NAME

pppd − Point to Point Protocol daemon

SYNOPSIS

pppd [*tty_name*] [*speed*] [*options*]

DESCRIPTION

The Point-to-Point Protocol (PPP) provides a method for transmitting datagrams over serial point-to-point links. PPP is composed of three parts: a method for encapsulating datagrams over serial links, an extensible Link Control Protocol (LCP), and a family of Network Control Protocols (NCP) for establishing and configuring different network-layer protocols.

The encapsulation scheme is provided by driver code in the kernel. PPPD provides the basic LCP, authentication support, and an NCP for establishing and configuring the Internet Protocol (IP) (called the IP Control Protocol, IPCP).

This version includes support for multilink operations, Microsoft specific CHAP authentication algorithms (known as CHAP-MS-V1 and CHAP-MS-V2) and transparent encryption (Microsoft Point-to-Point Encryption, MPPE) also.

FREQUENTLY USED OPTIONS

<tty_name>

Communicate over the named device. The string "/dev/" is prepended if necessary. If no device name is given, or if the name of the terminal connected to the standard input is given, pppd will use that terminal, and will not fork to put itself in the background. A value for this option from a privileged source cannot be overridden by a non-privileged user.

<speed>

Set the baud rate to <speed> (a decimal number). On systems such as 4.4BSD and NetBSD, any speed can be specified. Other systems (e.g. SunOS) allow only a limited set of speeds.

asyncmap *<map>*

Set the async character map to <map>. This map describes which control characters cannot be successfully received over the serial line. Pppd will ask the peer to send these characters as a 2-byte escape sequence. The argument is a 32 bit hex number with each bit representing a character to escape. Bit 0 (00000001) represents the character 0x00; bit 31 (80000000) represents the character 0x1f or ^_. If multiple *asyncmap* options are given, the values are ORed together. If no *asyncmap* option is given, no async character map will be negotiated for the receive direction; the peer should then escape *all* control characters. To escape transmitted characters, use the *escape* option.

auth

Require the peer to authenticate itself before allowing network packets to be sent or received. This option is the default if the system has a default route. If neither this option nor the *noauth* option is specified, pppd will only allow the peer to use IP addresses to which the system does not already have a route.

call *name*

Read options from the file /etc/ppp/peers/*name*. This file may contain privileged options, such as *noauth*, even if pppd is not being run by root. The *name* string may not begin with / or include .. as a pathname component. The format of the options file is described below.

callback *phone*

Request client to call back to specified *phone* number.

nocallback

Don’t argee to use callback negotiation.

+callback

Request server to negotiate callback and log negotiated results.

connect *script*
Use the executable or shell command specified by *script* to set up the serial line. This script would typically use the chat(8) program to dial the modem and start the remote ppp session. A value for this option from a privileged source cannot be overridden by a non-privileged user.

crtscts Use hardware flow control (i.e. RTS/CTS) to control the flow of data on the serial port. If neither the *crtscts*, the *nocrtscts*, the *cdtrcts* nor the *nocdtrcts* option is given, the hardware flow control setting for the serial port is left unchanged. Some serial ports (such as Macintosh serial ports) lack a true RTS output. Such serial ports use this mode to implement unidirectional flow control. The serial port will suspend transmission when requested by the modem (via CTS) but will be unable to request the modem stop sending to the computer. This mode retains the ability to use DTR as a modem control line.

defaultroute
Add a default route to the system routing tables, using the peer as the gateway, when IPCP negotiation is successfully completed. This entry is removed when the PPP connection is broken. This option is privileged if the *nodefaultroute* option has been specified.

disconnect *script*
Run the executable or shell command specified by *script* after pppd has terminated the link. This script could, for example, issue commands to the modem to cause it to hang up if hardware modem control signals were not available. The disconnect script is not run if the modem has already hung up. A value for this option from a privileged source cannot be overridden by a non-privileged user.

escape *xx,yy,...*
Specifies that certain characters should be escaped on transmission (regardless of whether the peer requests them to be escaped with its async control character map). The characters to be escaped are specified as a list of hex numbers separated by commas. Note that almost any character can be specified for the *escape* option, unlike the *asyncmap* option which only allows control characters to be specified. The characters which may not be escaped are those with hex values 0x20 - 0x3f or 0x5e.

file *name*
Read options from file *name* (the format is described below). The file must be readable by the user who has invoked pppd.

init *script*
Run the executable or shell command specified by *script* to initialize the serial line. This script would typically use the chat(8) program to configure the modem to enable auto answer. A value for this option from a privileged source cannot be overridden by a non-privileged user.

lock Specifies that pppd should create a UUCP-style lock file for the serial device to ensure exclusive access to the device.

mru *n* Set the MRU [Maximum Receive Unit] value to *n*. Pppd will ask the peer to send packets of no more than *n* bytes. The minimum MRU value is 128. The default MRU value is 1500. A value of 296 is recommended for slow links (40 bytes for TCP/IP header + 256 bytes of data). (Note that for IPv6 MRU must be at least 1280)

mtu *n* Set the MTU [Maximum Transmit Unit] value to *n*. Unless the peer requests a smaller value via MRU negotiation, pppd will request that the kernel networking code send data packets of no more than *n* bytes through the PPP network interface. (Note that for IPv6 MTU must be at least 1280)

passive Enables the "passive" option in the LCP. With this option, pppd will attempt to initiate a connection; if no reply is received from the peer, pppd will then just wait passively for a valid LCP packet from the peer, instead of exiting, as it would without this option.

OPTIONS

<local_IP_address>:*<remote_IP_address>*

Set the local and/or remote interface IP addresses. Either one may be omitted. The IP addresses can be specified with a host name or in decimal dot notation (e.g. 150.234.56.78). The default local address is the (first) IP address of the system (unless the *noipdefault* option is given). The remote address will be obtained from the peer if not specified in any option. Thus, in simple cases, this option is not required. If a local and/or remote IP address is specified with this option, pppd will not accept a different value from the peer in the IPCP negotiation, unless the *ipcp-accept-local* and/or *ipcp-accept-remote* options are given, respectively.

ipv6 *<local_interface_identifier>*,*<remote_interface_identifier>*

Set the local and/or remote 64-bit interface identifier. Either one may be omitted. The identifier must be specified in standard ascii notation of IPv6 addresses (e.g. ::dead:beef). If the *ipv6cp-use-ipaddr* option is given, the local identifier is the local IPv4 address (see above). On systems which supports a unique persistent id, such as EUI-48 derived from the Ethernet MAC address, *ipv6cp-use-persistent* option can be used to replace the *ipv6 <local>*,*<remote>* option. Otherwise the identifier is randomized.

active-filter *filter-expression*

Specifies a packet filter to be applied to data packets to determine which packets are to be regarded as link activity, and therefore reset the idle timer, or cause the link to be brought up in demand-dialling mode. This option is useful in conjunction with the **idle** option if there are packets being sent or received regularly over the link (for example, routing information packets) which would otherwise prevent the link from ever appearing to be idle. The *filter-expression* syntax is as described for tcpdump(1), except that qualifiers which are inappropriate for a PPP link, such as **ether** and **arp**, are not permitted. Generally the filter expression should be enclosed in single-quotes to prevent whitespace in the expression from being interpreted by the shell. This option is currently only available under NetBSD, and then only if both the kernel and pppd were compiled with PPP_FILTER defined.

allow-ip *address(es)*

Allow peers to use the given IP address or subnet without authenticating themselves. The parameter is parsed as for each element of the list of allowed IP addresses in the secrets files (see the AUTHENTICATION section below).

bsdcomp *nr,nt*

Request that the peer compress packets that it sends, using the BSD-Compress scheme, with a maximum code size of *nr* bits, and agree to compress packets sent to the peer with a maximum code size of *nt* bits. If *nt* is not specified, it defaults to the value given for *nr*. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values give better compression but consume more kernel memory for compression dictionaries. Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use *nobsdcomp* or *bsdcomp 0* to disable BSD-Compress compression entirely.

cdtrcts Use a non-standard hardware flow control (i.e. DTR/CTS) to control the flow of data on the serial port. If neither the *crtscts*, the *nocrtscts*, the *cdtrcts* nor the *nocdtrcts* option is given, the hardware flow control setting for the serial port is left unchanged. Some serial ports (such as Macintosh serial ports) lack a true RTS output. Such serial ports use this mode to implement true bi-directional flow control. The sacrifice is that this flow control mode does not permit using DTR as a modem control line.

chap-interval *n*

If this option is given, pppd will rechallenge the peer every *n* seconds.

chap-max-challenge *n*

Set the maximum number of CHAP challenge transmissions to *n* (default 10).

chap-restart *n*

Set the CHAP restart interval (retransmission timeout for challenges) to *n* seconds (default 3).

connect-delay *n*
Wait for up *n* milliseconds after the connect script finishes for a valid PPP packet from the peer. At the end of this time, or when a valid PPP packet is received from the peer, pppd will commence negotiation by sending its first LCP packet. The default value is 1000 (1 second). This wait period only applies if the **connect** or **pty** option is used.

debug Enables connection debugging facilities. If this option is given, pppd will log the contents of all control packets sent or received in a readable form. The packets are logged through syslog with facility *daemon* and level *debug*. This information can be directed to a file by setting up /etc/syslog.conf appropriately (see syslog.conf(5)).

default-asyncmap
Disable asyncmap negotiation, forcing all control characters to be escaped for both the transmit and the receive direction.

default-mru
Disable MRU [Maximum Receive Unit] negotiation. With this option, pppd will use the default MRU value of 1500 bytes for both the transmit and receive direction.

deflate *nr,nt*
Request that the peer compress packets that it sends, using the Deflate scheme, with a maximum window size of *2**nr* bytes, and agree to compress packets sent to the peer with a maximum window size of *2**nt* bytes. If *nt* is not specified, it defaults to the value given for *nr*. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values give better compression but consume more kernel memory for compression dictionaries. Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use *nodeflate* or *deflate 0* to disable Deflate compression entirely. (Note: pppd requests Deflate compression in preference to BSD-Compress if the peer can do either.)

demand
Initiate the link only on demand, i.e. when data traffic is present. With this option, the remote IP address must be specified by the user on the command line or in an options file. Pppd will initially configure the interface and enable it for IP traffic without connecting to the peer. When traffic is available, pppd will connect to the peer and perform negotiation, authentication, etc. When this is completed, pppd will commence passing data packets (i.e., IP packets) across the link.

The *demand* option implies the *persist* option. If this behaviour is not desired, use the *nopersist* option after the *demand* option. The *idle* and *holdoff* options are also useful in conjuction with the *demand* option.

domain *d*
Append the domain name *d* to the local host name for authentication purposes. For example, if gethostname() returns the name porsche, but the fully qualified domain name is porsche.Quotron.COM, you could specify *domain Quotron.COM*. Pppd would then use the name *porsche.Quotron.COM* for looking up secrets in the secrets file, and as the default name to send to the peer when authenticating itself to the peer. This option is privileged.

dryrun With the **dryrun** option, pppd will print out all the option values which have been set and then exit, after parsing the command line and options files and checking the option values, but before initiating the link. The option values are logged at level info, and also printed to standard output unless the device on standard output is the device that pppd would be using to communicate with the peer.

dump With the **dump** option, pppd will print out all the option values which have been set. This option is like the **dryrun** option except that pppd proceeds as normal rather than exiting.

endpoint <*epdisc*>
Sets the endpoint discriminator sent by the local machine to the peer during multilink negotiation to <*epdisc*>. The default is to use the MAC address of the first ethernet interface on the system, if any, otherwise the IPv4 address corresponding to the hostname, if any, provided it is not in the

multicast or locally-assigned IP address ranges, or the localhost address. The endpoint discriminator can be the string **null** or of the form *type*:*value*, where type is a decimal number or one of the strings **local**, **IP**, **MAC**, **magic**, or **phone**. The value is an IP address in dotted-decimal notation for the **IP** type, or a string of bytes in hexadecimal, separated by periods or colons for the other types. For the MAC type, the value may also be the name of an ethernet or similar network interface. This option is currently only available under Linux.

hide-password
When logging the contents of PAP packets, this option causes pppd to exclude the password string from the log. This is the default.

holdoff *n*
Specifies how many seconds to wait before re-initiating the link after it terminates. This option only has any effect if the *persist* or *demand* option is used. The holdoff period is not applied if the link was terminated because it was idle.

idle *n* Specifies that pppd should disconnect if the link is idle for *n* seconds. The link is idle when no data packets (i.e. IP packets) are being sent or received. Note: it is not advisable to use this option with the *persist* option without the *demand* option. If the **active-filter** option is given, data packets which are rejected by the specified activity filter also count as the link being idle.

ip Enable IPCP and IP protocols. You can disable IP negotiation if you want to use other protocols only.

ipcp-accept-local
With this option, pppd will accept the peer's idea of our local IP address, even if the local IP address was specified in an option.

ipcp-accept-remote
With this option, pppd will accept the peer's idea of its (remote) IP address, even if the remote IP address was specified in an option.

ipcp-max-configure *n*
Set the maximum number of IPCP configure-request transmissions to *n* (default 10).

ipcp-max-failure *n*
Set the maximum number of IPCP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

ipcp-max-terminate *n*
Set the maximum number of IPCP terminate-request transmissions to *n* (default 3).

ipcp-restart *n*
Set the IPCP restart interval (retransmission timeout) to *n* seconds (default 3).

ipparam *string*
Provides an extra parameter to the ip-up and ip-down scripts. If this option is given, the *string* supplied is given as the 6th parameter to those scripts.

ipv6cp-max-configure *n*
Set the maximum number of IPv6CP configure-request transmissions to *n* (default 10).

ipv6cp-max-failure *n*
Set the maximum number of IPv6CP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

ipv6cp-max-terminate *n*
Set the maximum number of IPv6CP terminate-request transmissions to *n* (default 3).

ipv6cp-restart *n*
Set the IPv6CP restart interval (retransmission timeout) to *n* seconds (default 3).

ipx Enable the IPXCP and IPX protocols. This option is presently only supported under Linux, and only if your kernel has been configured to include IPX support.

ipx-network *n*

Set the IPX network number in the IPXCP configure request frame to *n*, a hexadecimal number (without a leading 0x). There is no valid default. If this option is not specified, the network number is obtained from the peer. If the peer does not have the network number, the IPX protocol will not be started.

ipx-node *n*:*m*

Set the IPX node numbers. The two node numbers are separated from each other with a colon character. The first number *n* is the local node number. The second number *m* is the peer's node number. Each node number is a hexadecimal number, at most 10 digits long. The node numbers on the ipx-network must be unique. There is no valid default. If this option is not specified then the node numbers are obtained from the peer.

ipx-router-name <*string*>

Set the name of the router. This is a string and is sent to the peer as information data.

ipx-routing *n*

Set the routing protocol to be received by this option. More than one instance of *ipx-routing* may be specified. The '*none*' option (0) may be specified as the only instance of ipx-routing. The values may be *0* for *NONE*, *2* for *RIP/SAP*, and *4* for *NLSP*.

ipxcp-accept-local

Accept the peer's NAK for the node number specified in the ipx-node option. If a node number was specified, and non-zero, the default is to insist that the value be used. If you include this option then you will permit the peer to override the entry of the node number.

ipxcp-accept-network

Accept the peer's NAK for the network number specified in the ipx-network option. If a network number was specified, and non-zero, the default is to insist that the value be used. If you include this option then you will permit the peer to override the entry of the node number.

ipxcp-accept-remote

Use the peer's network number specified in the configure request frame. If a node number was specified for the peer and this option was not specified, the peer will be forced to use the value which you have specified.

ipxcp-max-configure *n*

Set the maximum number of IPXCP configure request frames which the system will send to *n*. The default is 10.

ipxcp-max-failure *n*

Set the maximum number of IPXCP NAK frames which the local system will send before it rejects the options. The default value is 3.

ipxcp-max-terminate *n*

Set the maximum nuber of IPXCP terminate request frames before the local system considers that the peer is not listening to them. The default value is 3.

kdebug *n*

Enable debugging code in the kernel-level PPP driver. The argument values depend on the specific kernel driver, but in general a value of 1 will enable general kernel debug messages. (Note that these messages are usually only useful for debugging the kernel driver itself.) For the Linux 2.2.x kernel driver, the value is a sum of bits: 1 to enable general debug messages, 2 to request that the contents of received packets be printed, and 4 to request that the contents of transmitted packets be printed. On most systems, messages printed by the kernel are logged by syslog(1) to a file as directed in the /etc/syslog.conf configuration file.

ktune Enables pppd to alter kernel settings as appropriate. Under Linux, pppd will enable IP forwarding (i.e. set /proc/sys/net/ipv4/ip_forward to 1) if the *proxyarp* option is used, and will enable the dynamic IP address option (i.e. set /proc/sys/net/ipv4/ip_dynaddr to 1) in demand mode if the local address changes.

lcp-echo-failure *n*
If this option is given, pppd will presume the peer to be dead if *n* LCP echo-requests are sent without receiving a valid LCP echo-reply. If this happens, pppd will terminate the connection. Use of this option requires a non-zero value for the *lcp-echo-interval* parameter. This option can be used to enable pppd to terminate after the physical connection has been broken (e.g., the modem has hung up) in situations where no hardware modem control lines are available.

lcp-echo-interval *n*
If this option is given, pppd will send an LCP echo-request frame to the peer every *n* seconds. Normally the peer should respond to the echo-request by sending an echo-reply. This option can be used with the *lcp-echo-failure* option to detect that the peer is no longer connected.

lcp-max-configure *n*
Set the maximum number of LCP configure-request transmissions to *n* (default 10).

lcp-max-failure *n*
Set the maximum number of LCP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

lcp-max-terminate *n*
Set the maximum number of LCP terminate-request transmissions to *n* (default 3).

lcp-restart *n*
Set the LCP restart interval (retransmission timeout) to *n* seconds (default 3).

linkname *name*
Sets the logical name of the link to *name*. Pppd will create a file named **ppp-***name***.pid** in /var/run (or /etc/ppp on some systems) containing its process ID. This can be useful in determining which instance of pppd is responsible for the link to a given peer system. This is a privileged option.

local Don't use the modem control lines. With this option, pppd will ignore the state of the CD (Carrier Detect) signal from the modem and will not change the state of the DTR (Data Terminal Ready) signal.

logfd *n* Send log messages to file descriptor *n*. Pppd will send log messages to at most one file or file descriptor (as well as sending the log messages to syslog), so this option and the **logfile** option are mutually exclusive. The default is for pppd to send log messages to stdout (file descriptor 1), unless the serial port is already open on stdout.

logfile *filename*
Append log messages to the file *filename* (as well as sending the log messages to syslog). The file is opened with the privileges of the user who invoked pppd, in append mode.

login Use the system password database for authenticating the peer using PAP, and record the user in the system wtmp file. Note that the peer must have an entry in the /etc/ppp/pap-secrets file as well as the system password database to be allowed access.

maxconnect *n*
Terminate the connection when it has been available for network traffic for *n* seconds (i.e. *n* seconds after the first network control protocol comes up).

maxfail *n*
Terminate after *n* consecutive failed connection attempts. A value of 0 means no limit. The default value is 10.

modem
Use the modem control lines. This option is the default. With this option, pppd will wait for the CD (Carrier Detect) signal from the modem to be asserted when opening the serial device (unless a connect script is specified), and it will drop the DTR (Data Terminal Ready) signal briefly when the connection is terminated and before executing the connect script. On Ultrix, this option implies hardware flow control, as for the *crtscts* option.

mp Enables the use of PPP multilink; this is an alias for the ‘multilink’ option. This option is currently only available under Linux.

mpshortseq
Enables the use of short (12-bit) sequence numbers in multilink headers, as opposed to 24-bit sequence numbers. This option is only available under Linux, and only has any effect if multilink is enabled (see the multilink option).

mrru *n* Sets the Maximum Reconstructed Receive Unit to *n*. The MRRU is the maximum size for a received packet on a multilink bundle, and is analogous to the MRU for the individual links. This option is currently only available under Linux, and only has any effect if multilink is enabled (see the multilink option).

ms-dns <*addr*>
If pppd is acting as a server for Microsoft Windows clients, this option allows pppd to supply one or two DNS (Domain Name Server) addresses to the clients. The first instance of this option specifies the primary DNS address; the second instance (if given) specifies the secondary DNS address. (This option was present in some older versions of pppd under the name **dns-addr**.)

ms-wins <*addr*>
If pppd is acting as a server for Microsoft Windows or "Samba" clients, this option allows pppd to supply one or two WINS (Windows Internet Name Services) server addresses to the clients. The first instance of this option specifies the primary WINS address; the second instance (if given) specifies the secondary WINS address.

multilink
Enables the use of the PPP multilink protocol. If the peer also supports multilink, then this link can become part of a bundle between the local system and the peer. If there is an existing bundle to the peer, pppd will join this link to that bundle, otherwise pppd will create a new bundle. See the MULTILINK section below. This option is currently only available under Linux.

name *name*
Set the name of the local system for authentication purposes to *name*. This is a privileged option. With this option, pppd will use lines in the secrets files which have *name* as the second field when looking for a secret to use in authenticating the peer. In addition, unless overridden with the *user* option, *name* will be used as the name to send to the peer when authenticating the local system to the peer. (Note that pppd does not append the domain name to *name*.)

netmask *n*
Set the interface netmask to *n*, a 32 bit netmask in "decimal dot" notation (e.g. 255.255.255.0). If this option is given, the value specified is ORed with the default netmask. The default netmask is chosen based on the negotiated remote IP address; it is the appropriate network mask for the class of the remote IP address, ORed with the netmasks for any non point-to-point network interfaces in the system which are on the same network. (Note: on some platforms, pppd will always use 255.255.255.255 for the netmask, if that is the only appropriate value for a point-to-point interface.)

noaccomp
Disable Address/Control compression in both directions (send and receive).

noauth Do not require the peer to authenticate itself. This option is privileged.

nobsdcomp
Disables BSD-Compress compression; **pppd** will not request or agree to compress packets using the BSD-Compress scheme.

noccp Disable CCP (Compression Control Protocol) negotiation. This option should only be required if the peer is buggy and gets confused by requests from pppd for CCP negotiation.

nocrtscts
Disable hardware flow control (i.e. RTS/CTS) on the serial port. If neither the *crtscts* nor the *nocrtscts* nor the *cdtrcts* nor the *nocdtrcts* option is given, the hardware flow control setting for the

serial port is left unchanged.

nocdtrcts
This option is a synonym for *nocrtscts*. Either of these options will disable both forms of hardware flow control.

nodefaultroute
Disable the *defaultroute* option. The system administrator who wishes to prevent users from creating default routes with pppd can do so by placing this option in the /etc/ppp/options file.

nodeflate
Disables Deflate compression; pppd will not request or agree to compress packets using the Deflate scheme.

nodetach
Don't detach from the controlling terminal. Without this option, if a serial device other than the terminal on the standard input is specified, pppd will fork to become a background process.

noendpoint
Disables pppd from sending an endpoint discriminator to the peer or accepting one from the peer (see the MULTILINK section below). This option should only be required if the peer is buggy.

noip Disable IPCP negotiation and IP communication. This option should only be required if the peer is buggy and gets confused by requests from pppd for IPCP negotiation.

noipv6 Disable IPv6CP negotiation and IPv6 communication. This option should only be required if the peer is buggy and gets confused by requests from pppd for IPv6CP negotiation.

noipdefault
Disables the default behaviour when no local IP address is specified, which is to determine (if possible) the local IP address from the hostname. With this option, the peer will have to supply the local IP address during IPCP negotiation (unless it specified explicitly on the command line or in an options file).

noipx Disable the IPXCP and IPX protocols. This option should only be required if the peer is buggy and gets confused by requests from pppd for IPXCP negotiation.

noktune
Opposite of the *ktune* option; disables pppd from changing system settings.

nolog Do not send log messages to a file or file descriptor. This option cancels the **logfd** and **logfile** options.

nomagic
Disable magic number negotiation. With this option, pppd cannot detect a looped-back line. This option should only be needed if the peer is buggy.

nomp Disables the use of PPP multilink. This option is currently only available under Linux.

nompshortseq
Disables the use of short (12-bit) sequence numbers in the PPP multilink protocol, forcing the use of 24-bit sequence numbers. This option is currently only available under Linux, and only has any effect if multilink is enabled.

nomultilink
Disables the use of PPP multilink. This option is currently only available under Linux.

nopcomp
Disable protocol field compression negotiation in both the receive and the transmit direction.

nopersist
Exit once a connection has been made and terminated. This is the default unless the *persist* or *demand* option has been specified.

nopredictor1
Do not accept or agree to Predictor-1 compression.

noproxyarp
Disable the *proxyarp* option. The system administrator who wishes to prevent users from creating proxy ARP entries with pppd can do so by placing this option in the /etc/ppp/options file.

notty Normally, pppd requires a terminal device. With this option, pppd will allocate itself a pseudo-tty master/slave pair and use the slave as its terminal device. Pppd will create a child process to act as a 'character shunt' to transfer characters between the pseudo-tty master and its standard input and output. Thus pppd will transmit characters on its standard output and receive characters on its standard input even if they are not terminal devices. This option increases the latency and CPU overhead of transferring data over the ppp interface as all of the characters sent and received must flow through the character shunt process. An explicit device name may not be given if this option is used.

novj Disable Van Jacobson style TCP/IP header compression in both the transmit and the receive direction.

novjccomp
Disable the connection-ID compression option in Van Jacobson style TCP/IP header compression. With this option, pppd will not omit the connection-ID byte from Van Jacobson compressed TCP/IP headers, nor ask the peer to do so.

papcrypt
Indicates that all secrets in the /etc/ppp/pap-secrets file which are used for checking the identity of the peer are encrypted, and thus pppd should not accept a password which, before encryption, is identical to the secret from the /etc/ppp/pap-secrets file.

pap-max-authreq *n*
Set the maximum number of PAP authenticate-request transmissions to *n* (default 10).

pap-restart *n*
Set the PAP restart interval (retransmission timeout) to *n* seconds (default 3).

pap-timeout *n*
Set the maximum time that pppd will wait for the peer to authenticate itself with PAP to *n* seconds (0 means no limit).

pass-filter *filter-expression*
Specifies a packet filter to applied to data packets being sent or received to determine which packets should be allowed to pass. Packets which are rejected by the filter are silently discarded. This option can be used to prevent specific network daemons (such as routed) using up link bandwidth, or to provide a basic firewall capability. The *filter-expression* syntax is as described for tcpdump(1), except that qualifiers which are inappropriate for a PPP link, such as **ether** and **arp**, are not permitted. Generally the filter expression should be enclosed in single-quotes to prevent whitespace in the expression from being interpreted by the shell. Note that it is possible to apply different constraints to incoming and outgoing packets using the **inbound** and **outbound** qualifiers. This option is currently only available under NetBSD, and then only if both the kernel and pppd were compiled with PPP_FILTER defined.

persist Do not exit after a connection is terminated; instead try to reopen the connection.

plugin *filename*
Load the shared library object file *filename* as a plugin. This is a privileged option.

predictor1
Request that the peer compress frames that it sends using Predictor-1 compression, and agree to compress transmitted frames with Predictor-1 if requested. This option has no effect unless the kernel driver supports Predictor-1 compression.

privgroup *group-name*
Allows members of group *group-name* to use privileged options. This is a privileged option. Use of this option requires care as there is no guarantee that members of *group-name* cannot use pppd to become root themselves. Consider it equivalent to putting the members of *group-name* in the kmem or disk group.

proxyarp
Add an entry to this system's ARP [Address Resolution Protocol] table with the IP address of the peer and the Ethernet address of this system. This will have the effect of making the peer appear to other systems to be on the local ethernet.

pty *script*
Specifies that the command *script* is to be used to communicate rather than a specific terminal device. Pppd will allocate itself a pseudo-tty master/slave pair and use the slave as its terminal device. The *script* will be run in a child process with the pseudo-tty master as its standard input and output. An explicit device name may not be given if this option is used. (Note: if the *record* option is used in conjuction with the *pty* option, the child process will have pipes on its standard input and output.)

receive-all
With this option, pppd will accept all control characters from the peer, including those marked in the receive asyncmap. Without this option, pppd will discard those characters as specified in RFC1662. This option should only be needed if the peer is buggy.

record *filename*
Specifies that pppd should record all characters sent and received to a file named *filename*. This file is opened in append mode, using the user's user-ID and permissions. This option is implemented using a pseudo-tty and a process to transfer characters between the pseudo-tty and the real serial device, so it will increase the latency and CPU overhead of transferring data over the ppp interface. The characters are stored in a tagged format with timestamps, which can be displayed in readable form using the pppdump(8) program.

remotename *name*
Set the assumed name of the remote system for authentication purposes to *name*.

refuse-chap
With this option, pppd will not agree to authenticate itself to the peer using any CHAP protocol.

refuse-chap-md5
With this option, pppd will not agree to authenticate itself to the peer using standard MD5 CHAP.

refuse-pap
With this option, pppd will not agree to authenticate itself to the peer using PAP.

require-chap
Require the peer to authenticate itself using CHAP [Challenge Handshake Authentication Protocol] authentication.

require-pap
Require the peer to authenticate itself using PAP [Password Authentication Protocol] authentication.

show-password
When logging the contents of PAP packets, this option causes pppd to show the password string in the log message.

silent With this option, pppd will not transmit LCP packets to initiate a connection until a valid LCP packet is received from the peer (as for the 'passive' option with ancient versions of pppd).

sync Use synchronous HDLC serial encoding instead of asynchronous. The device used by pppd with this option must have sync support. Currently supports Microgate SyncLink adapters under Linux and FreeBSD 2.2.8 and later.

updetach
With this option, pppd will detach from its controlling terminal once it has successfully established the ppp connection (to the point where the first network control protocol, usually the IP control protocol, has come up).

usehostname
Enforce the use of the hostname (with domain name appended, if given) as the name of the local system for authentication purposes (overrides the *name* option). This option is not normally needed since the *name* option is privileged.

usepeerdns
Ask the peer for up to 2 DNS server addresses. The addresses supplied by the peer (if any) are passed to the /etc/ppp/ip-up script in the environment variables DNS1 and DNS2. In addition, pppd will create an /etc/ppp/resolv.conf file containing one or two nameserver lines with the address(es) supplied by the peer.

user *name*
Sets the name used for authenticating the local system to the peer to *name*.

vj-max-slots *n*
Sets the number of connection slots to be used by the Van Jacobson TCP/IP header compression and decompression code to *n*, which must be between 2 and 16 (inclusive).

welcome *script*
Run the executable or shell command specified by *script* before initiating PPP negotiation, after the connect script (if any) has completed. A value for this option from a privileged source cannot be overridden by a non-privileged user.

xonxoff
Use software flow control (i.e. XON/XOFF) to control the flow of data on the serial port.

MICROSOFT SPECIFIC EXTENSIONS

This version of PPPD supports some Microsoft-specific extensions such as non-standard CHAP algorithms (known as CHAP-MS-V1 and CHAP-MS-V2) and transparent encryption of all sending and receiving traffic.

Those options allows to configure PPPD to work with Microsoft extensions of standard PPP protocol.

require-chapms
Require the peer to authenticate itself using CHAP [Challenge Handshake Authentication Protocol] authentication with Microsoft extensions [known as CHAP-MS-V1].

require-chapms-v2
Require the peer to authenticate itself using CHAP authentication with Microsoft extensions [known as CHAP-MS-V2].

refuse-chapms
With this option, pppd will not agree to authenticate itself to the peer using CHAP-MS-V1.

refuse-chapms-v2
With this option, pppd will not agree to authenticate itself to the peer using CHAP-MS-V2.

mppe-40
This option enables use of Microsoft Point-to-point Encryption (MPPE) using 40-bit encryption keys. These keys can be used with any Microsoft software (Windows 95, Windows 98, Windows NT) because their cryptographic strength is relatively low. This option requires that chapms or chapms-v2 be enabled.

mppe-128
This option enables use of Microsoft Point-to-Point Encryption (MPPE) using 128-bit encryption keys. These keys can be used with software designed for domestic usage (within US and Canada). This option requires that chapms or chapms-v2 be enabled.

mppe-stateless

This option negotiates stateless mode for Microsoft Point-to-Point Encryption (MPPE), which changes the encryption keys on every packet. The default mode is stateful (non-stateless, or single key).

OPTIONS FILES

Options can be taken from files as well as the command line. Pppd reads options from the files /etc/ppp/options, ~/.ppprc and /etc/ppp/options.*ttyname* (in that order) before processing the options on the command line. (In fact, the command-line options are scanned to find the terminal name before the options.*ttyname* file is read.) In forming the name of the options.*ttyname* file, the initial /dev/ is removed from the terminal name, and any remaining / characters are replaced with dots.

An options file is parsed into a series of words, delimited by whitespace. Whitespace can be included in a word by enclosing the word in double-quotes ("). A backslash (\) quotes the following character. A hash (#) starts a comment, which continues until the end of the line. There is no restriction on using the *file* or *call* options within an options file.

SECURITY

pppd provides system administrators with sufficient access control that PPP access to a server machine can be provided to legitimate users without fear of compromising the security of the server or the network it's on. This control is provided through restrictions on which IP addresses the peer may use, based on its authenticated identity (if any), and through restrictions on which options a non-privileged user may use. Several of pppd's options are privileged, in particular those which permit potentially insecure configurations; these options are only accepted in files which are under the control of the system administrator, or if pppd is being run by root.

The default behaviour of pppd is to allow an unauthenticated peer to use a given IP address only if the system does not already have a route to that IP address. For example, a system with a permanent connection to the wider internet will normally have a default route, and thus all peers will have to authenticate themselves in order to set up a connection. On such a system, the *auth* option is the default. On the other hand, a system where the PPP link is the only connection to the internet will not normally have a default route, so the peer will be able to use almost any IP address without authenticating itself.

As indicated above, some security-sensitive options are privileged, which means that they may not be used by an ordinary non-privileged user running a setuid-root pppd, either on the command line, in the user's ~/.ppprc file, or in an options file read using the *file* option. Privileged options may be used in /etc/ppp/options file or in an options file read using the *call* option. If pppd is being run by the root user, privileged options can be used without restriction.

When opening the device, pppd uses either the invoking user's user ID or the root UID (that is, 0), depending on whether the device name was specified by the user or the system administrator. If the device name comes from a privileged source, that is, /etc/ppp/options or an options file read using the *call* option, pppd uses full root privileges when opening the device. Thus, by creating an appropriate file under /etc/ppp/peers, the system administrator can allow users to establish a ppp connection via a device which they would not normally have permission to access. Otherwise pppd uses the invoking user's real UID when opening the device.

AUTHENTICATION

Authentication is the process whereby one peer convinces the other of its identity. This involves the first peer sending its name to the other, together with some kind of secret information which could only come from the genuine authorized user of that name. In such an exchange, we will call the first peer the "client" and the other the "server". The client has a name by which it identifies itself to the server, and the server also has a name by which it identifies itself to the client. Generally the genuine client shares some secret (or password) with the server, and authenticates itself by proving that it knows that secret. Very often, the names used for authentication correspond to the internet hostnames of the peers, but this is not essential.

At present, pppd supports two authentication protocols: the Password Authentication Protocol (PAP) and the Challenge Handshake Authentication Protocol (CHAP). PAP involves the client sending its name and a cleartext password to the server to authenticate itself. In contrast, the server initiates the CHAP

authentication exchange by sending a challenge to the client (the challenge packet includes the server's name). The client must respond with a response which includes its name plus a hash value derived from the shared secret and the challenge, in order to prove that it knows the secret.

The PPP protocol, being symmetrical, allows both peers to require the other to authenticate itself. In that case, two separate and independent authentication exchanges will occur. The two exchanges could use different authentication protocols, and in principle, different names could be used in the two exchanges.

The default behaviour of pppd is to agree to authenticate if requested, and to not require authentication from the peer. However, pppd will not agree to authenticate itself with a particular protocol if it has no secrets which could be used to do so.

Pppd stores secrets for use in authentication in secrets files (/etc/ppp/pap-secrets for PAP, /etc/ppp/chap-secrets for CHAP). Both secrets files have the same format. The secrets files can contain secrets for pppd to use in authenticating itself to other systems, as well as secrets for pppd to use when authenticating other systems to itself.

Each line in a secrets file contains one secret. A given secret is specific to a particular combination of client and server - it can only be used by that client to authenticate itself to that server. Thus each line in a secrets file has at least 3 fields: the name of the client, the name of the server, and the secret. These fields may be followed by a list of the IP addresses that the specified client may use when connecting to the specified server.

A secrets file is parsed into words as for a options file, so the client name, server name and secrets fields must each be one word, with any embedded spaces or other special characters quoted or escaped. Note that case is significant in the client and server names and in the secret.

If the secret starts with an '@', what follows is assumed to be the name of a file from which to read the secret. A "*" as the client or server name matches any name. When selecting a secret, pppd takes the best match, i.e. the match with the fewest wildcards.

Any following words on the same line are taken to be a list of acceptable IP addresses for that client. If there are only 3 words on the line, or if the first word is "-", then all IP addresses are disallowed. To allow any address, use "*". A word starting with "!" indicates that the specified address is *not* acceptable. An address may be followed by "/" and a number *n*, to indicate a whole subnet, i.e. all addresses which have the same value in the most significant *n* bits. In this form, the address may be followed by a plus sign ("+") to indicate that one address from the subnet is authorized, based on the ppp network interface unit number in use. In this case, the host part of the address will be set to the unit number plus one.

Thus a secrets file contains both secrets for use in authenticating other hosts, plus secrets which we use for authenticating ourselves to others. When pppd is authenticating the peer (checking the peer's identity), it chooses a secret with the peer's name in the first field and the name of the local system in the second field. The name of the local system defaults to the hostname, with the domain name appended if the *domain* option is used. This default can be overridden with the *name* option, except when the *usehostname* option is used.

When pppd is choosing a secret to use in authenticating itself to the peer, it first determines what name it is going to use to identify itself to the peer. This name can be specified by the user with the *user* option. If this option is not used, the name defaults to the name of the local system, determined as described in the previous paragraph. Then pppd looks for a secret with this name in the first field and the peer's name in the second field. Pppd will know the name of the peer if CHAP authentication is being used, because the peer will have sent it in the challenge packet. However, if PAP is being used, pppd will have to determine the peer's name from the options specified by the user. The user can specify the peer's name directly with the *remotename* option. Otherwise, if the remote IP address was specified by a name (rather than in numeric form), that name will be used as the peer's name. Failing that, pppd will use the null string as the peer's name.

When authenticating the peer with PAP, the supplied password is first compared with the secret from the secrets file. If the password doesn't match the secret, the password is encrypted using crypt() and checked against the secret again. Thus secrets for authenticating the peer can be stored in encrypted form if desired.

If the *papcrypt* option is given, the first (unencrypted) comparison is omitted, for better security.

Furthermore, if the *login* option was specified, the username and password are also checked against the system password database. Thus, the system administrator can set up the pap-secrets file to allow PPP access only to certain users, and to restrict the set of IP addresses that each user can use. Typically, when using the *login* option, the secret in /etc/ppp/pap-secrets would be "", which will match any password supplied by the peer. This avoids the need to have the same secret in two places.

Authentication must be satisfactorily completed before IPCP (or any other Network Control Protocol) can be started. If the peer is required to authenticate itself, and fails to do so, pppd will terminated the link (by closing LCP). If IPCP negotiates an unacceptable IP address for the remote host, IPCP will be closed. IP packets can only be sent or received when IPCP is open.

In some cases it is desirable to allow some hosts which can't authenticate themselves to connect and use one of a restricted set of IP addresses, even when the local host generally requires authentication. If the peer refuses to authenticate itself when requested, pppd takes that as equivalent to authenticating with PAP using the empty string for the username and password. Thus, by adding a line to the pap-secrets file which specifies the empty string for the client and password, it is possible to allow restricted access to hosts which refuse to authenticate themselves.

ROUTING

When IPCP negotiation is completed successfully, pppd will inform the kernel of the local and remote IP addresses for the ppp interface. This is sufficient to create a host route to the remote end of the link, which will enable the peers to exchange IP packets. Communication with other machines generally requires further modification to routing tables and/or ARP (Address Resolution Protocol) tables. In most cases the *defaultroute* and/or *proxyarp* options are sufficient for this, but in some cases further intervention is required. The /etc/ppp/ip-up script can be used for this.

Sometimes it is desirable to add a default route through the remote host, as in the case of a machine whose only connection to the Internet is through the ppp interface. The *defaultroute* option causes pppd to create such a default route when IPCP comes up, and delete it when the link is terminated.

In some cases it is desirable to use proxy ARP, for example on a server machine connected to a LAN, in order to allow other hosts to communicate with the remote host. The *proxyarp* option causes pppd to look for a network interface on the same subnet as the remote host (an interface supporting broadcast and ARP, which is up and not a point-to-point or loopback interface). If found, pppd creates a permanent, published ARP entry with the IP address of the remote host and the hardware address of the network interface found.

When the *demand* option is used, the interface IP addresses have already been set at the point when IPCP comes up. If pppd has not been able to negotiate the same addresses that it used to configure the interface (for example when the peer is an ISP that uses dynamic IP address assignment), pppd has to change the interface IP addresses to the negotiated addresses. This may disrupt existing connections, and the use of demand dialling with peers that do dynamic IP address assignment is not recommended.

MULTILINK

Multilink PPP provides the capability to combine two or more PPP links between a pair of machines into a single 'bundle', which appears as a single virtual PPP link which has the combined bandwidth of the individual links. Currently, multilink PPP is only supported under Linux.

Pppd detects that the link it is controlling is connected to the same peer as another link using the peer's endpoint discriminator and the authenticated identity of the peer (if it authenticates itself). The endpoint discriminator is a block of data which is hopefully unique for each peer. Several types of data can be used, including locally-assigned strings of bytes, IP addresses, MAC addresses, randomly strings of bytes, or E-164 phone numbers. The endpoint discriminator sent to the peer by pppd can be set using the endpoint option.

In circumstances the peer may send no endpoint discriminator or a non-unique value. The optional bundle option adds an extra string which is added to the peer's endpoint discriminator and authenticated identity when matching up links to be joined together in a bundle. The bundle option can also be used to allow the establishment of multiple bundles between the local system and the peer. Pppd uses a TDB database in

/var/run/pppd.tdb to match up links.

Assuming that multilink is enabled and the peer is willing to negotiate multilink, then when pppd is invoked to bring up the first link to the peer, it will detect that no other link is connected to the peer and create a new bundle, that is, another ppp network interface unit. When another pppd is invoked to bring up another link to the peer, it will detect the existing bundle and join its link to it. Currently, if the first pppd terminates (for example, because of a hangup or a received signal) the bundle is destroyed.

EXAMPLES

The following examples assume that the /etc/ppp/options file contains the *auth* option (as in the default /etc/ppp/options file in the ppp distribution).

Probably the most common use of pppd is to dial out to an ISP. This can be done with a command such as

```
pppd call isp
```

where the /etc/ppp/peers/isp file is set up by the system administrator to contain something like this:

```
ttyS0 19200 crtscts
connect '/usr/sbin/chat -v -f /etc/ppp/chat-isp'
noauth
```

In this example, we are using chat to dial the ISP's modem and go through any logon sequence required. The /etc/ppp/chat-isp file contains the script used by chat; it could for example contain something like this:

```
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
ABORT "BUSY"
ABORT "Username/Password Incorrect"
"" "at"
OK "at&d0&c1"
OK "atdt2468135"
"name:" "^Umyuserid"
"word:" "\qmypassword"
"ispts" "\q^Uppp"
"~ -^Uppp-~ "
```

See the chat(8) man page for details of chat scripts.

Pppd can also be used to provide a dial-in ppp service for users. If the users already have login accounts, the simplest way to set up the ppp service is to let the users log in to their accounts and run pppd (installed setuid-root) with a command such as

```
pppd proxyarp
```

To allow a user to use the PPP facilities, you need to allocate an IP address for that user's machine and create an entry in /etc/ppp/pap-secrets or /etc/ppp/chap-secrets (depending on which authentication method the PPP implementation on the user's machine supports), so that the user's machine can authenticate itself. For example, if Joe has a machine called "joespc" which is to be allowed to dial in to the machine called "server" and use the IP address joespc.my.net, you would add an entry like this to /etc/ppp/pap-secrets or /etc/ppp/chap-secrets:

```
joespc    server    "joe's secret"      joespc.my.net
```

Alternatively, you can create a username called (for example) "ppp", whose login shell is pppd and whose home directory is /etc/ppp. Options to be used when pppd is run this way can be put in /etc/ppp/.ppprc.

If your serial connection is any more complicated than a piece of wire, you may need to arrange for some control characters to be escaped. In particular, it is often useful to escape XON (^Q) and XOFF (^S), using *asyncmap a0000*. If the path includes a telnet, you probably should escape ^] as well (*asyncmap 200a0000*). If the path includes an rlogin, you will need to use the *escape ff* option on the end which is

running the rlogin client, since many rlogin implementations are not transparent; they will remove the sequence [0xff, 0xff, 0x73, 0x73, followed by any 8 bytes] from the stream.

EXAMPLES OF MICROSOFT EXTENSIONS

To enable full-featured Microsoft encryption you must add those options to your */etc/ppp/options/fR fi le:*

```
+chapms
+chapms-v2
mppe-40
mppe-128
mppe-stateless
```

To authentifi cate client with Microsoft CHAP version 1 or 2, you need to use those options:

```
auth
+chap
+chapms
+chapms-v2
```

To make possible to authentifi cate itself with Microsoft CHAP version 1 or 2, you need to add those options:

```
+chapms
+chapms-v2
```

DIAGNOSTICS

(This can be overriden by recompiling pppd with the macro LOG_PPP defi ned as the desired facility.) In order to see the error and debug messages, you will need to edit your /etc/syslog.conf fi le to direct the messages to the desired output device or fi le.

The *debug* option causes the contents of all control packets sent or received to be logged, that is, all LCP, PAP, CHAP or IPCP packets. This can be useful if the PPP negotiation does not succeed or if authentication fails. If debugging is enabled at compile time, the *debug* option also causes other debugging messages to be logged.

Debugging can also be enabled or disabled by sending a SIGUSR1 signal to the pppd process. This signal acts as a toggle.

EXIT STATUS

The exit status of pppd is set to indicate whether any error was detected, or the reason for the link being terminated. The values used are:

0 Pppd has detached, or otherwise the connection was successfully established and terminated at the peer's request.

1 An immediately fatal error of some kind occurred, such as an essential system call failing, or running out of virtual memory.

2 An error was detected in processing the options given, such as two mutually exclusive options being used.

3 Pppd is not setuid-root and the invoking user is not root.

4 The kernel does not support PPP, for example, the PPP kernel driver is not included or cannot be loaded.

5 Pppd terminated because it was sent a SIGINT, SIGTERM or SIGHUP signal.

6 The serial port could not be locked.

7 The serial port could not be opened.

8 The connect script failed (returned a non-zero exit status).

9 The command specifi ed as the argument to the *pty* option could not be run.

10 The PPP negotiation failed, that is, it didn't reach the point where at least one network protocol (e.g. IP) was running.

11 The peer system failed (or refused) to authenticate itself.

12 The link was established successfully and terminated because it was idle.

13 The link was established successfully and terminated because the connect time limit was reached.

14 Callback was negotiated and an incoming call should arrive shortly.

15 The link was terminated because the peer is not responding to echo requests.

16 The link was terminated by the modem hanging up.

17 The PPP negotiation failed because serial loopback was detected.

18 The init script failed (returned a non-zero exit status).

19 We failed to authenticate ourselves to the peer.

SCRIPTS

Pppd invokes scripts at various stages in its processing which can be used to perform site-specific ancillary processing. These scripts are usually shell scripts, but could be executable code files instead. Pppd does not wait for the scripts to finish. The scripts are executed as root (with the real and effective user-id set to 0), so that they can do things such as update routing tables or run privileged daemons. Be careful that the contents of these scripts do not compromise your system's security. Pppd runs the scripts with standard input, output and error redirected to /dev/null, and with an environment that is empty except for some environment variables that give information about the link. The environment variables that pppd sets are:

DEVICE
The name of the serial tty device being used.

IFNAME
The name of the network interface being used.

IPLOCAL
The IP address for the local end of the link. This is only set when IPCP has come up.

IPREMOTE
The IP address for the remote end of the link. This is only set when IPCP has come up.

PEERNAME
The authenticated name of the peer. This is only set if the peer authenticates itself.

SPEED
The baud rate of the tty device.

ORIG_UID
The real user-id of the user who invoked pppd.

PPPLOGNAME
The username of the real user-id that invoked pppd. This is always set.

For the ip-down and auth-down scripts, pppd also sets the following variables giving statistics for the connection:

CONNECT_TIME
The number of seconds from when the PPP negotiation started until the connection was terminated.

BYTES_SENT
The number of bytes sent (at the level of the serial port) during the connection.

BYTES_RCVD
The number of bytes received (at the level of the serial port) during the connection.

LINKNAME
The logical name of the link, set with the *linkname* option.

Pppd invokes the following scripts, if they exist. It is not an error if they don't exist.

/etc/ppp/auth-up
A program or script which is executed after the remote system successfully authenticates itself. It is executed with the parameters

interface-name peer-name user-name tty-device speed

Note that this script is not executed if the peer doesn't authenticate itself, for example when the *noauth* option is used.

/etc/ppp/auth-down
A program or script which is executed when the link goes down, if /etc/ppp/auth-up was previously executed. It is executed in the same manner with the same parameters as /etc/ppp/auth-up.

/etc/ppp/ip-up
A program or script which is executed when the link is available for sending and receiving IP packets (that is, IPCP has come up). It is executed with the parameters

interface-name tty-device speed local-IP-address remote-IP-address ipparam

/etc/ppp/ip-down
A program or script which is executed when the link is no longer available for sending and receiving IP packets. This script can be used for undoing the effects of the /etc/ppp/ip-up script. It is invoked in the same manner and with the same parameters as the ip-up script.

/etc/ppp/ipv6-up
Like /etc/ppp/ip-up, except that it is executed when the link is available for sending and receiving IPv6 packets. It is executed with the parameters

interface-name tty-device speed local-link-local-address remote-link-local-address ipparam

/etc/ppp/ipv6-down
Similar to /etc/ppp/ip-down, but it is executed when IPv6 packets can no longer be transmitted on the link. It is executed with the same parameters as the ipv6-up script.

/etc/ppp/ipx-up
A program or script which is executed when the link is available for sending and receiving IPX packets (that is, IPXCP has come up). It is executed with the parameters

interface-name tty-device speed network-number local-IPX-node-address remote-IPX-node-address local-IPX-routing-protocol remote-IPX-routing-protocol local-IPX-router-name remote-IPX-router-name ipparam pppd-pid

The local-IPX-routing-protocol and remote-IPX-routing-protocol field may be one of the following:

NONE to indicate that there is no routing protocol
RIP to indicate that RIP/SAP should be used
NLSP to indicate that Novell NLSP should be used
RIP NLSP to indicate that both RIP/SAP and NLSP should be used

/etc/ppp/ipx-down
A program or script which is executed when the link is no longer available for sending and receiving IPX packets. This script can be used for undoing the effects of the /etc/ppp/ipx-up script. It is invoked in the same manner and with the same parameters as the ipx-up script.

FILES

/var/run/ppp*n*.pid (BSD or Linux), **/etc/ppp/ppp*n*.pid** (others)
Process-ID for pppd process on ppp interface unit *n*.

/var/run/ppp-*name*.pid (BSD or Linux), **/etc/ppp/ppp-*name*.pid** (others)
Process-ID for pppd process for logical link *name* (see the *linkname* option).

/etc/ppp/pap-secrets
Usernames, passwords and IP addresses for PAP authentication. This file should be owned by root and not readable or writable by any other user. Pppd will log a warning if this is not the case.

/etc/ppp/chap-secrets
Names, secrets and IP addresses for CHAP authentication. As for /etc/ppp/pap-secrets, this file should be owned by root and not readable or writable by any other user. Pppd will log a warning if this is not the case.

/etc/ppp/options
System default options for pppd, read before user default options or command-line options.

~/.ppprc
User default options, read before /etc/ppp/options.*ttyname*.

/etc/ppp/options.*ttyname*
System default options for the serial port being used, read after ~ /.ppprc. In forming the *ttyname* part of this filename, an initial /dev/ is stripped from the port name (if present), and any slashes in the remaining part are converted to dots.

/etc/ppp/peers
A directory containing options files which may contain privileged options, even if pppd was invoked by a user other than root. The system administrator can create options files in this directory to permit non-privileged users to dial out without requiring the peer to authenticate, but only to certain trusted peers.

SEE ALSO

RFC1144
Jacobson, V. *Compressing TCP/IP headers for low-speed serial links.* February 1990.

RFC1321
Rivest, R. *The MD5 Message-Digest Algorithm.* April 1992.

RFC1332
McGregor, G. *PPP Internet Protocol Control Protocol (IPCP).* May 1992.

RFC1334
Lloyd, B.; Simpson, W.A. *PPP authentication protocols.* October 1992.

RFC1661
Simpson, W.A. *The Point−to−Point Protocol (PPP).* July 1994.

RFC1662
Simpson, W.A. *PPP in HDLC-like Framing.* July 1994.

RFC2472
Haskin, D. *IP Version 6 over PPP* December 1998.

NOTES

The following signals have the specified effect when sent to pppd.

SIGINT, SIGTERM
These signals cause pppd to terminate the link (by closing LCP), restore the serial device settings, and exit.

SIGHUP
This signal causes pppd to terminate the link, restore the serial device settings, and close the serial device. If the *persist* or *demand* option has been specified, pppd will try to reopen the serial device and start another connection (after the holdoff period). Otherwise pppd will exit. If this signal is received during the holdoff period, it causes pppd to end the holdoff period immediately.

SIGUSR1

This signal toggles the state of the *debug* option.

SIGUSR2

This signal causes pppd to renegotiate compression. This can be useful to re-enable compression after it has been disabled as a result of a fatal decompression error. (Fatal decompression errors generally indicate a bug in one or other implementation.)

AUTHORS

Paul Mackerras (Paul.Mackerras@cs.anu.edu.au), based on earlier work by Drew Perkins, Brad Clements, Karl Fox, Greg Christy, and Brad Parker.