

NAME

Net::Cmd - Network Command class (as used by FTP, SMTP etc)

SYNOPSIS

```
use Net::Cmd;

@ISA = qw(Net::Cmd);
```

DESCRIPTION

Net::Cmd is a collection of methods that can be inherited by a sub-class of IO::Socket::INET. These methods implement the functionality required for a command based protocol, for example FTP and SMTP.

If your sub-class does not also derive from IO::Socket::INET or similar (e.g. IO::Socket::IP, IO::Socket::INET6 or IO::Socket::SSL) then you must provide the following methods by other means yourself: `close()` and `timeout()`.

USER METHODS

These methods provide a user interface to the Net::Cmd object.

`debug (VALUE)`

Set the level of debug information for this object. If `VALUE` is not given then the current state is returned. Otherwise the state is changed to `VALUE` and the previous state returned.

Different packages may implement different levels of debug but a non-zero value results in copies of all commands and responses also being sent to `STDERR`.

If `VALUE` is `undef` then the debug level will be set to the default debug level for the class.

This method can also be called as a *static* method to set/get the default debug level for a given class.

`message ()`

Returns the text message returned from the last command. In a scalar context it returns a single string, in a list context it will return each line as a separate element. (See *PSEUDO RESPONSES* below.)

`code ()`

Returns the 3-digit code from the last command. If a command is pending then the value 0 is returned. (See *PSEUDO RESPONSES* below.)

`ok ()`

Returns non-zero if the last code value was greater than zero and less than 400. This holds true for most command servers. Servers where this does not hold may override this method.

`status ()`

Returns the most significant digit of the current status code. If a command is pending then `CMD_PENDING` is returned.

`datasend (DATA)`

Send data to the remote server, converting LF to CRLF. Any line starting with a '.' will be prefixed with another '.'. `DATA` may be an array or a reference to an array. The `DATA` passed in must be encoded by the caller to octets of whatever encoding is required, e.g. by using the Encode module's `encode()` function.

`dataend ()`

End the sending of data to the remote server. This is done by ensuring that the data already

sent ends with CRLF then sending '.CRLF' to end the transmission. Once this data has been sent `dataend` calls `response` and returns true if `response` returns `CMD_OK`.

CLASS METHODS

These methods are not intended to be called by the user, but used or over-ridden by a sub-class of `Net::Cmd`

`debug_print (DIR, TEXT)`

Print debugging information. `DIR` denotes the direction *true* being data being sent to the server. Calls `debug_text` before printing to `STDERR`.

`debug_text (DIR, TEXT)`

This method is called to print debugging information. `TEXT` is the text being sent. The method should return the text to be printed.

This is primarily meant for the use of modules such as FTP where passwords are sent, but we do not want to display them in the debugging information.

`command (CMD [, ARGS, ...])`

Send a command to the command server. All arguments are first joined with a space character and CRLF is appended, this string is then sent to the command server.

Returns `undef` upon failure.

`unsupported ()`

Sets the status code to 580 and the response text to 'Unsupported command'. Returns zero.

`response ()`

Obtain a response from the server. Upon success the most significant digit of the status code is returned. Upon failure, timeout etc., `CMD_ERROR` is returned.

`parse_response (TEXT)`

This method is called by `response` as a method with one argument. It should return an array of 2 values, the 3-digit status code and a flag which is true when this is part of a multi-line response and this line is not the last.

`getline ()`

Retrieve one line, delimited by CRLF, from the remote server. Returns *undef* upon failure.

NOTE: If you do use this method for any reason, please remember to add some `debug_print` calls into your method.

`ungetline (TEXT)`

Unget a line of text from the server.

`rawdatasend (DATA)`

Send data to the remote server without performing any conversions. `DATA` is a scalar. As with `datasend ()`, the `DATA` passed in must be encoded by the caller to octets of whatever encoding is required, e.g. by using the Encode module's `encode ()` function.

`read_until_dot ()`

Read data from the remote server until a line consisting of a single '.'. Any lines starting with '.' will have one of the '.'s removed.

Returns a reference to a list containing the lines, or *undef* upon failure.

`tied_fh ()`

Returns a filehandle tied to the `Net::Cmd` object. After issuing a command, you may read from this filehandle using `read()` or `<>`. The filehandle will return EOF when the final dot is

encountered. Similarly, you may write to the filehandle in order to send data to the server after issuing a command that expects data to be written.

See the Net::POP3 and Net::SMTP modules for examples of this.

PSEUDO RESPONSES

Normally the values returned by `message()` and `code()` are obtained from the remote server, but in a few circumstances, as detailed below, `Net::Cmd` will return values that it sets. You can alter this behavior by overriding `DEF_REPLY_CODE()` to specify a different default reply code, or overriding one of the specific error handling methods below.

Initial value

Before any command has executed or if an unexpected error occurs `code()` will return "421" (temporary connection failure) and `message()` will return undef.

Connection closed

If the underlying `IO::Handle` is closed, or if there are any read or write failures, the file handle will be forced closed, and `code()` will return "421" (temporary connection failure) and `message()` will return "`[$pkg]` Connection closed" (where `$pkg` is the name of the class that subclassed `Net::Cmd`). The `_set_status_closed()` method can be overridden to set a different message (by calling `set_status()`) or otherwise trap this error.

Timeout

If there is a read or write timeout `code()` will return "421" (temporary connection failure) and `message()` will return "`[$pkg]` Timeout" (where `$pkg` is the name of the class that subclassed `Net::Cmd`). The `_set_status_timeout()` method can be overridden to set a different message (by calling `set_status()`) or otherwise trap this error.

EXPORTS

`Net::Cmd` exports six subroutines, five of these, `CMD_INFO`, `CMD_OK`, `CMD_MORE`, `CMD_REJECT` and `CMD_ERROR`, correspond to possible results of `response` and `status`. The sixth is `CMD_PENDING`.

AUTHOR

Graham Barr <gbarr@pobox.com>

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

COPYRIGHT

Versions up to 2.29_1 Copyright (c) 1995-2006 Graham Barr. All rights reserved. Changes in Version 2.29_2 onwards Copyright (C) 2013-2014 Steve Hay. All rights reserved.

This module is free software; you can redistribute it and/or modify it under the same terms as Perl itself, i.e. under the terms of either the GNU General Public License or the Artistic License, as specified in the *LICENCE* file.