whenever the client fails to make the modem connection. This may mean that the server modem is already in use by another client, or that it has failed to terminate a previous connection cleanly. It may also mean that there is a problem with the equipment. In this example, the error was provoked by using tone dialing with an office exchange that expected pulse dialing.

Try telephoning the number directly to make sure that you can reach the modem. If you cannot reach the modem, check the telephone number the client is dialing.

Problems with the CHAT Script

The CHAT scrip is executed automatically each time a client initiates a call. It is used to exchange information with the server during the connection phase.

The following log file extract shows the log messages generated when the client did not receive the response it expected from the server:

```
11/14/95 15:38:52 - Connection requested to miles
11/14/95 15:38:53 - Dialing number P365 ...
11/14/95 15:39:35 - Got modem connection
11/14/95 15:39:53 - Timeout expired
11/14/95 15:40:03 - Timeout expired
11/14/95 15:40:13 - Timeout expired
11/14/95 15:40:23 - Timeout expired
11/14/95 15:40:23 - fail at line 18 in chat script
/etc/opt/SUNWconn/ppp/script/miles.scr
```

The following log file extract shows the log messages generated when the client did not send the response expected by the server:

```
11/14/95 15:58:03 - Connection requested to miles
11/14/95 15:58:04 - Dialing number P365 ...
11/14/95 15:58:37 - Got modem connection
11/14/95 15:59:16 - Timeout expired
11/14/95 15:59:16 - fail at line 25 in chat script
/etc/opt/SUNWconn/ppp/script/miles.scr
```

In some circumstances, an error in the CHAT script can provoke a loopback condition when the login sequence is not completed successfully. The following log file extract show the error messages generated by a loopback condition:

```
11/14/95 15:45:05 - Connection requested to miles
11/14/95 15:45:06 - Dialing number P365 ...
11/14/95 15:45:42 - Got modem connection
11/14/95 15:45:58 - PPP error on ipdptp0: Loop back detected
11/14/95 15:45:58 - Check if your connect script is correct.
11/14/95 15:45:58 - Or PPP may not be started on the remote host
```

Problems with PPP Negotiation

Once you have a modem connection and completed the login phase, the next step in the connection phase is the PPP negotiation. The client and the server communicate to negotiate a common configuration for the PPP link. The policy is to converge if at all possible; however, failure to agree on certain mandatory parameters will cause the negotiation to fail, as shown in the following log file extract:

```
11/13/95 20:34:42 - Connection requested to miles
11/13/95 20:34:43 - Dialing number P365 ...
11/13/95 20:35:16 - Got modem connection
11/13/95 20:35:19 - LCP up on ipdptp0
11/13/95 20:35:19 - PPP error on ipdptp0: Negotiation of
mandatory options failed
```

Problems with PAP and CHAP Authentication

The following log file extract shows what happens when a client fails to respond to a request for authentication:

```
11/14/95 10:22:41 - Connection requested to miles
11/14/95 10:22:42 - Dialing number P365 ...
11/14/95 10:23:16 - Got modem connection
11/14/95 10:23:47 - PPP error on ipdptp0: Maximum number of
configure requests exceeded
```

In this example, the server requested PAP authentication and the client rejected the request. After a specified number of requests, the server broke the connection without starting the PPP negotiation.

The following log file extract shows what happens when the client responds to the request for authentication with the wrong password:

```
11/14/95 15:20:04 - Connection requested to miles
11/14/95 15:20:04 - Dialing number P365 ...
11/14/95 15:20:09 - Got modem connection
11/14/95 15:20:09 - LCP up on ipdptp0
11/14/95 15:20:09 - PPP error on ipdptp0: Negotiation of
mandatory options failed
```

In both cases, the server sees a failed PPP negotiation and generates the following log file messages:

```
11/14/95 15:20:08 - Received a call on pppdev0
11/14/95 15:20:09 - Using dialup_path with expect_login_id ppp2
11/14/95 15:20:09 - PPP error on ipdptp0: Negotiation of
mandatory options failed
11/14/95 15:20:09 - Device pppdev0 has been disconnected
```

Problems with the Inactivity Timeout

The inactivity timeout closes the connection automatically when it remains unused for a specified number of seconds. This means that you do not pay for telephone connections that are left open accidently. However, if the inactivity timeout is too short, your connection may be closed prematurely.

The following log file extract shows an inactivity timeout triggered by the client after 60 seconds:

Disconnect triggered by inactivity timeout on the client side

```
11/13/95 18:00:03 - IP up on interface ipdptp0, with timeout set
to 60 seconds
11/13/95 18:00:14 - Interface ipdptp0 has timed out
```

If the connection times out systematically, increase the inactivity timeout for calls to the server.

The following log file extract shows a connection that has been disconnected by the server for some reason. One possible cause is an inactivity timeout on the server that is shorter than the inactivity timeout on the client. It may also have been caused by a normal disconnect request from the server side.

Disconnect triggered by server for an unspecified reason

```
11/13/95 18:13:46 - IP up on interface ipdptp0, with timeout set
to 240 seconds
11/13/95 18:15:47 - interface ipdptp0 has been disconnected
```

Status and Error Messages

The following sections list the status messages and most common error messages that appear in the log file `/var/opt/SUNWconn/ppp.log`.

Note – The log file can grow very large and should be truncated regularly.

To display the messages logged in `ppp.log`, type:

```
prompt% tail -f /var/opt/SUNWconn/ppp.log
```

Status Messages

The following status messages trace the progress of successful IP connections over PPP links, and are displayed during normal link operation. They may also indicate problems that occur as a result of configurations errors.

`add_default_route:default route not allowed on ipd device`
The keyword `default_route` is not allowed in path definitions associated with point-to-multipoint (`ipd`*n*) IP interfaces.

`Check that the connect script is correct,`
`or PPP may not be started on the remote host`
Indicates that the link manager failed to establish a PPP link to the remote host. This may be because PPP is not running on the remote host, or it may indicate that there is an error in the login dialog defined in the connect script. Check that there is a valid login id and password contained in the script, and that a corresponding user account exists on the remote host. If

the remote host is not running Solaris, you may need to change other components in the login dialog. See "Editing the CHAT (or Connect) Scripts" on page 72 for detailed instructions.

`Connection closed on` *ip_interface*
Indicates that an IP connection for the specified interface (`ipd`*n* or `ipdptp`*n*) was closed normally, following a terminate request from either the local or the remote host.

`Connection requested to` *remote_host*
Indicates that the link manager received a request for an IP connection to the specified remote host. The remote host must be defined in the file `link.conf`.

`Device` *unix_device* `has been disconnected`
Indicates that the modem connection has been closed normally, following a disconnect request generated by one of the endpoints in the link.

`Dialing number` *phone_number* `...`
Indicates that the modem is dialing the specified number to establish a physical connection to the remote host.

`Dialup_path` *login_id* `already used for another connection`
Indicates that two remote hosts, using the same login id, have made simultaneous connections. Only one connection can be treated at a time.

`Dialup_path not defined for outgoing call`
Indicates that there is no remote host associated with the dialup path definition that is being used in an attempt to initiate a connection. Modify the dialup path definition in the file `ppp.conf` to add a remote host, and add a corresponding remote host definition to the file `link.conf`, if necessary.

`Disconnect indication on` *ip_interface*
Indicates that a PPP terminate request was generated by one of the endpoints in the link. This usually indicates a normal disconnection following the expiry of the inactivity timeout, for example.

`Got modem connection`
Indicates that the modem has completed dialing successfully and that the physical connection has been established.

`Host [`*remote_host*`] not found in the host list`
Indicates that the link manager is trying to establish a connection to a remote host that is not defined in the file `link.conf`. Check that the `remote_host` keyword in the dialup path definition points to a valid remote host definition.

`Host` *remote_host* `is not available`
Indicates that the remote host could not be accessed for some reason. It may indicate that the remote device is busy.

`IP up on interface` *ip_interface*`, with timeout set to` *timeout*
Indicates that an IP connection has been established successfully for the specified interface (`ipd`*n* or `ipdptp`*n*). The IP connection will be closed automatically after *timeout* seconds of inactivity, or earlier if the inactivity timeout set at the remote end of the connection is set to a lower value.

`IP up on` *ip_interface*
Indicates that the IP layer for the specified IP interface (`ipd`*n* or `ipdptp`*n*) has been configured and brought up. The IP connection has been completed successfully, and the timeout has been set to zero. The IP connection will remain open until closed explicitly, by stopping Solstice PPP or running `pppdisc(1M)`.

`IP_NCP down on` *ip_interface*
Indicates that the IP network control protocol (NCP) layer for the specified interface (`ipd`*n* or `ipdptp`*n*) has been brought down. This may be due to a termination request generated by either host.

`IP_NCP up on` *ip_interface*
Indicates that the IP network control protocol (NCP) layer for the specified IP interface (`ipd`*n* or `ipdptp`*n*) has been configured and brought up. See Appendix A, “PPP Link Operation" for more information about this phase of the PPP link negotiation.

`Interface` *ip_interface* `has timed out`
Indicates that the specified IP interface (`ipd`*n* or `ipdptp`*n*) has been closed automatically after the inactivity timeout at one end of the link has expired.

`Invalid empty name for chat script`
Indicates that there is no CHAT script defined in the file `link.conf` for this remote host. You must create a unique CHAT script for each remote host to which the local host will initiate connections. See “Editing the CHAT (or Connect) Scripts” on page 72 for detailed instructions.

`LCP down on` *ip_interface*
: Indicates that the link control protocol (LCP) layer for the specified IP interface (`ipd`*n* or `ipdptp`*n*) has been brought down. This may be due to a termination request generated by either host.

`LCP up on` *ip_interface*
: Indicates that the link control protocol (LCP) layer for the specified IP interface (`ipd`*n* or `ipdptp`*n*) has been configured and brought up. See Appendix A, “PPP Link Operation" for more information about this phase of the PPP link negotiation.

`License manager has exited`
: Indicates that there is a problem with the license system. Check that the license daemon is configured and running.

`Link manager (`*pid*`) has started` *date*
: Indicates that the PPP link manager has been configured and started successfully.

`No device available for` *ip_interface*
: Indicates that all there are no more modems available. This is normally a temporary situation, which occurs when there are more IP connections requested than there are modems.

`PPP error on` *ip_interface*`:` *reason*
: Indicates that an error occurred during either the link establishment, or the peer authentication phase. The reason for the error is given as part of the error message. Use `ppptrace(1M)` for further diagnosis.

`Received incoming call while dialing`
: Indicates that the local host received a connection request from a remote host, while it was attempting to initiate an outgoing call on the same modem. This message is displayed for information only, local host will continue it call.

`Received a call on` *ip_interface*
: Indicates that the local host accepted an incoming connection for the specified IP interface (`ipd`*n* or `ipdptp`*n*).

`Remote host is busy`
: Indicates that the call could not be completed because the remote modem is already being used for another connection.

`Resetting the modem to server mode`
Indicates that the program `pppsetmod(1M)` has been used to initialize the modem in server (or answer) mode so that it waits for incoming connections. This programs sends the `server` string contained in the modems database file `/etc/opt/SUNWconn/ppp/modems`.

`Successful configuration`
Indicates that the configuration files `ppp.conf` and `link.conf` were parsed successfully when Solstice PPP was started.

`PPP has found a valid license`
Indicates that the license daemon is running and that Solstice PPP has been given a license token.

`PPP is waiting to get a license`
Indicates that the license daemon is running, but that has assigned all the available license tokens to other users. PPP is waiting for a token to be released.

`Timeout expired`
Indicates that the IP connection has been inactive for the number of seconds defined by the `inactivity_timeout` parameter (default 120 seconds). The connection is closed automatically.

`Unable to find the required expect_login_id` *login_id*
Indicates that the PPP login service has accepted an incoming connection, following a successful UNIX login sequence, but that the link manager has not found a dialup path definition in the file `ppp.conf` with a `expect_login_id` parameter that matches the login id sent by the remote host.

`Unable to get license, PPP exiting`
Indicates that Solstice PPP was unable to get a license token from the license daemon after a certain number of retries. Check that the license daemon is installed correctly and running. If the license daemon is not running on the same machine as Solstice PPP, check that there is a network connection that allows access to the daemon.

`Unable to regain license, PPP closing down`
Indicates that Solstice PPP lost the license token was unable to get it back after a certain number of retries. Check that the license daemon is still running.

`Using dialup_path with expect_login_id` *login_id*
Indicates that the PPP login service has accepted an incoming connection, following a successful UNIX login sequence, and that the link manager has identified the corresponding dialup path definition.

Configuration Error Messages

The following configuration error messages are displayed if errors are encountered when the PPP path configuration file `ppp.conf` is parsed. In all cases, you should modify the file to correct the error and restart Solstice PPP to correct the problem.

`expect_chap_name "`*name*`" ends in CR/LF`
Indicates that the specified CHAP name is terminated with a carriage return (CR) or line-feed (LF) escape sequence.

`expect_chap_name "`*name*`" ends in NUL`
Indicates that the specified CHAP name is terminated with a NULL character.

`default_route: no path defined`
Indicates that the specified `default_route` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`host IP address not found:` *hostname*
Indicates that the parser was unable to resolve specified hostname to find the IP address. For a synchronous path, the hostname must be associated with an IP address in the file `/etc/hosts` on the local machine. For an asynchronous path, the hostname may appear in the file `/etc/hosts`, or be specified by some other naming system.

`inactivity_timeout` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `inactivity_timeout` parameter. You must enter an integer.

`inactivity_timeout: no path defined`
Indicates that the specified `inactivity_timeout` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`interface number` *n* `not recognized`
Indicates that the specified number does not correspond to any of the IP interfaces defined in the configuration file.

`invalid ip_interface:` *string*
Indicates that the specified string does not correspond to a valid point-to-point (`ipdptp`*n*) or point-to-multipoint (`ipd`*n*) IP interface.

`ip_interface: no path defined`
Indicates that the specified `ip_interface` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`ipcp_compression value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `ipcp_compression` parameter. Valid values are `vj` (enabled) and `off` (disabled).

`ipcp_compression: no path defined`
Indicates that the specified `inactivity_timeout` keyword appears in the configuration file, but is not associated with any dialup path definition.

`ipd or ipdptp keyword expected`
Indicates that the `ip_interface` keyword appears without specifying a valid point-to-point (`ipdptp`*n*) or point-to-multipoint (`ipd`*n*) IP interface.

`lcp_async_map value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `lcp_async_map` parameter. You must enter a hexadecimal value between `0x0` and `0xffffffff`.

`lcp_async_map: no path defined`
Indicates that the specified `lcp_async_map` keyword appears in the configuration file, but is not associated with any dialup path definition.

`lcp_compression value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `lcp_compression` parameter. Valid values are `on` (enabled) and `off` (disabled).

`lcp_compression: no path defined`
Indicates that the specified `lcp_compression` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`lcp_max_restart value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `lcp_max_restart` parameter. You must enter an integer between 1 and 255.

`lcp_max_restart: no path defined`
Indicates that the specified `lcp_max_restart` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`lcp_mru value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `lcp_mru` parameter. You must enter an integer between 1 and 255.

`lcp_mru: no path defined`
Indicates that the specified `lcp_mru` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`lcp_restart_timer value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `lcp_restart_timer` parameter. You must enter an integer.

`lcp_restart_timer: no path defined`
Indicates that the specified `lcp_restart_timer` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`link_monitor value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `link_monitor` parameter. Valid values are `on` (enabled) and `off` (disabled).

`link_monitor: no path defined`
Indicates that the specified `link_monitor` keyword appears in the configuration file, but is not associated with any synchronous path definition.

`link_monitor_retries value` *string* `not recognized`
Indicates that the specified string is not a valid value for the `link_monitor_retries` parameter. You must enter an integer.

`link_monitor_retries: no path defined`
Indicates that the specified `link_monitor_retries` keyword appears in the configuration file, but is not associated with any synchronous path definition.

`link_monitor_timer value` *string* `not recognized`
Indicates that the specified string is not a valid value for the `link_monitor_timer` parameter. You must enter an integer.

`link_monitor_timer: no path defined`
Indicates that the specified `link_monitor_timer` keyword appears in the configuration file, but is not associated with any synchronous path definition.

`malformed ip address:` *string*
Indicates that the specified string is not a valid IP address, entered in Internet dot notation. IP addresses should have the form: `xxx.xxx.xxx.xxx`

`must specify chap_own_secret`
Indicates that the `send_authentication` keyword has been used to enable CHAP authentication when requested by a remote host, but that the mandatory CHAP parameter `chap_own_secret` has not been set.

`must specify chap_peer_secret`
Indicates that the `send_authentication` keyword has been used to enable CHAP authentication when requested by a remote host, but that the mandatory CHAP parameter `chap_peer_secret` has not been set.

`must specify expect_chap_name`
Indicates that the `expect_authentication` keyword has been used to request CHAP authentication from a remote host, but that the mandatory CHAP parameter `expect_chap_name` has not been set.

`must specify send_chap_name`
Indicates that the `send_authentication` keyword has been used to request CHAP authentication from a remote host, but that the mandatory CHAP parameter `send_chap_name` has not been set.

`no paths defined in` *filename*
The PPP path configuration file ppp.conf must contain at least one synchronous or asynchronous path definition. See “Editing the PPP Path Configuration File (ppp.conf)” on page 46 for detailed instructions.

`off, pap, or chap required:` *string*
Indicates that the specified string is not a valid value for the `expect_authentication` parameter. Valid values are `off` (no authentication), `pap` (authentication using PAP), `chap` (authentication using CHAP), or `pap|chap` (authentication using both PAP and CHAP).

`ppp_link_name: no path defined`
Indicates that the specified `ppp_link_name` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`remote_host too long:` *string*
Indicates that there was not enough memory available to store the value of the `remote_host` parameter. This error message may indicate a more serious system error.

`remote_host: no path defined`
Indicates that the specified `remote_host` keyword appears in the configuration file, but is not associated with any dialup path definition.

`remote_ip_addr: no path defined`
Indicates that the specified `remote_ip_addr` keyword appears in the configuration file, but is not associated with any dialup path definition.

`request_ip_addr value` *string* `not recognized`
Indicates that the specifies string is not a valid value for the `request_ip_addr` parameter. Valid values are `on` (enabled) and `off` (disabled).

`unix_device: no path defined`
Indicates that the specified `unix_device` keyword appears in the configuration file, but is not associated with any synchronous or dialup path definition.

`wildcards (* | ?) not legal for ipd`
You cannot use wildcards to specify point-to-multipoint (`ipd`*n*) interfaces. In particular, you cannot use an asterisk (*) to specify a dynamic IP interface.

System Error Messages

System error messages indicate severe system or software errors, and should not appear during normal operation. They have the following form:

```
function_name: error_description failed
```

For example, the following system error indicates a memory allocation problem:

```
do_chap_name: malloc failed
```

If system errors occur, try stopping and restarting Solstice PPP. If system errors continue to occur systematically, contact your authorized service provider, and provide a description of the error message and the circumstances under which it is displayed.

Files and Directories

The following files and directories are created on your machine when you install Solstice PPP using the default base directories.

Solstice PPP Device Drivers (`SUNWpppk`*)*

Contains the device drivers for the Solstice PPP STREAMS module, connection manager, point-to-point and point-to-multipoint IP interfaces.

Table 6-2 Solstice PPP Device Drivers (`SUNWpppk`)

Files and Directories	**Description**
`/usr/kernel/drv/ipd`	Driver for IP point-to-multipoint interfaces
`/usr/kernel/drv/ipd.conf`	Configuration file for `ipd`
`/usr/kernel/drv/ipdcm`	Connection manager daemon
`/usr/kernel/drv/ipdcm.conf`	Configuration file for `ipdcm`
`/usr/kernel/drv/ipdptp`	Driver for IP point-to-point interfaces
`/usr/kernel/drv/ipdptp.conf`	Configuration file for `ipdptp`
`/usr/kernel/drv/ppp`	PPP STREAMS module
`/usr/kernel/drv/ppp.conf`	Configuration file for `ppp`

Solstice PPP Login Service (`SUNWppps`*)*

Contains the PPP login service daemon and connection process to support incoming calls. This package should only be installed if you have also purchased alicense for Solstice PPP to enable the server functionality.

Table 6-3 Solstice PPP Daemon and User Programs (`SUNWpppu`)

Files and Directories	**Description**
`/usr/bin/pppin`	Connection process for incoming calls
`/usr/sbin/pppls`	PPP login service daemon

Solstice PPP Daemon and User Programs (SUNWpppu)

Contains the PPP link manager daemon, and the Solstice PPP scripts and utilities.

Table 6-4 Solstice PPP Daemon and User Programs (SUNWpppu)

Files and Directories	Description
/usr/bin/pppconn	PPP connection program
/usr/bin/pppdisc	PPP disconnection program
/usr/bin/pppinit	PPP initialization script
/usr/bin/pppsetmod	Modem configuration program
/usr/bin/pppstat	Statistics collection utility
/usr/bin/ppptrace	Trace utility
/usr/sbin/pppd	Link manager daemon
/usr/sbin/ppptool	GUI for connect and disconnect

Solstice PPP Configuration Files (SUNWpppr)

Contains the initialization scripts and template configuration files for Solstice PPP.

Table 6-5 Solstice PPP Configuration Files (SUNWpppr)

Files and Directories	Description
/etc/opt/SUNWconn/ppp/modems	Modems database file
/etc/opt/SUNWconn/ppp/ppp.conf	Template PPP path configuration file
/etc/opt/SUNWconn/ppp/script/template	Template CHAT script
/etc/init.d/ppp	Initialization script for Solstice PPP
/etc/rc1.d/K48ppp	Symbolic link to /etc/init.d/ppp
/etc/rc2.d/S48ppp	Symbolic link to /etc/init.d/ppp
/etc/opt/SUNWconn/ppp/bitmaps	Icons for ppptool

*Solstice PPP Man Pages (*SUNWpppm*)*

Contains the product specific man pages for Solstice PPP.

Table 6-6 Solstice PPP Man Pages (SUNWpppm)

Files and Directories	Description
/usr/share/man/man4/ppp.conf	Man page for ppp.conf
/usr/share/man/man4/link.conf	Man page for link.conf
/usr/share/man/man4/modems	Man page for modem database
/usr/share/man/man1m/pppinit	Man page for pppinit
/usr/share/man/man1m/pppconn	Man page for pppconn
/usr/share/man/man1m/pppdisc	Man page for pppdisc
/usr/share/man/man1m/pppsetmod	Man page for pppsetmod
/usr/share/man/man1m/pppd	Man page for pppd
/usr/share/man/man1m/pppls	Man page for pppls
/usr/share/man/man1m/pppin	Man page for pppin
/usr/share/man/man1m/pppstat	Man page for pppstat
/usr/share/man/man1m/ppptrace	Man page for ppptrace
/usr/share/man/man7/ipd	Man page for PPP ptm IP interface
/usr/share/man/man7/ipdcm	Man page for PPP connection mgr
/usr/share/man/man7/ipdptp	Man page for PPP ptp IP interface
/usr/share/man/man7/ppp	Man page for PPP daemon

AnswerBook for Solstice PPP

The AnswerBook package SUNWabppp, which contains the on-line documentation for Solstice PPP can be installed on your local machine, on a server, or mounted from CD-ROM.

Other Files and Modifications

Solstice PPP modifies the file /etc/ttydefs when it is installed. These changes may be overwritten if you install another version of the operating system.

PPP Link Operation

A

This appendix provides a brief overview of PPP link operation, including a phase diagram and a description of the various PPP frames transmitted during the life of a PPP link.

PPP Phase Diagram

The process of establishing, maintaining, and terminating PPP links passes through several distinct phases, during which PPP frames are exchanged between the endpoints of the link.

- Link establishment phase
- Peer authentication phase (optional)
- Network layer protocol phase
- Link termination phase

Figure A-1 shows the transitions between these phases.

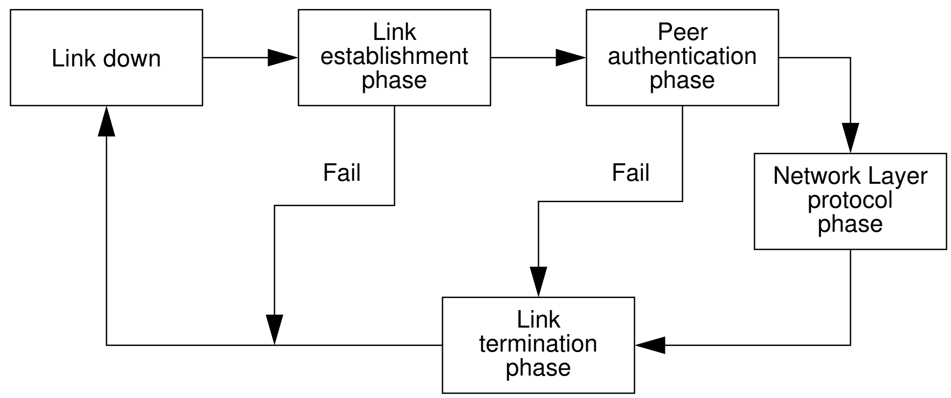


Figure A-1 PPP Link Operating Phases

Link Establishment Phase

The Link Control Protocol (LCP) is used to configure the PPP link during the link establishment phase. The configuration parameters are negotiated by an exchange of LCP frames. If the negotiation converges, the link is established and either the authentication protocol or network layer protocol phase is started. If the endpoints fail to negotiate a common configuration for the link, it is closed immediately.

During the link establishment phase only LCP frames should be transmitted in the PPP frames. All other frames are discarded.

Peer Authentication Phase (optional)

Optionally, one or both of the endpoints can request peer authentication. Solstice PPP supports peer authentication based on the Password Authentication Protocol (PAP) and the Challenge-Handshake Authentication Protocol (CHAP). If authentication is successful, the link remains up and the network layer protocol phase is started. If authentication fails, the link termination phase is started to close the link.

During the peer authentication phase only LCP and authentication frames should be transmitted in the PPP frames. All other frames are discarded.

Network Layer Protocol Phase

The Network Control Protocols (NCP) are used to configure the appropriate network layer. Solstice PPP implements the Internet Protocol Control Protocol (IPCP) to configure IP over PPP.

Once the network layer is configured by an exchange of NCP frames, IP datagrams can be encapsulated for transmission across the link. LCP frames may be exchanged periodically to test and maintain the link.

During the network layer protocol phase, LCP, NPC, and IP frames should be transmitted in the PPP frames. All other frames are discarded.

Link Termination Phase

The link may be terminated at any time, at the request of either endpoint. Termination may occur due to authentication failure, carrier loss, or the expiration of the inactivity timeout. Link control protocol (LCP) frames are exchanged to terminate the link.

During the link termination phase only LCP frames should be transmitted in the PPP frames. All other frames are discarded.

PPP Frames

PPP frames encapsulate packets of information that contain either configuration information or data. The PPP frames have the general format shown in Figure A-2:

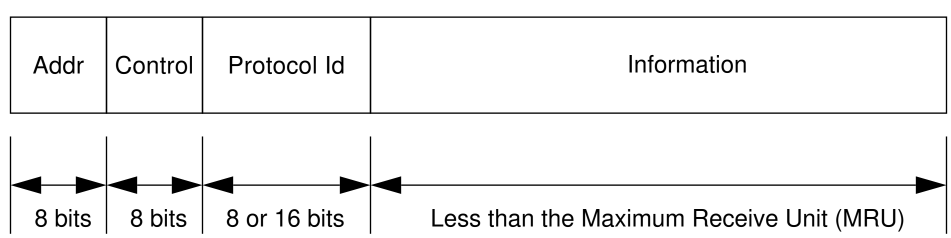


Figure A-2 PPP Frame Format

Address Field

The address field is one octet in length, and is part of the HDLC-like framing for PPP defined by RFC 1662. It is always set to `0xff`, which is the `All-Stations` address. Frames that contain any other address value are discarded silently.

Control Field

The control field is one octet in length, and is part of the HDLC-like framing for PPP defined by RFC 1662. It is always set to 0x03, which is the Unnumbered Information (`UI`) command with the `Poll/Final` bit set to zero. Frames that contain any other control value are silently discarded.

Protocol Id

The protocol id is one or two octets in length, and its value identifies the type of information contained in the information field of the frame.

The following values are significant for Solstice PPP:

`0x0001`	Padding protocol
`0x0021`	Internet protocol
`0x002d`	Van Jacobson Compressed TCP/IP
`0x002f`	Van Jacobson Uncompressed TCP/IP
`0x8021`	Internet Protocol Control Protocol
`0xc021`	Link Control Protocol
`0xc023`	Password Authentication Protocol
`0xc223`	Challenge Handshake Authentication Protocol

Information Field

The Information field is zero or more octets. The maximum length of the information field is determined by the maximum receive unit (MRU), which is a negotiated parameter. It is set to 1500 (for Ethernet networks) by default.

The information field can contain configuration information or data. In the case of Solstice PPP, the data represents compressed or uncompressed IP datagrams.

Link Control Protocol (LCP) Frames

The link control protocol (LCP) frames are transmitted during the link establishment and termination phases, and periodically during the life of the link. They are used to negotiate the configuration of the PPP link, and to test and maintain the link, once it is established. LCP frames have the general form shown in Figure A-3.

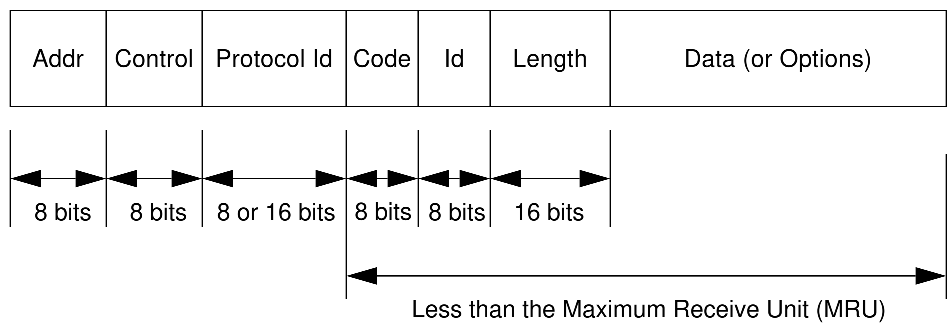


Figure A-3 LCP Frame Format

Address Field

The address field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0xff`.

Control Field

The control field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0x03`.

Protocol Id

The protocol id identifies the type of information contained in the information field of the frame, and is always `0xc021` for LCP frames.

Code Field

The code field is one octet in length and identifies the type of LCP frame, based on the following codes:

`0x01`	`Configure-request`	`0x07`	`Code-reject`
`0x02`	`Configure-ack`	`0x08`	`Protocol-reject`
`0x03`	`Configure-nak`	`0x09`	`Echo-request`
`0x04`	`Configure-reject`	`0x0a`	`Echo-reply`
`0x05`	`Terminate-request`	`0x0b`	`Discard-request`
`0x06`	`Terminate-ack`		

Id Field

The id field is one octet in length, and carries an identifier that is used to match associated requests and replies.

Length Field

The length field is two octets in length, and indicates the total length of the LCP frame including the Code, Id, length, and data fields. The length must not exceed the maximum receive unit (MRU).

Data Field

The data field is zero or more octets in length, as indicated by the length field. It contains the information associated with the frame, which may be configuration options, frame information, or simple data.

In some cases, the data field includes a *magic number*, which is four octets in length, and which is used to detect a looped-back condition and other link level anomalies. The magic number is a negotiated parameter, and is set to zero by default. When set, it contains a unique, randomly-generated sequence of digits that identifies the initial source of the frame. If two frames of the same type with the same magic number are received, there is a high probability that the link is looped back, and that the second frame is a reflection of the original.

Link Configuration Frames

Link configuration frames are transmitted during the link establishment phase. The data field of a link configuration frame carries information used to negotiate the configuration options for the link. The Link configuration frames are:

- `Configure-request`
  Code `0x01`. Request the establishment of a link with a particular configuration. Represents the start of the link establishment phase. `Terminate-request` frames are transmitted periodically until either a valid response is received, or the number of frames sent exceeds the value of the `lcp_max_restart` parameter in the file `ppp.conf`.

- `Configure-ack`
  Code `0x02`. Acknowledge the receipt of a recognizable `Configure-request` frame, and accept the requested configuration. Represents the end of the link establishment phase.

- `Configure-nak`
  Code `0x03`. Acknowledge the receipt of a recognizable `Configure-request` frame, but reject some or all of the requested configuration.

- `Configure-reject`
  Code `0x03`. Reject a `Configure-request` frame because it is not recognizable or because the requested configuration is not acceptable.

Link Termination Frames

Link termination frames are transmitted during the link termination phase. The link termination frames are:

- `Terminate-request`
  Code `0x05`. Request the termination of a link. Represents the start of the link termination phase.

- `Terminate-ack`
  Code `0x06`. Acknowledge the receipt of a recognizable `Terminate-request` frame, and accept the termination request. Represents the end of the link termination phase.

Link Maintenance Frames

Link maintenance frames are transmitted periodically to test and maintain the link. The link termination frames are:

- `Code-reject`
  Code `0x07`. Rejects an LCP frame that has an invalid code field.
- `Protocol-reject`
  Code `0x08`. Rejects a PPP frame that has an invalid protocol id.
- `Echo-request`
  Code `0x09`. Requests a response, in the form of an `Echo-reply` frame, from the remote end-point. Used to test that the link is still up.
- `Echo-reply`
  Code `0x10`. Responds to a valid `Echo-request` frame. Used to test that the link is still up.
- `Discard-request`
  Code `0x11`. Sends a frame which is silently discarded at the remote endpoint. Used as a debugging mechanism.

Password Authentication Protocol (PAP) Frames

PAP frames are exchanged during the peer authentication phase, when peer authentication based on the Password Authentication Protocol (PAP) is requested as one of the configuration options during the link establishment phase. They have the general form shown in Figure A-4.

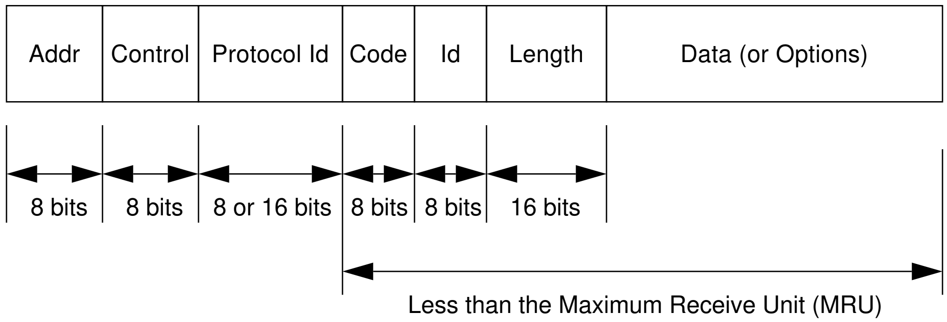


Figure A-4 PAP Frame Format

Address Field

The address field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0xff`.

Control Field

The control field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0x03`.

Protocol Id

The protocol id identifies the type of information contained in the information field of the frame, and is always `0xc023` for PAP frames.

Code Field

The code field is one octet in length and identifies the type of PAP frame, based on the following codes:

```
0x01    Authenticate-request

0x02    Authenticate-ack

0x03    Authenticate-nak
```

Id Field

The id field is one octet in length, and carries an identifier that is used to match associated requests and replies.

Length Field

The length field is two octets in length, and indicates the total length of the PAP frame including the code, id, length, and data fields. The length must not exceed the maximum receive unit (MRU).

Data Field

The data field is zero or more octets in length, as indicated by the length field. It contains information associated with the authentication negotiation, in a format determined by the code field.

PAP `Authenticate-request` Frames

PAP `Authenticate-request` frames (code `0x01`) are transmitted to start the authentication phase, and contain the PAP id and PAP password sent for authentication. Up to ten PAP `Authenticate-request` frames are transmitted without receiving a response before the authentication phase fails. PAP `Authenticate-request` frames have the format shown in Figure A-5:

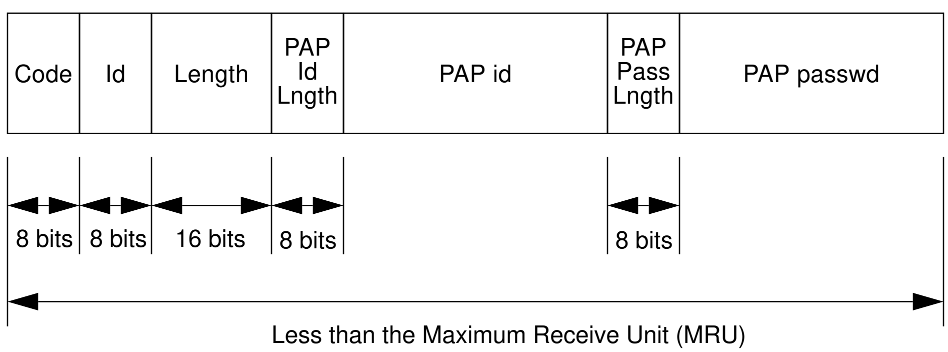


Figure A-5 PAP Request Frame Format

The PAP id is zero or more octets in length, as indicated by the PAP id length field, and contains the character string specified by the `send_pap_id` parameter in the file `ppp.conf`.

The PAP password is zero or more octets in length, as indicated by the PAP password length field, and contains the character string specified by the `send_pap_passwd` parameter in the file `ppp.conf`.

PAP `Authenticate-ack` *and* `Authenticate-nak` *Frames*

A PAP `Authenticate-ack` frame (code `0x02`) is transmitted by the authenticator when it receives a recognizable PAP `Authenticate-request` frame that contains an acceptable PAP id and PAP password.

A PAP `Authenticate-nak` frame (code `0x03`) is transmitted by the authenticator when it receives a PAP `Authenticate-request` frame that is not recognizable, or that contains an unacceptable PAP id and PAP password pair. The link is always terminated.

Challenge-Handshake Authentication Protocol (CHAP) Frames

CHAP frames are exchanged during the peer authentication phase, when peer authentication based on the Challenge-Handshake Authentication Protocol (CHAP) is requested as one of the configuration options during the link establishment phase. They have the general form shown in Figure A-6.

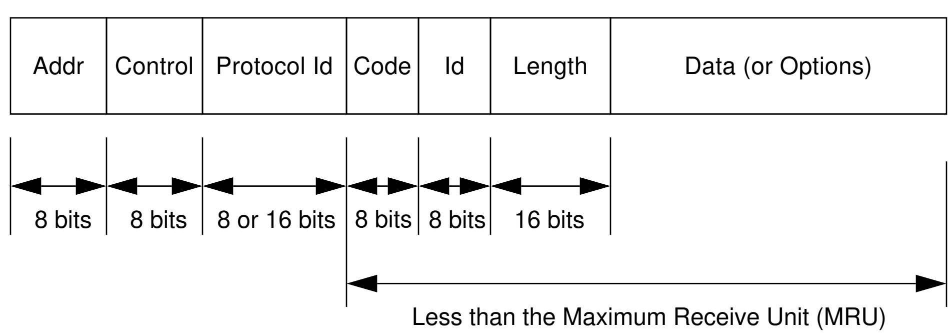


Figure A-6 CHAP Frame Format

Address Field

The address field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0xff`.

Control Field

The control field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0x03`.

Protocol Id

The protocol id identifies the type of information contained in the information field of the frame, and is always `0xc223` for CHAP frames.

Code Field

The code field is one octet in length and identifies the type of CHAP frame, based on the following codes:

```
0x01   Challenge

0x02   Response

0x03   Success

0x04   Failure
```

Id Field

The id field is one octet in length, and carries an identifier that is used to match associated requests and replies.

Length Field

The length field is two octets in length, and indicates the total length of the CHAP frame including the code, id, length, and data fields. The length must not exceed the maximum receive unit (MRU).

Data Field

The data field is zero or more octets in length, as indicated by the length field. It contains information associated with the authentication negotiation, in a format determined by the code field.

CHAP `Challenge` *and* `Response` *Frames*

CHAP `Challenge` frames (code `0x01`) are used to start the authentication negotiation, and are transmitted by the authenticator. They contain the CHAP name and a challenge value, which is calculated from the CHAP secret using a one-way hash algorithm. Up to ten CHAP `Challenge` frames are sent without receiving a `Response` frame before the authentication phase fails.

A CHAP `Response` frame (code `0x02`) is sent on receipt of a recognized CHAP `Challenge` frame. It contains a response value, which is calculated using the CHAP secret, the challenge value received, and the same one-way hash algorithm.

CHAP `Challenge` and `Response` frames have the format shown in Figure A-7:

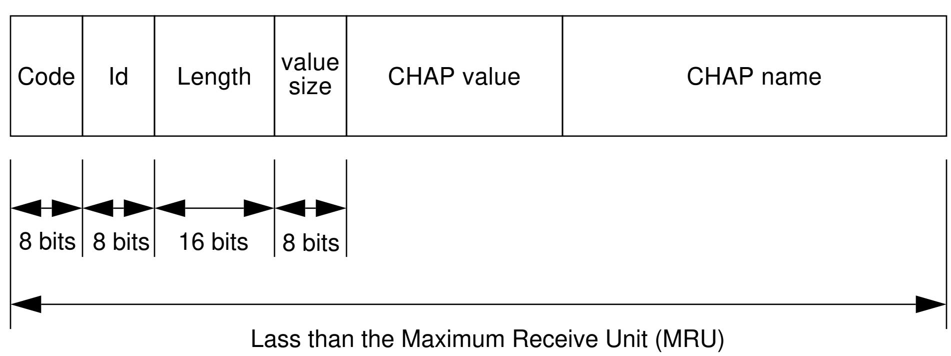


Figure A-7 CHAP `Challenge` and `Response` Frame Format

The CHAP name is one or more octets in length and contains the character string specified by the `send_chap_name` parameter in the file `ppp.conf`.

CHAP `Success` *and* `Failure` *Frames*

A CHAP `Success` frame (code `0x03`) is transmitted by the authenticator when it receives a recognizable CHAP `response` frame that contains an acceptable CHAP name and response value.

A CHAP `Failure` frame (code `0x04`) is transmitted by the authenticator when it receives a CHAP `response` frame that is not recognizable, or that contains an unacceptable PAP id and PAP password pair. The link is always terminated.

Internet Protocol Control Protocol (IPCP) Frames

The Internet Protocol Control Protocol (IPCP) is used to configure, enable, and disable IP over PPP links. It uses the same frame exchange mechanism as the link control protocol (LCP).

IPCP frames have the general form shown in Figure A-8 on page 170.

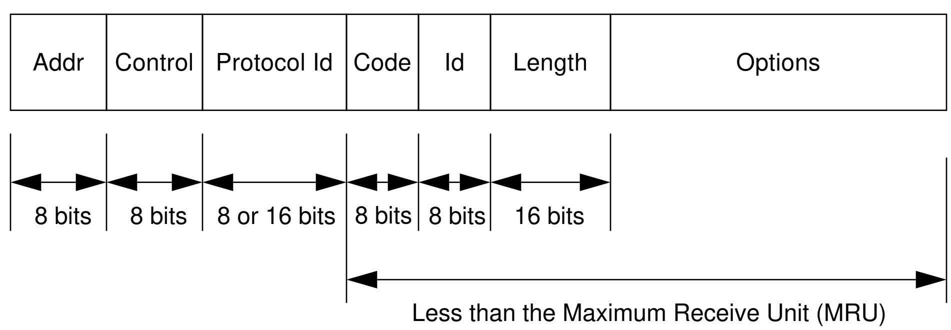


Figure A-8 IPCP Frame Format

Address Field

The address field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0xff`.

Control Field

The control field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0x03`.

Protocol Id

The protocol id identifies the type of information contained in the information field of the frame, and is always `0x8021` for IPCP frames.

Code Field

The code field is one octet in length and identifies the type of IPCP frame, based on the following codes:

`0x01`	`Configure-request`	`0x05`	`Terminate-request`
`0x02`	`Configure-ack`	`0x06`	`Terminate-ack`
`0x03`	`Configure-nak`	`0x07`	`Code-reject`
`0x04`	`Configure-reject`		

Id Field

The id field is one octet in length, and carries an identifier that is used to match associated requests and replies.

Length Field

The length field is two octets in length, and indicates the total length of the IPCP frame including the code, id, length, and data fields. The length must not exceed the maximum receive unit (MRU).

Data Field

The data field is zero or more octets in length, as indicated by the length field. It contains the information associated with the frame, which may be configuration options, frame information, or simple data, in a format determined by the code field.

Internet Protocol (IP) Frames

Internet Protocol (IP) frames contain encapsulated IP datagrams for transmission across PPP links. A single IP datagram is contained in the information field of a PPP frame. Large IP datagrams may be fragmented for transmission, if necessary. IP frames have the general form shown in Figure A-9.

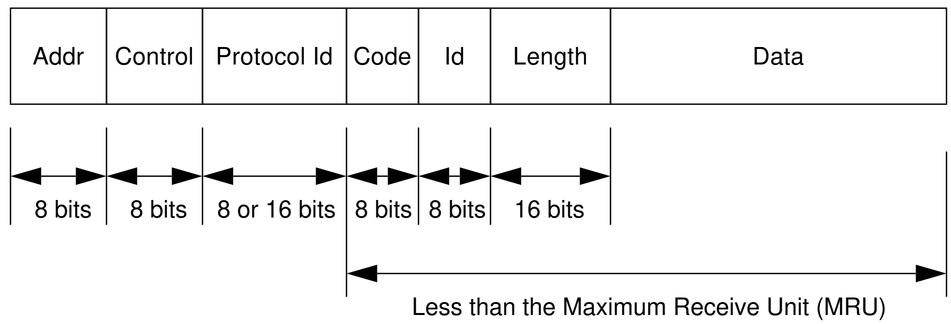


Figure A-9 IP Frame Format

Address Field

The address field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0xff`.

Control Field

The control field is one octet in length, and is part of the HDLC-like framing for PPP. It is always set to `0x03`.

Protocol Id

The protocol id identifies the type of information contained in the information field of the frame. It can have the following values for IP frames:

`0x0021`	Internet protocol
`0x002d`	Van Jacobson Compressed TCP/IP
`0x002f`	Van Jacobson Uncompressed TCP/IP

Id Field

The id field is one octet in length, and carries an identifier that is used to match associated requests and replies.

Length Field

The length field is two octets in length, and indicates the total length of the IPCP frame including the code, id, length, and data fields. The length must not exceed the maximum receive unit (MRU).

Data Field

The data field is zero or more octets in length, as indicated by the length field. It contains the IP datagram for transmission over the PPP link.

Modem and Null Modem Cables B

This appendix provides the pinouts of cables that can be used for most modem and null modem configurations. Refer to the manufacturer's documentation for a detailed description of special cables you may need to connect your particular modem.

Standard EIA-232-E Modem Cables

Use the cable shown in Figure B-1 to connect a standard EIA-232-E serial port a synchronous modem:

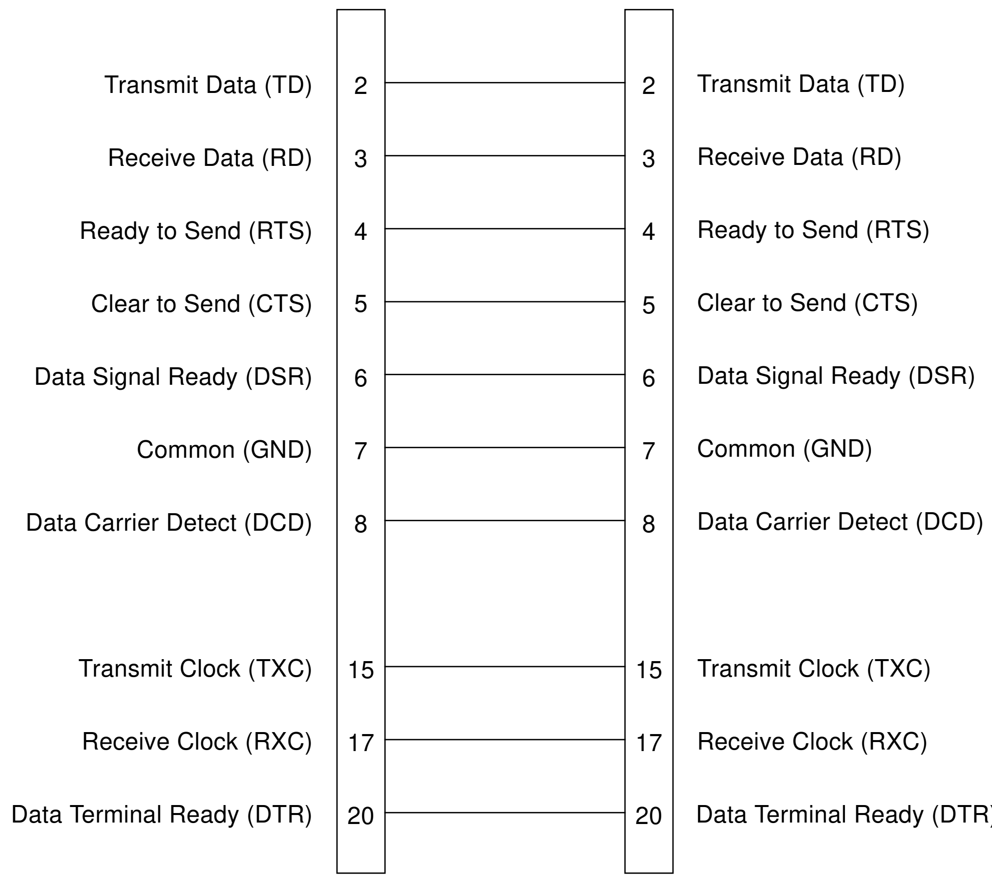


Figure B-1 Synchronous EIA-232-E Modem Cable

Use the cable in Figure B-2 to connect a standard EIA-232-E serial port to an asynchronous modem:

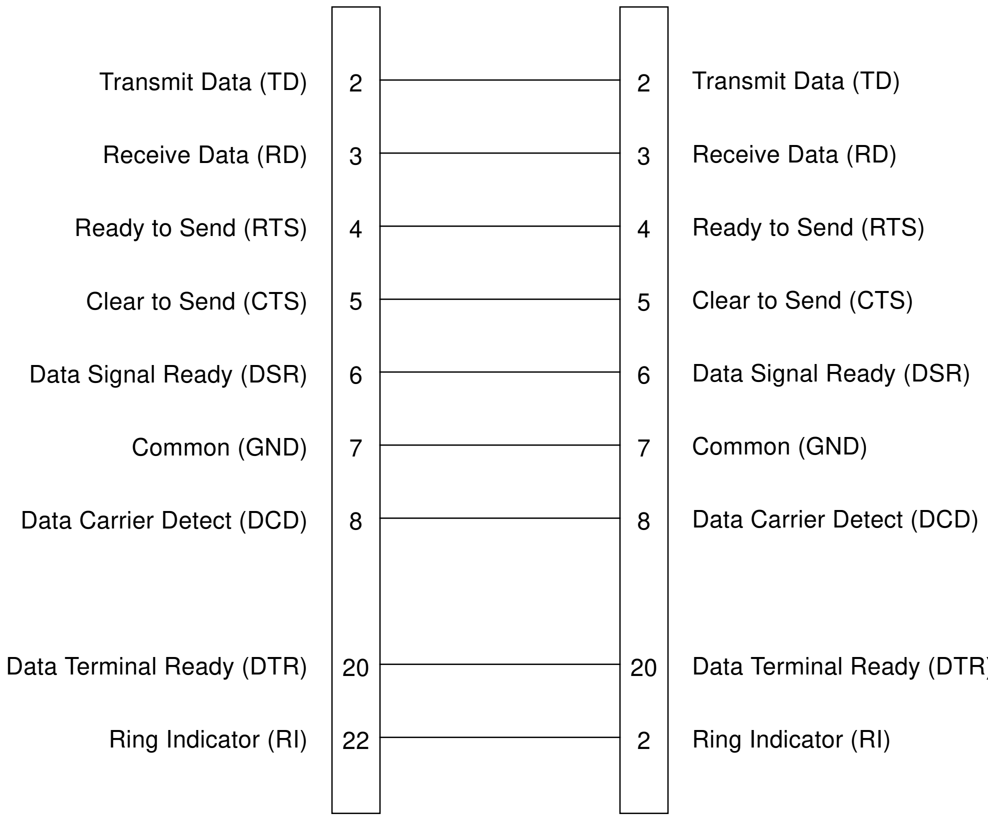


Figure B-2 Asynchronous EIA-232-E Modem Cable

IPC/IPX Adapter Cables

Figure B-3 and Figure B-4 show the pinouts for a cables used to convert the 8-pin serial port on a SPARCstation IPC or IPX to an EIA-232-E serial port.

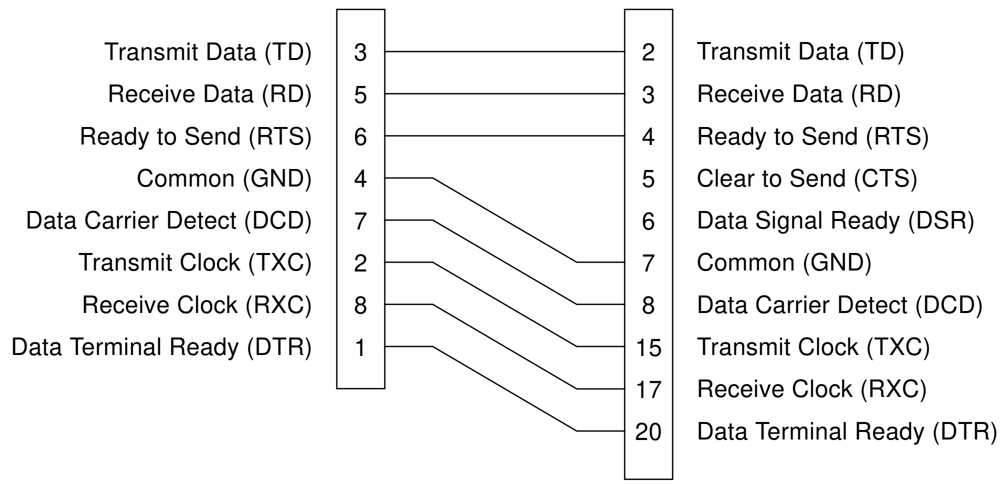


Figure B-3 Synchronous IPC/IPX Adapter Cable

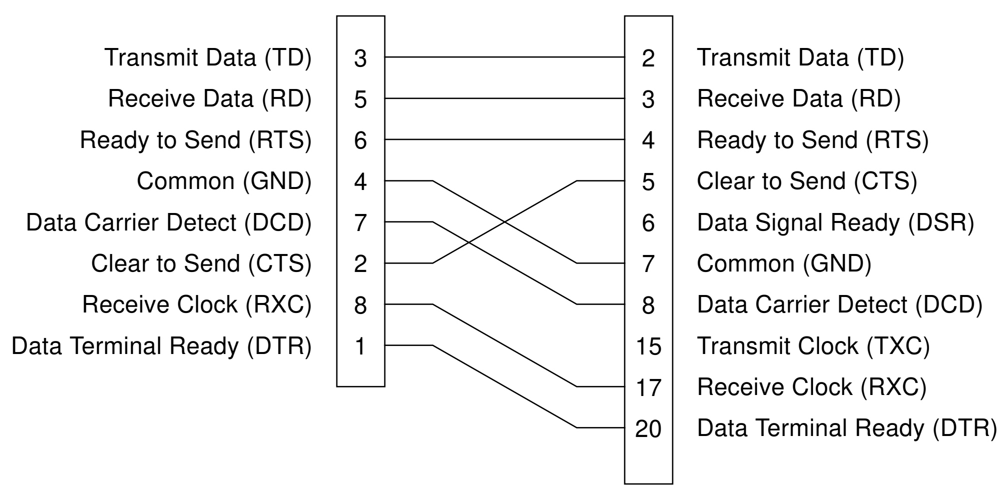


Figure B-4 Asynchronous IPC/IPX Adapter Cable

Asynchronous EIA-232-E Null Modem

Use the cable shown in Figure B-5 to create a direct asynchronous connection between two hosts:

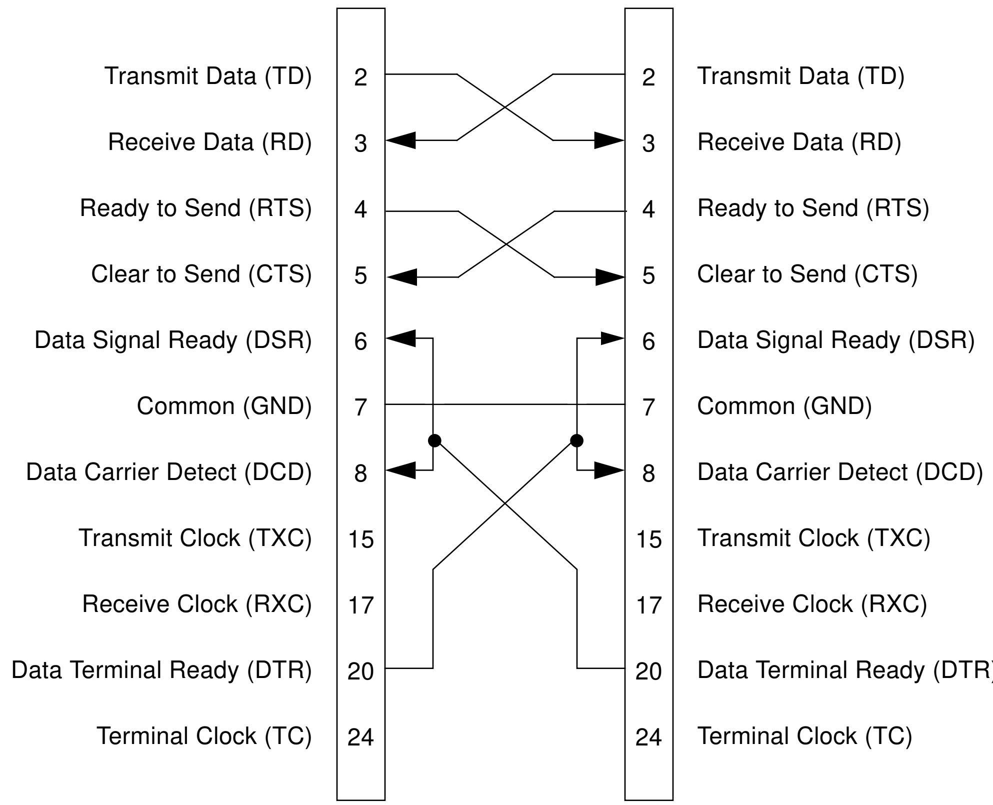


Figure B-5 Asynchronous EIA-232-E Null Modem

Synchronous EIA-232-E Null Modems

Use the cable shown in Figure B-6 to create a direct synchronous connection between two hosts when both sides provide a clock signal:

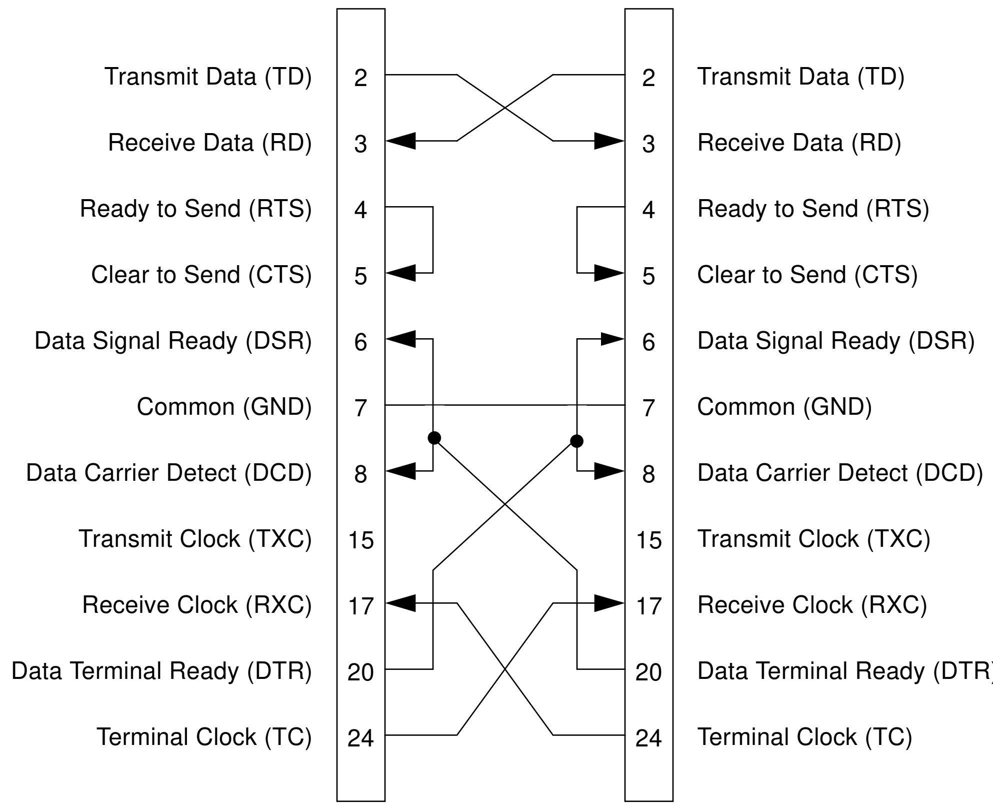


Figure B-6 Synchronous EIA-232 Null Modem (both sides provide clock)

Use the cable shown in Figure B-7 to create a direct synchronous connection between two hosts when only one side provides a clock signal:

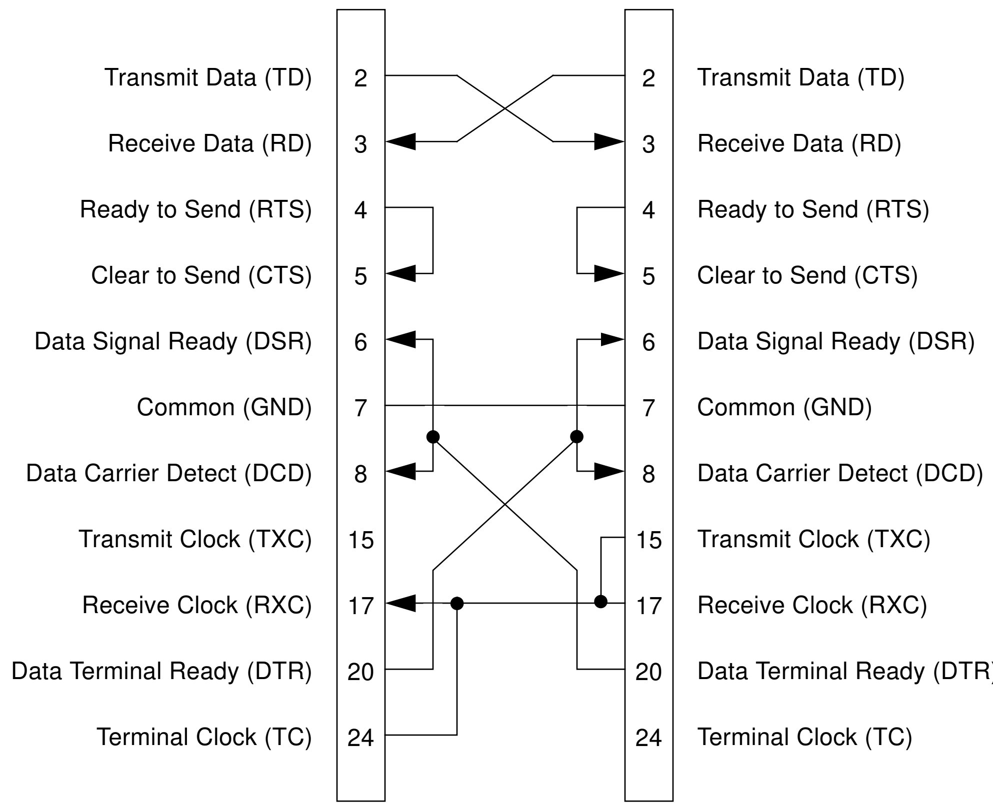


Figure B-7 Synchronous EIA-232 Null Modem (one side supplies clock)

Synchronous EIA-449 Null Modems

The high-speed serial interface for Sbus (HSI/S) has a 37-way EIA-449 port. This type of port uses balanced transmission to provide improved speed and distance characteristics; therefore, there are two pins for each signal.

When both sides of the connection provide a clock signal, pins 17 and 35 (Terminal Timing) and pins 8 and 26 (Receive Timing) are crossed-over, as shown in Figure B-8.

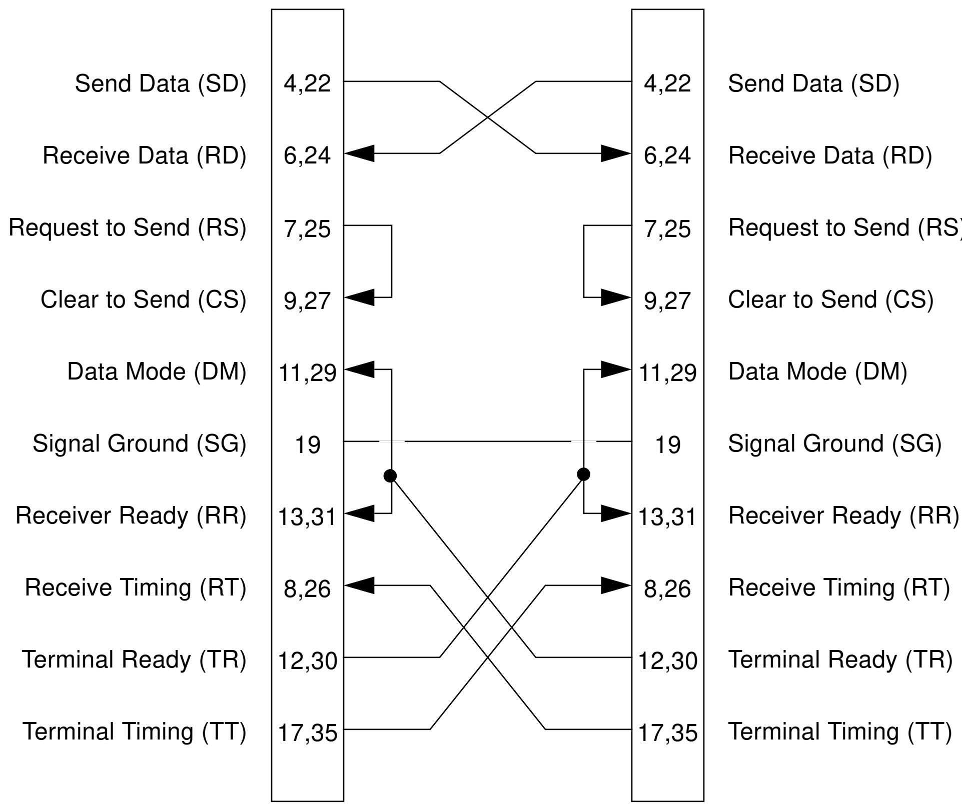


Figure B-8 Synchronous EIA-449 Null Modem (both sides supply clock)

When one side of the connection provides a clock signal, pins 17 and 35 (Terminal Timing) on one side are connected to pins 8 and 26 (Receive Timing) on both sides, as shown in Figure B-9.

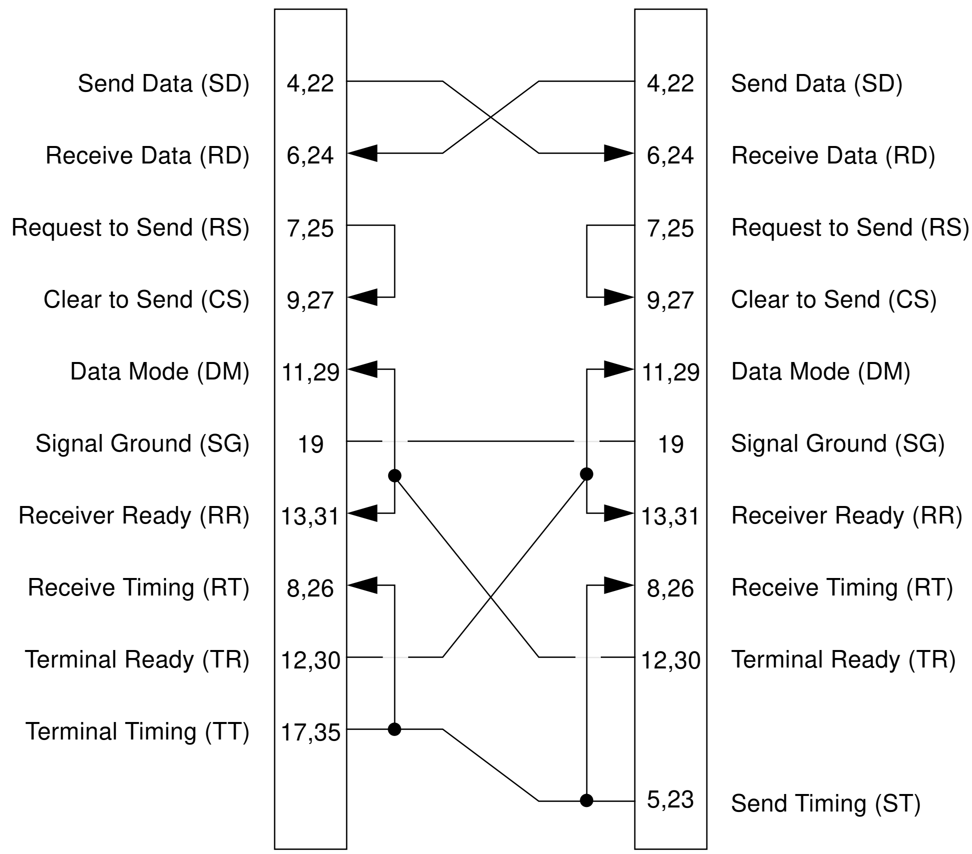


Figure B-9 Synchronous EIA-449 Null Modem (one side supplies clock)

X.21 to EIA-449 Converter

Use the cable shown in Figure B-10 to convert between standard X.21 and EIA-449 interfaces:

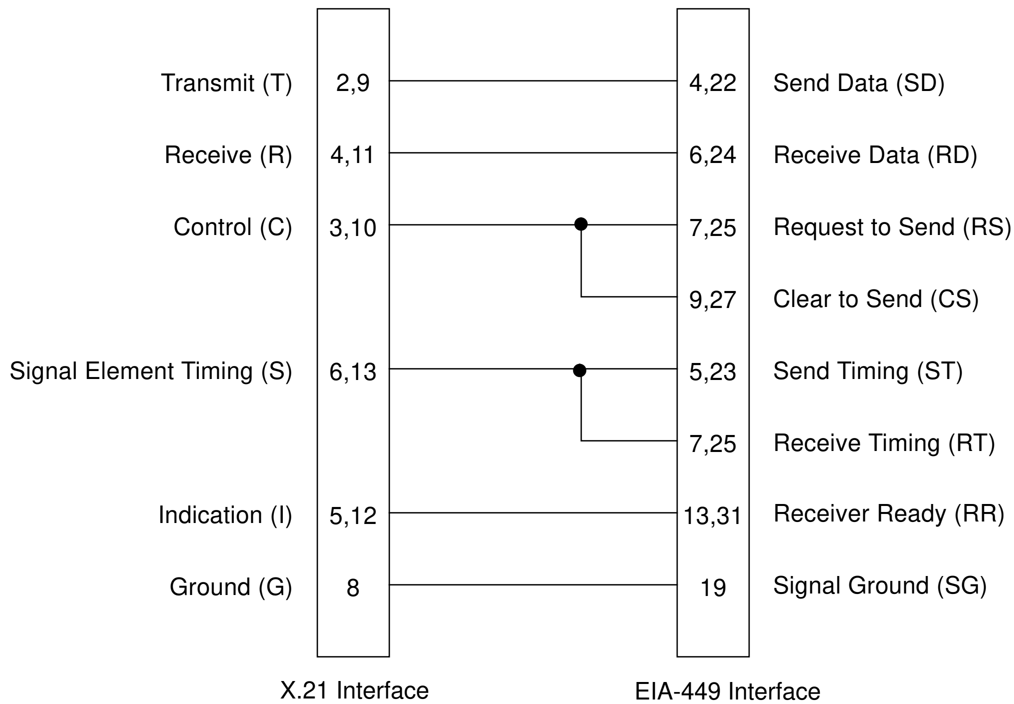


Figure B-10 X.21 to EIA-449 Converter

Index

D

L

M

N

T

U

V