Support Group Application Note
Number: 282
Issue: 1.0
Author: DW/DNW

Writing Command Scripts for Acorn InterTalk

This Application Note details the scripting language supported by InterTalk, which is used for purposes such as automating the connection and authentication procedures required at each login by Internet service providers. Example scripts are given, and the prerequisite services required are detailed.

NB. The details of pppconnect and PPP compatibility apply to InterTalk version 2.x and later; earlier versions support SLIP only.

Applicable Hardware :	All machines equipped with RISC OS 3.1 or later	Related Application Notes:	234: Peripheral Interfacing via the Serial Port

Support Group
Acorn Computers Limited
Acorn House
Vision Park
Histon, Cambridge
CB4 4AE

Introduction

One of the most useful facilities to bear in mind when setting up an InterTalk system is the InterTalk scripting language. This language gives you the flexibility to configure InterTalk so that it can operate with a wide range of connection and authentication interfaces used by Internet service providers, and automate the procedure for connecting your system to the Internet so that your machine can download your email and USENET News at any configured time of day.

This Application Note leads you through a typical script, and contains further details about IP allocation in InterTalk and InterTalk's requirements regarding protocols which must be supported by service providers, in addition to supplying a reference to the InterTalk script command set.

Protocols and Interfaces

Currently, InterTalk is capable of transmitting and receiving IP datagrams as SLIP packets, PPP packets or "unwrapped" IP packets.

SLIP provides a point-to-point connection between two devices for the transmission of Internet datagrams; the devices can be either two computers, or a computer and an Internet router. SLIP modifies a standard Internet datagram by appending a special SLIP END character to it, which allows datagrams to be distinguished as separate when transmitted serially.

PPP provides a point to point connection also. The advantages PPP has over SLIP is that it can compress the header information (which contains such information as the intended destination of the packet), and test the integrity of inter-system connections, rerouting packets where appropriate. PPP will also take care of much of the connection configuration, thus making life easier for Intertalk script authors.

The single "device" which is used to transmit and receive SLIP packets is the single serial port fitted as standard to your machine; when referenced by the SLIP driver, this device is referred to as "sl0" (SLIP port 0). This is currently the only device which InterTalk supports for communication with the Service Provider when using SLIP.

The InterTalk PPP driver supports the "block driver" interface, which has been available for some time as a Freeware interface specification and conformant device driver set. This now means that, in addition to the internal serial port, third party devices such as high-speed dual serial port cards may be used which enable pre Risc PC machine to exchange data with a modem at rates > 19200 baud. Whichever physical interface is used, the PPP driver referes to it as "ppp0" (PPP port 0).

If you have a direct ethernet connection to the Internet (eg via ISDN), Intertalk can make use of this as well.

The use of SLIP or PPP only affects the fundamental IP layer, and is therefore transparent to the upper-layer protocols; hence service-oriented protocols such as SMTP (email), NNTP (USENET News), FTP (File Transfer Protocol) HTTP (World Wide Web) and other equivalent-layer protocols such as telnet will all work correctly over SLIP or PPP.

Other factors to bear in mind are that the configuration of InterTalk is subtly different depending on whether your provider allocates you a fixed IP address or has dynamic IP allocation (in which your IP address is determined as part of each logon procedure), and that the Point of Presence (PoP) you plan to use supports modem speeds up to the maximum rate supported by your own modem, subject to the limitations imposed by the serial port (see Application Note 234 for further details).

Anatomy of a Logon Script

The following paragraphs provide a line-by-line dissection of a typical InterTalk logon script; lines printed in the **CompuText** font comprise the script itself. The scripts themselves are stored with the leafname "Script" in the directory relevant to the service provider in the !MailServ.Providers hierarchy; for example, the script for Demon Internet Services is stored as "!Mailserv.Providers.Demon.Script". The remaining files in the !Mailserv.<provider> directory contain details of the telephone numbers of the currently-known Points of Presence (PoPs) used by each provider.

#Demon Internet SLIP

This line is a comment, as it starts with a #. In this case, the line just signifies that this script is intended for use when connecting to Demon Internet Services, in this case using the SLIP protocols. Comments can be freely distributed throughout the script.

retry 5 5 10

Set the connection retry system so that connection is attempted five times. The delay between the first and second attempts is five seconds, after which it tries at 10 second intervals.

NewsRetry 50

Sets the connection retry system to attempt up to 50 retries when contacting the USENET News server.

***rmensure slip 2.07 rmload System:Modules.Network.SLIP**
***rmensure slip 2.07 Error Slip version 2.07 or later is required**

These lines ensure that the correct version of the SLIP driver is loaded.
If you are using the PPP driver, replace these lines with:

***rmensure PPP 1.02 rmload System:Modules.Network.PPP**
***rmensure PPP 1.02 Error PPP version 1.02 or later is required**

+ifconfig -e sl0 down

This switches the SLIP driver off; all transactions which take place between this line of the script and a line containing **ifconfig -e sl0 up** take place using "ordinary" V.*nn* modem protocols and flow control.
The PPP driver initialises in a switched-off state, so this line can be omitted in the case of PPP being used.
The use of a "+" as opposed to a "*" to prefix the command signifies that, if the command should fail to execute successfully, it should fail silently and hence allow the login script to continue executing rather than aborting and reporting an error.

SerialSetup

This initialises the serial port to the appropriate baud rate, word length, parity etc.

Timeout 10

This is used by the Wait command; a Wait is now configured to time out (reporting an error, terminating the script and closing the serial connection) 10 seconds after it is encountered.

Abort Busy
Abort Carrier
Abort incorrect
Abort tone

If any of the strings following "Abort" are sent to the computer by the modem during the next phase of establishing the connection, the script will close the connection to the modem and terminate.

`Echo`
Tells the system to start copying incoming data to its logging system.

`Send ATZ`
Sends the Hayes command string to the modem to reset it to its default configuration.

`Wait OK`
Pauses until the string "OK" is sent to the computer by the modem, or generates an error, closes the connection and terminates the script if "OK" is not received before 10 seconds have elapsed.

`Init`
Sends the configured initialisation string to the modem.

`Wait OK`
As above.

`Timeout 120`
`Dial`
Resets the **`Wait`** timeout to 120 seconds, and uses the configured dialling method to contact the Service Provider.

`Wait ogin:`
Waits until the string "ogin:" is received, or until 120 seconds have elapsed. As the "L" at the front of "Login:" varies from upper to lower case between service providers (and occasionally between PoPs run by the same service provider), it has been omitted from the test.

`Timeout 60`
`Login`
Modified the timeout period again, and sends the configured login name terminated by a carriage return.

`Wait assword:`
`Password`
Waits for the tail end of the "Password:" prompt from the Service Provider, and sends the configured password in response.

`Wait ocol:`
`Send idle=120,SLIP`
This is a part of the connection dialogue which appears to be operational for a number of Service Providers, although the inclusion of the "idle" response seems unique to Demon Internet Services; the Service Provider requests details of the protocol you will be running once your full IP connection is brought up. "ocol:" is the tail-end of Demon's "Protocol:" query; everything following the "Send" in the line below it is sent verbatim in response.
If you were using the PPP driver you would have the line:
`Send idle=120,PPP`
instead. You may also wish to "fine-tune" the timeout value.

`Wait ddress:`
`GetIP`
This operation will, if dynamic IP allocation is enabled, configure your machine to use the IP address returned for this session by the Service Provider (see below); otherwise, if you have a static IP address, it

will be ignored.

`Wait HELLO`
In the case of Demon Internet Services, "HELLO" signifies that the logon and configuration procedure is complete; from here on, data sent to you by the Service Provider will comprise SLIP packets. Your own Service Provider may use a different string; check the documentation accompanying your contract with them for details.

`Config sl0`
This sets up the broadcast address, netmask etc which will be used by the SLIP interface during the session

`+ifconfig -e sl0 up`
This line activates the SLIP driver, bringing you fully online.

`Route`
This sets up the IP routing system between you and your Service Provider, taking into account which PoP you are using and which modem you are connected to at that PoP automatically.

If you are using the PPP driver you can ignore the above three commands. Instead you need to replace then with the following two lines.

```
*pppconnect InternalPC 0 noipdefault defaultroute crtscts modem asyncmap 0
getifaddr ppp0
```

This tells the PPP module to connect to the device "InternalPC" (which is the internal serial port using a standard PC cable to connect to the modem) using the speed already defined on the serial port, setting up all that is needed automatically. If you are using an Acorn-style cable (as described in Application Note 234, and intended for use with systems fitted with 6551 serial controller ICs) "InternalPC" should be replaced with "Internal". For external and third party high speed serial adaptors, details of the appropriate interface name should be supplied with the interface.
Your modem must be configured to use RTS/CTS, to hang up when DTR is dropped and to make DCD follow the state of the line.

The remaining parameters used by *pppconnect are detailed, along with all other *pppconnect options, in the pppconnect for RISC OS man page (Appendix B).

Writing your Own Scripts

The script above is designed to work with Points of Presence (PoPs) run by Demon Internet Services; if you are using a different Service Provider, you may well find that the detail of the logon procedure differs from the one above, or you may have to re-tune your timeout thresholds. Up to the point where the Service Provider is dialled, however, it should be possible to duplicate the script "as is" depending upon the reaction of your modem to standard Hayes commands.

The details supplied by your Service Provider when you commence your contract with them should provide sufficient information to write a script which will operate with their logon procedures, however tuning the timeouts can be a fine art and is best judged by experience.

Handling Dynamic IP Allocation

InterTalk is capable of handling the dynamic IP address allocation systems used by some Service Providers; if the IP address in the configuration file is left blank or set to Auto, the insertion of **`GetIP`** at the appropriate point in the auto-logon script will enable the IP address of your machine to be configured according to the dotted decimal IP address sent by the Service Provider.

If the IP address in the configuration file was left blank, the new IP address as returned by the service provider is put in the appropriate field in the configuration file and the file is then re-saved.

If the IP address field in the configuration file contains the string "Auto", the IP address returned by the Service Provider is used for the duration of the session, but the contents of the configuration file are not altered.

If the IP address in the configuration file already has an IP address in it, the GetIP command is ignored.

Appendix A: Scripting Language Command Set

Abort
Syntax: Abort <*string*>
This adds the specified string to the list of abort strings. If any string in this list is received from the service provider, the script is aborted with the error "Script aborted (<*string*>)". InterTalk will then try to re-establish the connection to the service provider, in accordance with the parameters passed to the **Retry** command (*q.v.*)

Config
Syntax: Config <*interface*>
Configures the named interface as a machine-to-provider link connecting the configured IP address with the configured gateway.

Dial
Syntax: Dial
Send either ATDT or ATDP (Hayes modem commands for tone and pulse dialling respectively) to the modem, depending on the configured dial method, followed by the configured telephone number.

Echo
Syntax: Echo
Sets the server to echo incoming data to the server log.

GetIFAddr
Syntax: GetIFAddr <*interface*>
Attempts to determine the IP address of the interface when using dynamic IP allocation. The command waits for five seconds, during which time the remote system should reply with the address; if this does not happen, the command times out, generating an error and aborting the logon script.

GetIP
Syntax: GetIP
If the IP address in the configuration file is left blank or set to Auto, this command looks in the input from the service provider for an IP address and configures it as the IP address to use for the local machine (giving the functionality of dynamic IP allocation).

If the IP address in the configuration file was left blank, the new IP address as returned by the service provider is put in the appropriate field in the configuration file and the file is then re-saved.

If the IP address in the configuration file already has an IP address in it, the GetIP command is ignored.

Hangup
Syntax: Hangup
This command waits for one second, then sends "+++" to the modem, followed by another one second pause.
It then sends the configured hangup string to the modem.

Init
Syntax: Init
This command sends the configured init string to the modem.

Login
Syntax: Login
This command sends the configured local system name to the service provider, followed by a carriage return.

NewsRetry
Syntax: NewsRetry <*n*>
Sets the number of retries to use when attempting to connect to the USENET News server.

NoEcho
Syntax: NoEcho
Stops the server from echoing incoming data in the system log.

OS Command
*Syntax: *<command>*
Passes a command to the RISC OS Supervisor for execution.

Password
Syntax: Password
Sends the configured password followed by a carriage return.

Retry
Syntax: Retry *x y z*
Sets the number of times the system attempts to make a connection to the service provider.

x = number of times to retry before failure
y = time in seconds between first and second attempts
z = time in seconds between second and subsequent attempts

Route
Syntax: Route
Sets a default IP route through the configured gateway address.

Send
Syntax: Send <*string*>
Sends the attached *string* to the modem, terminated by a carriage return.

SerialSetup
Syntax: SerialSetup
Configures the serial port to match the configured setup parameters.

Timeout
Syntax: Timeout <*n*>
Sets the delay in seconds to *n* before a wait times out.

Wait
Syntax: Wait <*string*>
Waits for *string* to be received from the service provider. The string is case insensitive. If the string is not received within the time set by the **Timeout** command (*q.v.*) the script will terminate with the error "Script timed out."

Appendix B: pppconnect for RISC OS

SYNOPSIS
*PPPConnect [<driver>[:<port>]] [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).

FREQUENTLY USED OPTIONS
<driver>[:<port>]
Load the named blockdriver from SerialDev:Modules, and use the given port for communication. If this is not specified, the previously used driver will be used. The PPP module contains the Internal driver, which will be used if this option is never specified.

<speed>
Set the baud rate to <speed>. The available speeds depend on the device you are using. If not specified, it will use the current setting of the chosen driver.

asyncmap <map>
Set the async character map to <map>. This map describes which control characters cannot be successfully received over the serial line. PPP 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 will then escape all control characters.

auth
Require the peer to authenticate itself before allowing network packets to be sent or received.

crtscts

Use hardware flow control (i.e. RTS/CTS) to control the flow of data on the serial port.

xonxoff

Use software flow control (i.e. XON/XOFF) to control the flow of data on the serial port. If neither this option nor the previous is used, the current state of the driver will be used

-crtscts

A synonym for xonxoff.

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.

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 <f>

Read options from file <f> (the format is described below).

mru <n>

Set the MRU [Maximum Receive Unit] value to <n> for negotiation. 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).

netmask <n>

Set the interface netmask to <n>, a 32 bit netmask in "decimal dot" notation (e.g. 255.255.255.0).

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 does without this option).

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 old versions of pppd).

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.

-all
Don't request or allow negotiation of any options for LCP and IPCP (use default values).

-ac
Disable Address/Control compression negotiation (use default, i.e. address/control field disabled).

-am
Disable asyncmap negotiation (use the default asyncmap, i.e. escape all control characters)

-as <n>
Same as asyncmap <n>

-ip
Disable IP address negotiation (with this option, the remote IP address must be specified with an option on the command line or in an options file).

-mn
Disable magic number negotiation. With this option, pppd cannot detect a looped-back line.

-mru
Disable MRU [Maximum Receive Unit] negotiation (use default, i.e. 1500).

-p
Same as the passive option.

-pc
Disable protocol field compression negotiation (use default, i.e. protocol field compression disabled).

+ua <p>
Agree to authenticate using PAP [Password Authentication Protocol] if requested by the peer, and use the data in file <p> for the user and password to send to the peer. The file contains the remote user name, followed by a newline, followed by the remote password, followed by a newline. This option is obsolescent.

+pap
Require the peer to authenticate itself using PAP.

-pap
Don't agree to authenticate using PAP.

+chap
Require the peer to authenticate itself using CHAP [Cryptographic Handshake Authentication Protocol] authentication.

-chap
Don't agree to authenticate using CHAP.

-vj
Disable negotiation of Van Jacobson style IP header compression. This is the default above line speeds of 19200 baud, as Van Jacobson suffers from diminishing returns at high line speeds, and can be badly affected by a poor link. The main aim of VJ compression is to improve responsiveness of interactive traffic, eg typing via Telnet, over slow lines. It will not generally speed up file transfers, such as FTP.

+vj
Enable negotiation of Van Jacobson stype IP header compression. This is the default at line speeds of 19200 baud or below.

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 would use the domain option to set the domain name to Quotron.COM

modem
Use the modem control lines, ie signal to the modem using DTR, and check for hang-ups via DCD.

local
Don't use the modem control lines.

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.

name <n>
Set the name of the local system for authentication purposes to <n>.

user <u>
Set the user name to use for authenticating this machine with the peer using PAP to <u>.

usehostname
Enforce the use of the hostname as the name of the local system for authentication purposes (overrides the name option).

remotename <n>
Set the assumed name of the remote system for authentication purposes to <n>.

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).

lcp-echo-interval <n>
If this option is given, pppd will send an LCP echo-request frame to the peer every n seconds. Under Linux, the echo-request is sent when no packets have been received from the peer for 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-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-restart <n>
Set the LCP restart interval (retransmission timeout) to <n> seconds (default 3).

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

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).

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

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

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).

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

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

chap-restart <n>
Set the CHAP restart interval (retransmission timeout for challenges) to <n> seconds (default 3).

chap-max-challenge <n>
Set the maximum number of CHAP challenge transmissions to <n> (default 10).

chap-interval <n>
If this option is given, pppd will rechallenge the peer every <n> seconds.

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.

OPTIONS FILES

Options can be taken from files as well as the command line. PPP reads options from the files

InetDbase:PPP.options before looking at the command line. 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 quotes ("). A backslash (\) quotes the following character. A hash (# starts a comment, which continues until the end of the line.

AUTHENTICATION

PPP 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. In part this is provided by the InetDbase:PPP.options file, where the administrator can place options to require authenticationwhenever PPP is run, and in part by the PAP and CHAP secrets files, where the administrator can restrict the set of IP addresses which individual users may use.

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

Authentication is based on secrets, which are selected from secrets files (InetDBase:PPP.PAPSecrets for PAP, InetDBase:PPP.CHAPSecret for CHAP). Both secrets files have the same format, and both can store secrets for several combinations of server (authenticating peer) and client (peer being authenticated). Note that pppd can be both a server and client, and that different protocols can be used in the two directions if desired.

A secrets file is parsed into words as for a options file. A secret is specified by a line containing at least 3 words, in the order client, server, secret. 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, it is assumed that any IP address is OK; to disallow all IP addresses, use "-". 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.

Thus a secrets file contains both secrets for use in authenticating other hosts, plus secrets which we use for authenticating ourselves to others. Which secret to use is chosen based on the names of the host (the `local name') and its peer (the `remote name'). The local name is set as follows:

if the usehostname option is given, then the local name is the hostname of this machine (with the domain appended, if given)

else if the name option is given, then use the argument of the first name option seen

else if the local IP address is specified with a hostname, then use that name

else use the hostname of this machine (with the domain appended, if given)

When authenticating ourselves using PAP, there is also a `username' which is the local name by default, but can be set with the user option or the +ua option.

The remote name is set as follows:

if the remotename option is given, then use the argument of the last remotename option seen

else if the remote IP address is specified with a hostname, then use that host name else the remote name is the null string "".

Secrets are selected from the PAP secrets file as follows:

* For authenticating the peer, look for a secret with client == username specified in the PAP authenticate-request, and server == local name.

* For authenticating ourselves to the peer, look for a secret with client == our username, server == remote name.

When authenticating the peer with PAP, a secret of "" matches any password supplied by the peer. 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. 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.

Secrets are selected from the CHAP secrets file as follows:

* For authenticating the peer, look for a secret with client == name specified in the CHAP-Response message, and server == local name.

* For authenticating ourselves to the peer, look for a secret with client == local name, and server == name specified in the CHAP-Challenge message.

Authentication must be satisfactorily completed before IPCP (or any other Network Control Protocol) can be started. If authentication fails, 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 some cases this will be done automatically through the actions of the routed or gated daemons, but in most cases some further intervention is required.

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 PPP 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 proxy-arp 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, PPP creates a permanent, published ARP entry with the IP address of the remote host and the hardware address of the network interface found. (Not implemented yet)

EXAMPLES

In the simplest case, you can connect the serial ports of two machines and issue a command like

```
*PPPConnect InternalPC 9600 passive
```

If you are using a modem, you will need to run a dialler, such as Chat, to make the initial connection.

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 you need to authenticate yourself to the remote end, using PAP, you would use a command like

```
*PPPConnect 115200 user m.jones
```

and have a line in your PAP secrets file

```
m.jones * abcdefg
```

meaning that user m.jones should be authenticated to any host with the password abcdefg.

To close the link, issue the command

```
*PPPDisconnect
```

DIAGNOSTICS

The contents of all control packets sent or received are logged, that is, all LCP, PAP, CHAP or IPCP packets. This can be useful if the PPP negotiation does not succeed. The log can be displayed using the *PPPLog command.

The *PPPInfo command provides standard network driver statistics.

FILES

InetDBase: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 driver speed local-IP-address remote-IP-address.

This file is restricted in what it can do. In particular it should not start a new application (eg running ifconfig).

InetDBase: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 InetDBase:PPP.IP-Up script. It is invoked with the same parameters as the ip-up script. The same restrictions apply.

InetDBase:PPP.PAPSecrets
Usernames, passwords and IP addresses for PAP authentication.

InetDBase:PPP.CHAPSecret
Names, secrets and IP addresses for CHAP authentication.

InetDBase:PPP.Options
System default options for PPP, read before command-line options.

NOTES

If you wish to check the state of the PPP connection from an application, there are two things you need to check.

Using socketioctl(SIOCGIFFLAGS) will tell you whether the IP layer is up (the IFF_UP flags). To tell whether the PPP daemon is actually active you will need to use the PPP_Stats SWI to examine the running flag; the IFF_RUNNING flag from the socketioctl does not correctly indicate the state of the daemon.

SEE ALSO

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

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

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

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

RFC1548
Simpson, W.A. The Point-to-Point Protocol (PPP). 1993 December.

RFC1549
Simpson, W.A. PPP in HDLC Framing. 1993 December

AUTHORS

Drew Perkins, Brad Clements, Karl Fox, Greg Christy, Brad Parker (brad@fcr.com), Paul Mackerras (paulus@cs.anu.edu.au), Kevin Bracey