



Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'Glib::Log.3pm'

\$ man Glib::Log.3pm

Glib::Log(3pm) User Contributed Perl Documentation Glib::Log(3pm)

NAME

Glib::Log - A flexible logging mechanism

METHODS

scalar = Glib::Log->set_always_fatal (\$fatal_mask)

? \$fatal_mask (scalar)

Glib->critical (\$domain, \$message)

? \$domain (string or undef)

? \$message (string)

Glib->debug (\$domain, \$message)

? \$domain (string or undef)

? \$message (string)

Glib::Log::default_handler (\$log_domain, \$log_level, \$message, ...)

? \$log_domain (string)

? \$log_level (scalar)

? \$message (string)

? ... (list) possible "userdata" argument ignored

The arguments are the same as taken by the function for `set_handler` or `set_default_handler`.

```
prev_log_func = Glib::Log->set_default_handler ($log_func, $user_data)
```

? \$log_func (subroutine) handler function or undef

? \$user_data (scalar)

Install `log_func` as the default log handler. `log_func` is called for anything which doesn't otherwise have a handler (either `Glib::Log->set_handler`, or the `Glib::xsapi_gperl_handle_logs_for`),

```
&$log_func ($log_domain, $log_levels, $message, $user_data)
```

where `$log_domain` is a string, and `$log_levels` is a `Glib::LogLevelFlags` of level and flags being reported.

If `log_func` is `\&Glib::Log::default_handler` or `undef` then Glib's default handler is set.

The return value from "set_default_handler" is the previous handler. This is `\&Glib::Log::default_handler` for Glib's default, otherwise a Perl function previously installed. If the handler is some other non-Perl function then currently the return is `undef`, but perhaps that will change to some wrapped thing, except that without associated `userdata` there's very little which could be done with it (it couldn't be reinstalled later without its `userdata`).

Since: glib 2.6

Glib->error (\$domain, \$message)

? \$domain (string or undef)

? \$message (string)

scalar = Glib::Log->set_fatal_mask (\$log_domain, \$fatal_mask)

? \$log_domain (string)

? \$fatal_mask (scalar)

integer = Glib::Log->set_handler (\$log_domain, \$log_levels, \$log_func, \$user_data=undef)

? \$log_domain (string or undef) name of the domain to handle with this callback.

? \$log_levels (Glib::LogLevelFlags) log levels to handle with this callback

? \$log_func (subroutine) handler function

? \$user_data (scalar)

\$log_func will be called as

```
&$log_func ($log_domain, $log_levels, $message, $user_data);
```

where \$log_domain is the name requested and \$log_levels is a Glib::LogLevelFlags of level and flags being reported.

Glib->info (\$domain, \$message)

? \$domain (string or undef)

? \$message (string)

Glib->log (\$log_domain, \$log_level, \$message)

? \$log_domain (string or undef)

? \$log_level (scalar)

? \$message (string)

Glib->message (\$domain, \$message)

? \$domain (string or undef)

? \$message (string)

Glib::Log->remove_handler (\$log_domain, \$handler_id)

? \$log_domain (string or undef)

? \$handler_id (integer) as returned by "set_handler"

Glib->warning (\$domain, \$message)

? \$domain (string or undef)

? \$message (string)

ENUMS AND FLAGS

flags Glib::LogLevelFlags

? 'recursion' / 'G_LOG_FLAG_RECURSION'

? 'fatal' / 'G_LOG_FLAG_FATAL'

? 'error' / 'G_LOG_LEVEL_ERROR'

? 'critical' / 'G_LOG_LEVEL_CRITICAL'

? 'warning' / 'G_LOG_LEVEL_WARNING'

? 'message' / 'G_LOG_LEVEL_MESSAGE'

? 'info' / 'G_LOG_LEVEL_INFO'

? 'debug' / 'G_LOG_LEVEL_DEBUG'

? 'fatal-mask' / 'G_LOG_FATAL_MASK'

SEE ALSO

Glib

COPYRIGHT

Copyright (C) 2003-2011 by the gtk2-perl team.

This software is licensed under the LGPL. See Glib for a full notice.

perl v5.34.0

2022-02-06

Glib::Log(3pm)