TPENQUEUE(3)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SYNOPSIS

#include <atmi.h>

int tpenqueue (char *_qspace_, char *_qname_, TPQCTL *_ctl_, char *_data_, long _len_, long _flags_);

int tpenqueueex (short _nodeid_, short _srvid_, char *_qname_, TPQCTL *_ctl_, char *_data_, long _len_, long _flags_);

For XATMI client link with *-latmiclt -latmi -lubf -lnstd -lpthread -lrt -lm*

For XATMI server link with *-latmisrv|-latmisrvnomain|-latmisrvinteg -latmi -lubf -lnstd -lpthread -lrt -lm*

Chapter 2

DESCRIPTION

Enqueue message to transactional message queue. Queue subsystem (see **tmqueue(8)** and **persistent_message_queues_overview(guide** must be configured before using this function. *qspace* is queue space name (logical queue sub-system name, that can be shared between multiple **tmqueue** servers). *qname* is queue name, *ctl* is control structure that contains various details for message, how it shall be enqueued. Also *ctl* returns some diagnostics information. *data*/*len* pair is XATMI buffer containing the message that must be enqueued. **tpenqueueex()** allows to specify exactly which **tmqueue** server will process the request, by giving the Enduro/X cluster node id in *nodeid* field and *srvid* from *ndrxconfig.xml(5).

Functions are transactional and can participate in user's global transaction. It is not possible to enqueue and dequeue message within same global transaction.

Valid flags

TPNOTRAN Do not run the equeue in the users global transaction. In this case **tmqueue** will open it's own transaction, and will commit transaction upon return.

TPSIGRSTRT Restart the system call in progress if interrupted by signal handler. This affects only underlaying mq_* function calls.

TPNOTIME Ignore timeout setting (**NDRX_TOUT** env variable). Wait for reply for infinitely.

TPNOBLOCK In case of queue server request, if request IPC queue is full, do not wait on queue, but return error. The error code for this situation is **TPEBLOCK**. This affects only request part of the call. This flag does not affect waiting for response from the TMQ server.

The **TPQCTL** structure is following:

```
/* Queue support structure: */
struct tpqctl_t
{
        long flags;             /* indicates which of the values are set */
        long deq_time;          /* absolute/relative  time for dequeuing */
        long priority;          /* enqueue priority */
        long diagnostic;        /* indicates reason for failure */
        char diagmsg[NDRX_QDIAG_MSG_SIZE]; /* diagnostic message */
        char msgid[TMMSGIDLEN]; /* id of message before which to queue */
        char corrid[TMCORRIDLEN];/* correlation id used to identify message */
        char replyqueue[TMQNAMELEN+1];  /* queue name for reply message */
        char failurequeue[TMQNAMELEN+1];/* queue name for failure message */
        CLIENTID cltid;         /* client identifier for originating client */
        long urcode;            /* application user-return code */
        long appkey;            /* application authentication client key */
        long delivery_qos;      /* delivery quality of service  */
        long reply_qos;         /* reply message quality of service  */
        long exp_time;          /* expiration time  */
};
typedef struct tpqctl_t TPQCTL;
```

Valid ***TPQCTL.flags***

TPNOFLAGS No flags set.

TPQCORRID Use the correlator ID set in *TPQCTL.corrid*. Note that each byte in *corrid* is significant.

TPQREPLYQ Use the *replyqueue* for submitting response messages of services which were invoked for automatic queues. If this flag is set, with response buffer message will be enqueued to *replyqueue*.

TPQFAILUREQ Send the message to *failurequeue* if **tpcall()** for service invoke for automatic queues failed.

Fields *TPQCTL.deq_time*, *TPQCTL.priority*, *TPQCTL.urcode*, *TPQCTL.appkey*, *TPQCTL.delivery_qos*, *TPQCTL.reply_qos*, *TPQCTL.exp_time* are reserved for future use. *TPQCTL.cltid* is automatically set by Enduro/X system when passing message to **tmqueue** server.

Chapter 3

RETURN VALUE

On success, **tpenqueue()** return 0; on error, -1 is returned, with **tperrno** set to indicate the error. Also *TPQCTL.diagnostic* and *TPQCTL.diagmsg* will contain additional information.

Chapter 4

ERRORS

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

TPEINVAL *data* is NULL, *qspace* is NULL, or *nodeid* and *srvid* is 0. Error can be generate in case if *qname* is empty or NULL. *ctl* is NULL or *data* does not point to **tpalloc()** allocated buffer.

TPENOENT Tmqueue server is not available.

TPETIME Service did not reply in given time (*NDRX_TOUT*).

TPEDIAGNOSTIC More information is provided in *TPQCTL.diagnostic* field.

TPESVCFAIL Tmqueue Service returned *TPFAIL*. This is application level failure.

TPESVCERR Tmqueue service got system level failure. Server died during the message presence in service queue.

TPESYSTEM Enduro/X internal error occurred. See logs for more info.

TPEOS System failure occurred during serving. See logs i.e. user log, or debugs for more info.

Values for *TPQCTL.diagnostic*

QMEINVAL Invalid request bugffer.

QMEOS Operating system problems. Might be insufficient memory.

QMESYSTEM Enduro/X internal problems. Might be issues with saving messages to disk.

TPEBLOCK XATMI IPC Service request queue was full and **TPNOBLOCK** flag was specified.

Chapter 5

EXAMPLE

See **atmitest/test028_tmq/atmiclt28.c** for sample code.

Chapter 6

BUGS

Report bugs to support@mavimax.com

Chapter 7

SEE ALSO

tpdequeue(3) tpdequeueex(3) tmqueue(8) persistent_message_queues_overview(guides)

Chapter 8

COPYING

© Mavimax, Ltd