NAME

aspppd, asppls – Asynchronous PPP Link Manager

SYNOPSIS

/usr/sbin/aspppd [–**d** *debug-level*]
/usr/sbin/aspppls

AVAILABILITY

SUNWapppu

DESCRIPTION

aspppd is the link manager for the asynchronous data link protocol specified in RFC1331, *The Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links*. It is a user level daemon that works in concert with the IP-Dialup driver (**ipdcm**) and PPP streams module (**ppp**(7M)) to provide IP network services over an analog modem using dialed voice grade telephone lines. The link manager automates the process of connecting to a peer (remote) host when PPP service with that host is required. The connection process can be initiated either by sending an IP datagram to a (disconnected) peer host or by receiving a notification that a peer host desires to establish a connection.

aspppls is the *login service* that connects the peer host machine to **aspppd**. **aspppls** is invoked by the serial port monitor when a peer machine logs into a PPP enabled account. Its purpose is to cause the link manager to accept the incoming call.

OPTIONS

–**d** *debug-level* The *debug-level* is a number between 0 and 9. Higher numbers give more detailed debugging information. The output is written to the log file **/etc/log/asppp.log**.

USAGE

The link manager is invoked at boot time if the configuration file **/etc/asppp.cf** is present. After parsing the configuration file and building a *path* object for each peer host, it sleeps until (1) an IP datagram is routed to one of the **ipd** or **ipdptp** interfaces (see **ppp**(7M)), or (2) it is notified by the login service that a peer host is attempting to make a connection.

In the first case, it consults the UUCP database, dials the modem, logs into the peer host, establishes the PPP data link, brings up IP, and forwards the IP datagram that initiated the process.

In the second case, the link manager opens the file descriptor supplied by the login service, establishes the PPP data link, and brings up IP.

If the link manager determines that there has been no IP traffic for the period specified by the **inactivity_timeout** keyword, it disconnects the link by bringing down IP and PPP and closing the connection with the peer host.

The link manager can be reinitialized by sending it the -HUP signal (with **kill**(1) for example), which causes it to disconnect all open PPP links and reread the configuration file.

Path

A *path* is an object that contains the state of a connection with a peer host. Information such as system names, interface names, timeout values, and other attributes are kept in the path object. There exists a path for each potential peer host. Paths are defined in the configuration file.

Interfaces

The link manager supports two types of IP layer interfaces; the point-to-multipoint interface (**ipd**) and the point-to-point interface (**ipdptp**) (see **ppp**(7M)).

The point-to-multipoint interface logically connects the host machine to a network containing one or more peer hosts. IP traffic to or from any of the peer hosts is routed through the point-to-multipoint interface. When an **ipd** interface is configured, only one IP address, that of the host, is assigned. In other words, it behaves very similarly to an Ethernet interface, although the broadcast capability is not supported. This type of interface is well suited for a dial in PPP server.

The point-to-point interface logically connects the host machine with one peer host. Only IP traffic to or from the peer host is routed through this interface. When an **ipdptp** interface is configured, two IP addresses are assigned. This type of interface is well suited to support a remote, or nomadic, machine.

An interface must be fully configured and enabled (i.e. up) before an IP datagram will be routed to it. It's also true that a point-to-multipoint interface must be fully configured and enabled before the link manager will associate an incoming connection with it. It's not necessary, however, for a point-to-point interface to be configured and enabled before an incoming connection will be assigned to it. A point-to-point interface that is "plumbed", but otherwise not configured or enabled (i.e. down), can be used to accept an incoming connection if the path associated with the potential connection contains a dynamic interface specification (e.g. **interface ipdptp***). In this case the link manager will select a disabled (down) interface, configure the host and peer addresses, bring it up, and assign it for the duration of the connection.

Routing

Special attention should paid to routing issues that may arise if a host has more than one interface configured and enabled. By definition, a host with more then one enabled interface is a *router*, and the routing daemon (typically **in.routed**) will advertise the routes provided by the PPP interfaces. This is normally acceptable behavior for a dial in server, but can cause network disruptions if not administered properly.

To prevent routing information packets (RIP) from flowing over point-to-point interfaces, specify the **norip** keyword followed by the interface name in the **/etc/gateways** file. These entries, for example, prevent RIP from being sent over ipdptp0 and ipdptp1:

```
norip   ipdptp0
norip   ipdptp1
```

See **in.routed**(1M) for further information.

Authentication

The link manager can be configured to support either the Password Authentication Protocol (PAP) or the Challenge Handshake Authentication Protocol (CHAP) as specified in RFC1334. Both protocols can be configured simultaneously, in which case, CHAP has precedence. A single host may participate as an authenticator (the local host requests that the peer host authenticate itself) or an authenticatee (the local host has been asked by the peer host to authenticate itself) or as both. It is also possible for a host to be an authenticator for one protocol and an authenticatee for the other protocol.

PAP is a simple protocol similar to a standard login/password type of authentication. The PAP authenticator sends a message to its peer requesting that the peer authenticate itself. The peer responds with an authenticate request packet that contains an id and a password (both in plaintext). The id and password are matched against a local copy, and if they match, the connection is established. If they don't match, the connection is dropped.

CHAP does not pass any plaintext authentication data across the link. The CHAP authenticator sends a challenge packet to the peer that contains a random string. The peer then takes the string in the challenge packet and computes a response string that is a function of the challenge string and a shared secret key. The peer then sends a response packet back to the authenticator. The authenticator computes a string based on the original challenge string and the shared secret key and matches that result with the received response. If they match, the connection is established. Otherwise the connection is dropped.

Configuration File

The primary purpose of the **/etc/asppp.cf** configuration file is to define each path used by the link manager to establish and maintain communication with a peer system.

The file consists of a sequence of tokens separated by white space (blanks, tabs, and new lines). There are no record boundaries or any other constraints on the placement of the tokens. If a token begins with a pound sign (#), all characters between the pound sign and the next newline (\n) are ignored (i.e. they are treated as a comment). Alphanumeric tokens are case insensitive and are translated by the lexical analyzer into lower case before further processing.

A *string* is a single token that does not contain embedded white space. The standard ANSI C \ escape sequence may be used to embed special characters (see an ANSI C manual for a list of escaped special characters). Use \s for the space character. If a pound sign appears at the beginning of a *string*, it must be escaped (\#) to avoid interpretation as a comment. A NULL (\0) will truncate the *string*.

Groups of tokens are assembled into units known as *paths* (essentially a human-readable form of the path object). A path begins with the keyword **path** and ends at the token found before any subsequent **path** (or **defaults**) keyword or at the last token in the file. The tokens comprising a path are further partitioned into small groups consisting mostly of keyword/value pairs that define the attributes of the current path. If a particular keyword/value pair is not listed for a path, the default value is assumed.

The token sequences that begin with the substrings **ipcp_** or **lcp_** refer to PPP initial configuration options as specified in RFC1332, *The PPP Internet Protocol Control Protocol (IPCP)*. See the RFC for a more complete definition of these options.

The following is an alphabetic list of the token sequences that can be contained in a configuration file. Required sequences are noted.

Keywords

chap_name *string*

One or more octets representing the identification of this host. The name should not be NUL or CR/LF terminated. The name is sent to the authenticator in a response packet. Place this key/value pair in the authenticatee's configuration file.

chap_peer_secret *string*
One or more octets, preferably at least sixteen, that contain the secret key that is used with the challenge value to generate the string to match with the response received from the peer. Place this key/value pair in the authenticator's configuration file.

chap_peer_name *string*
One or more octets representing the identification of the peer transmitting the packet. The name should not be NUL or CR/LF terminated. The name is received from the peer in a response packet. Place this key/value pair in the authenticator's configuration file.

chap_secret *string*
One or more octets, preferably at least sixteen, that contain the secret key that is used with the received challenge value to generate the response sent to the authenticator. Place this key/value pair in the authenticatee's configuration file.

debug_level *number*
number is between 0 and 9. Higher numbers give more detailed debugging information as shown in the table below. The output is written to the **/etc/log/asppp.log** file. The value set by the **debug_level** keyword overrides the **-d** command line option.

level	meaning
0	errors only
1	minimal information
4	some uucp chat-script info
5	all uucp chat-script info
7	maximum uucp info
8	PPP message traces
9	Raw IP packets

defaults Indicates that all following token sequences up the next **path** keyword, or the end of file, set default attributes that affect subsequently defined paths.

default_route
When the IP layer corresponding to the current path is fully operational, add the peer IP address to the route table as the default destination. The route is removed when the IP layer is brought down. Note: the **default_route** keyword is only installed by point-to-point interfaces.

ifconfig *parameters*
(Required) The **ifconfig** keyword and associated *parameters* are passed to the shell for evaluation and execution. It's used to define an interface. See the **ifconfig**(1M) man page for more information.

inactivity_timeout *seconds*
seconds is the maximum number of seconds that the connection associated

with the current path can remain idle before it is terminated. **0** may be specified to indicate no timeout. The default is 120 seconds.

interface (**ipd***n* | **ipdptp***n* | **ipdptp**∗)
: (Required) Associates a specific point-to-multipoint or point-to-point interface as denoted by the non-negative integer *n* with the current path. The third form, **ipdptp**∗, indicates that the interface associated with the path is a dynamic interface that will be selected at connect time from a pool of previously configured, inactive (down) point-to-point interfaces.

ipcp_async_map *hex-number*
: Specifies the async control character map for the current path. The *hex-number* is the natural (i.e. big endian) form representation of the four octets that comprise the map. The default value is ffffffff.

ipcp_compression (**vj** | **off**)
: Indicates whether IP compression is enabled or not. If enabled (**vj**), the Van Jacobson compression algorithm is used. The default is compression (**vj**).

lcp_compression (**on** | **off**)
: Indicates whether PPP address, control, and protocol field compression is enabled or not. If enabled, both the address and control field compression and the protocol field compression options are set. The default is compression (**on**).

lcp_mru *number*
: *number* specifies a desired maximum receive unit packet size in octets. The default is 1500.

negotiate_address (**on** | **off**)
: Indicates whether or not local IP address assignment is obtained through negotiation and assigned dynamically. If enabled, the local address will be obtained from the remote end of the PPP link. If so obtained, any local address other than 0.0.0.0 can be used to initially configure the interface. The default is to negotiate (**on**).

pap_id *string*
: One or more octets that represent the name of the host which will be sent to the authenticator. To indicate a zero length string, do not include the keyword. Place this key/value pair in the authenticatee's configuration file.

pap_password *string*
: One or more octets that indicate the password for this host which will be sent to the authenticator. To indicate a zero length string, do not include the keyword. Place this key/value pair in the authenticatee's configuration file.

pap_peer_id *string*
: One or more octets that indicate the name of the peer to be authenticated. To indicate a zero length string, do not include the keyword. Place this key/value pair in the authenticator's configuration file.

pap_peer_password *string*

One or more octets that indicate the password to be used for authentication. To indicate a zero length string, do not include the keyword. Place this key/value pair in the authenticator's configuration file.

path (Required) Indicates that all following token sequences are to be grouped together as attributes of this (current) path. The collection of attributes comprising the current path are terminated by the occurrence of a subsequent **path** or **defaults** keyword or by the end of file.

peer_ip_address *IP-address*
(Required for point-to-multipoint paths) Associates the *IP-address* with the current path. The value is ignored if the path specifies a point-to-point interface. The *IP-address* may be in "dotted decimal", hexadecimal, or symbolic (i.e. hostname) format.

peer_system_name *name*
(Required) Associates the peer system *name* with the current path. The name is used to look up modem and peer specific information for outbound connections in the UUCP **/etc/uucp/Systems** file. For incoming connections, the appropriate path is determined by matching *name* with the login name that was used to obtain the connection (i.e. an entry in the **/etc/passwd** file specifies *name* in the username field).

require_authentication (**off** | **pap** [**chap**] | **chap** [**pap**])
Indicates that the local host is the authenticator, and that the peer is required to authenticate itself. If either **pap** or **chap** is present, the peer must participate in the authentication protocol or the connection will be terminated. If both **pap** and **chap** are present, then the local host will first try to negotiate **chap**, and if that fails, it will try to negotiate **pap**. If both fail, the connection will be terminated. The default does not require authentication (**off**).

If **pap** is required, then the **pap_peer_id** and **pap_peer_password** keywords and values should be specified for the associated path. If they are not specified, the corresponding values are set to the NULL string. If **chap** is required then the **chap_peer_name** and **chap_peer_secret** keywords and values must be specified for the associated path.

version *n* Specifies that the contents of the configuration file correspond to format version *n*. If this keyword is present, it must be the first keyword in the file. If absent, the version is assumed to be 1. This document contains the definition of the version 1 format for the configuration file.

will_do_authentication (**off** | **pap** [**chap**] | **chap** [**pap**])
Indicates that the local host is a potential authenticatee and is willing to participate in the specified authentication protocol. If both **pap** and **chap** are present then the local host is willing to participate in either authentication protocol. The default does not participate in authentication (**off**).

If **pap** is available, then the **pap_id** and **pap_password** keywords and values should be specified for the associated path. If they are not specified, the corresponding values are set to the NULL string. If **chap** is available then the

chap_name and **chap_secret** keywords and values must be specified for the associated path.

EXAMPLES

The following are examples of typical configuration files.

Remote Machine

In this example, the remote machine is most likely a nomadic or home machine with a single modem.

```
#
# Dial in to two servers
#
ifconfig ipdptp0 plumb nomad1 dialin1 private up
path
      interface ipdptp0
      peer_system_name Pdialin1
      will_do_authentication pap
      pap_id nomad1
      pap_password secret

ifconfig ipdptp1 plumb nomad1 dialin2 private up
path
      interface ipdptp1
      peer_system_name Pdialin2
      lcp_mru 1006
```

Dial In Servers

This example shows a dial in server supporting a point-to-multipoint interface. There may be several modems attached to this server. Note that the network addressed by the **ipd** interface will be advertised by the router, and all traffic destined for that network will be routed through this host. For that reason, it is not wise to support multiple dial in servers with point-to-multipoint interfaces to the same network.

```
#
# A point-to-multipoint dial in server
#
ifconfig ipd0 plumb dialin1 netmask + up
defaults
      interface ipd0
      inactivity_timeout 900   # 15 minutes
      require_authentication chap pap
      chap_peer_name nomads
path
      peer_system_name Pnomad1
      pap_peer_id nomad1
      pap_peer_password secret
path
      peer_system_name Pnomad2
      chap_peer_secret a\sspace
```

```
path
    peer_system_name Pnomad3
    inactivity_timeout 0          # No timeout for this host
    chap_peer_secret \#123;.
path
    peer_system_name Pnomad4
    chap_peer_secret My\sSecret#Word
```

This is another dial in server that supports dynamic point-to-point interfaces. Usually the server has one modem for each interface. One advantage of using dynamic interfaces is that (host) routes will only be advertised when an interface is up. Therefore, multiple dial in servers can be supported.

```
#
# A dynamic point-to-point dial in server
#
ifconfig ipdptp0 plumb dialin2 down
ifconfig ipdptp1 plumb dialin2 down
ifconfig ipdptp2 plumb dialin2 down
defaults
    interface ipdptp*
    inactivity_timeout 900
    debug_level 5

path peer_system_name Pnomad1
path
    peer_system_name Pnomad2
    negotiate_address off
path peer_system_name Pnomad3
path peer_system_name Pnomad4
```

FILES

/etc/asppp.cf	configuration file
/etc/log/asppp.log	message log file
/etc/uucp/Devices	
/etc/uucp/Dialers	
/etc/uucp/Sysfiles	
/etc/uucp/Systems	
/tmp/.asppp.fifo	communication path between **aspppd** and **aspppls**
/usr/sbin/aspppd	link manager
/usr/sbin/aspppls	login service

SEE ALSO

ifconfig(1M), **in.routed**(1M), **ppp**(7M)

TCP/IP and Data Communications Guide