NAME

Net::NNTP - NNTP Client class

SYNOPSIS

```
use Net::NNTP;

$nntp = Net::NNTP->new("some.host.name");
$nntp->quit;

# start with SSL, e.g. nntps
$nntp = Net::NNTP->new("some.host.name", SSL => 1);

# start with plain and upgrade to SSL
$nntp = Net::NNTP->new("some.host.name");
$nntp->starttls;
```

DESCRIPTION

`Net::NNTP` is a class implementing a simple NNTP client in Perl as described in RFC977 and RFC4642. With *IO::Socket::SSL* installed it also provides support for implicit and explicit TLS encryption, i.e. NNTPS or NNTP+STARTTLS.

The Net::NNTP class is a subclass of Net::Cmd and (depending on avaibility) of IO::Socket::IP, IO::Socket::INET6 or IO::Socket::INET.

CONSTRUCTOR

new ([HOST] [, OPTIONS])

This is the constructor for a new Net::NNTP object. `HOST` is the name of the remote host to which a NNTP connection is required. If not given then it may be passed as the `Host` option described below. If no host is passed then two environment variables are checked, first `NNTPSERVER` then `NEWSHOST`, then `Net::Config` is checked, and if a host is not found then `news` is used.

`OPTIONS` are passed in a hash like fashion, using key and value pairs. Possible options are:

Host - NNTP host to connect to. It may be a single scalar, as defined for the `PeerAddr` option in *IO::Socket::INET*, or a reference to an array with hosts to try in turn. The *host* method will return the value which was used to connect to the host.

Port - port to connect to. Default - 119 for plain NNTP and 563 for immediate SSL (nntps).

SSL - If the connection should be done from start with SSL, contrary to later upgrade with `starttls`. You can use SSL arguments as documented in *IO::Socket::SSL*, but it will usually use the right arguments already.

Timeout - Maximum time, in seconds, to wait for a response from the NNTP server, a value of zero will cause all IO operations to block. (default: 120)

Debug - Enable the printing of debugging information to STDERR

Reader - If the remote server is INN then initially the connection will be to innd, by default `Net::NNTP` will issue a `MODE READER` command so that the remote server becomes nnrpd. If the `Reader` option is given with a value of zero, then this command will not be sent and the connection will be left talking to innd.

LocalAddr and **LocalPort** - These parameters are passed directly to IO::Socket to allow binding the socket to a specific local address and port.

Domain - This parameter is passed directly to IO::Socket and makes it possible to enforce IPv4 connections even if *IO::Socket::IP* is used as super class. Alternatively **Family** can be used.

METHODS

Unless otherwise stated all methods return either a *true* or *false* value, with *true* meaning that the operation was a success. When a method states that it returns a value, failure will be returned as *undef* or an empty list.

`Net::NNTP` inherits from `Net::Cmd` so methods defined in `Net::Cmd` may be used to send commands to the remote NNTP server in addition to the methods documented here.

host ()

Returns the value used by the constructor, and passed to IO::Socket::INET, to connect to the host.

starttls ()

Upgrade existing plain connection to SSL. Any arguments necessary for SSL must be given in `new` already.

article ([MSGID|MSGNUM], [FH])

Retrieve the header, a blank line, then the body (text) of the specified article.

If `FH` is specified then it is expected to be a valid filehandle and the result will be printed to it, on success a true value will be returned. If `FH` is not specified then the return value, on success, will be a reference to an array containing the article requested, each entry in the array will contain one line of the article.

If no arguments are passed then the current article in the currently selected newsgroup is fetched.

`MSGNUM` is a numeric id of an article in the current newsgroup, and will change the current article pointer. `MSGID` is the message id of an article as shown in that article's header. It is anticipated that the client will obtain the `MSGID` from a list provided by the `newnews` command, from references contained within another article, or from the message-id provided in the response to some other commands.

If there is an error then `undef` will be returned.

body ([MSGID|MSGNUM], [FH])

Like `article` but only fetches the body of the article.

head ([MSGID|MSGNUM], [FH])

Like `article` but only fetches the headers for the article.

articlefh ([MSGID|MSGNUM])

bodyfh ([MSGID|MSGNUM])

headfh ([MSGID|MSGNUM])

These are similar to article(), body() and head(), but rather than returning the requested data directly, they return a tied filehandle from which to read the article.

nntpstat ([MSGID|MSGNUM])

The `nntpstat` command is similar to the `article` command except that no text is returned. When selecting by message number within a group, the `nntpstat` command serves to set the "current article pointer" without sending text.

Using the `nntpstat` command to select by message-id is valid but of questionable value, since a selection by message-id does **not** alter the "current article pointer".

Returns the message-id of the "current article".

group ([GROUP])

Set and/or get the current group. If `GROUP` is not given then information is returned on the

current group.

In a scalar context it returns the group name.

In an array context the return value is a list containing, the number of articles in the group, the number of the first article, the number of the last article and the group name.

help ( )

Request help text (a short summary of commands that are understood by this implementation) from the server. Returns the text or undef upon failure.

ihave (MSGID [, MESSAGE])

The `ihave` command informs the server that the client has an article whose id is `MSGID`. If the server desires a copy of that article and `MESSAGE` has been given then it will be sent.

Returns *true* if the server desires the article and `MESSAGE` was successfully sent, if specified.

If `MESSAGE` is not specified then the message must be sent using the `datasend` and `dataend` methods from *Net::Cmd*

`MESSAGE` can be either an array of lines or a reference to an array and must be encoded by the caller to octets of whatever encoding is required, e.g. by using the Encode module's `encode()` function.

last ()

Set the "current article pointer" to the previous article in the current newsgroup.

Returns the message-id of the article.

date ()

Returns the date on the remote server. This date will be in a UNIX time format (seconds since 1970)

postok ()

`postok` will return *true* if the servers initial response indicated that it will allow posting.

authinfo (USER, PASS)

Authenticates to the server (using the original AUTHINFO USER / AUTHINFO PASS form, defined in RFC2980) using the supplied username and password. Please note that the password is sent in clear text to the server. This command should not be used with valuable passwords unless the connection to the server is somehow protected.

authinfo_simple (USER, PASS)

Authenticates to the server (using the proposed NNTP V2 AUTHINFO SIMPLE form, defined and deprecated in RFC2980) using the supplied username and password. As with *authinfo* the password is sent in clear text.

list ()

Obtain information about all the active newsgroups. The results is a reference to a hash where the key is a group name and each value is a reference to an array. The elements in this array are:- the last article number in the group, the first article number in the group and any information flags about the group.

newgroups (SINCE [, DISTRIBUTIONS])

`SINCE` is a time value and `DISTRIBUTIONS` is either a distribution pattern or a reference to a list of distribution patterns. The result is the same as `list`, but the groups return will be limited to those created after `SINCE` and, if specified, in one of the distribution areas in `DISTRIBUTIONS`.

newnews (SINCE [, GROUPS [, DISTRIBUTIONS]])

SINCE is a time value. GROUPS is either a group pattern or a reference to a list of group patterns. DISTRIBUTIONS is either a distribution pattern or a reference to a list of distribution patterns.

Returns a reference to a list which contains the message-ids of all news posted after SINCE, that are in a groups which matched GROUPS and a distribution which matches DISTRIBUTIONS.

next ()

Set the "current article pointer" to the next article in the current newsgroup.

Returns the message-id of the article.

post ([MESSAGE])

Post a new article to the news server. If MESSAGE is specified and posting is allowed then the message will be sent.

If MESSAGE is not specified then the message must be sent using the datasend and dataend methods from *Net::Cmd*

MESSAGE can be either an array of lines or a reference to an array and must be encoded by the caller to octets of whatever encoding is required, e.g. by using the Encode module's encode( ) function.

The message, either sent via datasend or as the MESSAGE parameter, must be in the format as described by RFC822 and must contain From:, Newsgroups: and Subject: headers.

postfh ()

Post a new article to the news server using a tied filehandle. If posting is allowed, this method will return a tied filehandle that you can print() the contents of the article to be posted. You must explicitly close() the filehandle when you are finished posting the article, and the return value from the close() call will indicate whether the message was successfully posted.

slave ()

Tell the remote server that I am not a user client, but probably another news server.

quit ()

Quit the remote server and close the socket connection.

can_inet6 ()

Returns whether we can use IPv6.

can_ssl ()

Returns whether we can use SSL.

Extension methods

These methods use commands that are not part of the RFC977 documentation. Some servers may not support all of them.

newsgroups ([PATTERN])

Returns a reference to a hash where the keys are all the group names which match PATTERN, or all of the groups if no pattern is specified, and each value contains the description text for the group.

distributions ()

Returns a reference to a hash where the keys are all the possible distribution names and the values are the distribution descriptions.

distribution_patterns ()

Returns a reference to an array where each element, itself an array reference, consists of the three fields of a line of the distrib.pats list maintained by some NNTP servers, namely: a weight, a wildmat and a value which the client may use to construct a Distribution header.

subscriptions ()

Returns a reference to a list which contains a list of groups which are recommended for a new user to subscribe to.

overview_fmt ()

Returns a reference to an array which contain the names of the fields returned by `xover`.

active_times ()

Returns a reference to a hash where the keys are the group names and each value is a reference to an array containing the time the groups was created and an identifier, possibly an Email address, of the creator.

active ([PATTERN])

Similar to `list` but only active groups that match the pattern are returned. `PATTERN` can be a group pattern.

xgtitle (PATTERN)

Returns a reference to a hash where the keys are all the group names which match `PATTERN` and each value is the description text for the group.

xhdr (HEADER, MESSAGE-SPEC)

Obtain the header field `HEADER` for all the messages specified.

The return value will be a reference to a hash where the keys are the message numbers and each value contains the text of the requested header for that message.

xover (MESSAGE-SPEC)

The return value will be a reference to a hash where the keys are the message numbers and each value contains a reference to an array which contains the overview fields for that message.

The names of the fields can be obtained by calling `overview_fmt`.

xpath (MESSAGE-ID)

Returns the path name to the file on the server which contains the specified message.

xpat (HEADER, PATTERN, MESSAGE-SPEC)

The result is the same as `xhdr` except the is will be restricted to headers where the text of the header matches `PATTERN`

xrover ()

The XROVER command returns reference information for the article(s) specified.

Returns a reference to a HASH where the keys are the message numbers and the values are the References: lines from the articles

listgroup ([GROUP])

Returns a reference to a list of all the active messages in `GROUP`, or the current group if `GROUP` is not specified.

reader ()

Tell the server that you are a reader and not another server.

This is required by some servers. For example if you are connecting to an INN server and you have transfer permission your connection will be connected to the transfer daemon, not the

NNTP daemon. Issuing this command will cause the transfer daemon to hand over control to the NNTP daemon.

Some servers do not understand this command, but issuing it and ignoring the response is harmless.

UNSUPPORTED

The following NNTP command are unsupported by the package, and there are no plans to do so.

```
AUTHINFO GENERIC
XTHREAD
XSEARCH
XINDEX
```

DEFINITIONS

MESSAGE-SPEC

MESSAGE-SPEC is either a single message-id, a single message number, or a reference to a list of two message numbers.

If MESSAGE-SPEC is a reference to a list of two message numbers and the second number in a range is less than or equal to the first then the range represents all messages in the group after the first message number.

NOTE For compatibility reasons only with earlier versions of Net::NNTP a message spec can be passed as a list of two numbers, this is deprecated and a reference to the list should now be passed

PATTERN

The NNTP protocol uses the WILDMAT format for patterns. The WILDMAT format was first developed by Rich Salz based on the format used in the UNIX "find" command to articulate file names. It was developed to provide a uniform mechanism for matching patterns in the same manner that the UNIX shell matches filenames.

Patterns are implicitly anchored at the beginning and end of each string when testing for a match.

There are five pattern matching operations other than a strict one-to-one match between the pattern and the source to be checked for a match.

The first is an asterisk * to match any sequence of zero or more characters.

The second is a question mark ? to match any single character. The third specifies a specific set of characters.

The set is specified as a list of characters, or as a range of characters where the beginning and end of the range are separated by a minus (or dash) character, or as any combination of lists and ranges. The dash can also be included in the set as a character it if is the beginning or end of the set. This set is enclosed in square brackets. The close square bracket] may be used in a set if it is the first character in the set.

The fourth operation is the same as the logical not of the third operation and is specified the same way as the third with the addition of a caret character ^ at the beginning of the test string just inside the open square bracket.

The final operation uses the backslash character to invalidate the special meaning of an open square bracket [, the asterisk, backslash or the question mark. Two backslashes in sequence will result in the evaluation of the backslash as a character with no special meaning.

Examples

[^]-]

matches any single character other than a close square bracket or a minus sign/dash.

`*bdc`

matches any string that ends with the string "bdc" including the string "bdc" (without quotes).

`[0-9a-zA-Z]`

matches any single printable alphanumeric ASCII character.

`a??d`

matches any four character string which begins with a and ends with d.

SEE ALSO

Net::Cmd, *IO::Socket::SSL*

AUTHOR

Graham Barr <*gbarr@pobox.com*>

Steve Hay <*shay@cpan.org*> is now maintaining libnet as of version 1.22_02

COPYRIGHT