TPLOGCONFIG(3)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SYNOPSIS

#include <ndebug.h> #include <nerror.h>

int tplogconfig(int *logger*, int *lev*, char **debug_string*, char **module*, char **new_file*);

Link with *-lnstd -lpthread -lrt -lm*

Chapter 2

DESCRIPTION

Function does configure logging facilities - **NDRX** (XATMI internal logs), **UBF** (UBF internal logs(and **TP** (User logs). If not already logger started, then this function will initiate Enduro/X framework to load the logging settings from [@debug] ini section or from **ndrxdebug.conf(5)**. Then with help of this function user is able to override loaded settings.

Also it is possible to set per thread logging, if facility code used here is **LOG_FACILITY_TP_THREAD**. Or it is possible to configure request based logging from this function, but **tplogsetreqfile_direct(3)** and **tplogsetreqfile(3)** is recommended to use instead.

The logging facility is set in *logger* parameter, you may unify settings for multiple facilities with bitwise OR. Th facilities defined are following:

```
#define LOG_FACILITY_NDRX       0x00001 /* settings for ATMI logging             */
#define LOG_FACILITY_UBF        0x00002 /* settings for UBF logging              */
#define LOG_FACILITY_TP         0x00004 /* settings for TP logging               */
#define LOG_FACILITY_TP_THREAD  0x00008 /* settings for TP, thread based logging */
#define LOG_FACILITY_TP_REQUEST 0x00010 /* Request logging, thread based         */
```

logger is mandatory parameter to the function. To configure logging level for given facilities, you may use *lev* to indicate one of the following levels:

```
#define log_always      1
#define log_error       2
#define log_warn        3
#define log_info        4
#define log_debug       5
#define log_dump        6 /* this is un-offical level, normally use 1..5 */
```

lev is optional parameter, and if not used, then set it to *-1*. Function accepts optional parameter *debug_string*, which is Enduro/X standard debug string. If field not set then it might be NULL or empty (EOS). If set, then it contains the configuration settings described in **ndrxdebug.conf(5)** or see **ex_devguide(guides)**. One special note here is that, with debug string you may set log levels for **NDRX**, **UBF** and **TP** facilities, thus **NDRX** and **UBF** log levels will affect the per-process common settings, i.e. they are common for thread and request logging. However **TP** will change the level of the given logger facility, per process (**LOG_FACILITY_TP**), per thread (**LOG_FACILITY_TP_THREAD**) or per request (**LOG_FACILITY_TP_REQUEST**).

If the *lev* parameter is present and it will override the given loggers settings from *debug_string*. The same applies if *debug_string* contains the logging file, the *new_file* is present (not NULL and not EOS) will override the output file setting if **file** param is present in *debug_string*.

module module indicates the module code string which is plotted in each logged line. This is valid only for **TP** loggers for **NDRX** and **UBF** it is ignored.

Chapter 3

RETURN VALUE

On success, **tplogconfig()** returns 0. On error, -1 is returned, with **Nerror** set to indicate the error.

Chapter 4

ERRORS

Note that **Nstrerror()** returns generic error message plus custom message with debug info from last function call.

NEFORMAT Invalid format for *debug_string*.

NESYSTEM System error occurred. See the logs for more info.

Chapter 5

EXAMPLE

See **atmitest/test031_logging/atmiclt31.c** for sample code.

Chapter 6

BUGS

Report bugs to support@mavimax.com

Chapter 7

SEE ALSO

tplogdump(3) tplogdumpdiff(3) tplog(3) tplogsetreqfile_direct(3) tplogsetreqfile(3) ex_devguide(guides) ndrxdebug.conf(5) tplogqinfo(3)

Chapter 8

COPYING