

**TPDEQUEUE(3)**

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>SYNOPSIS</b>	<b>1</b>
<b>2</b>	<b>DESCRIPTION</b>	<b>2</b>
<b>3</b>	<b>RETURN VALUE</b>	<b>4</b>
<b>4</b>	<b>ERRORS</b>	<b>5</b>
<b>5</b>	<b>EXAMPLE</b>	<b>6</b>
<b>6</b>	<b>BUGS</b>	<b>7</b>
<b>7</b>	<b>SEE ALSO</b>	<b>8</b>
<b>8</b>	<b>COPYING</b>	<b>9</b>

## Chapter 1

# SYNOPSIS

```
#include <atmi.h>
```

```
int tpdequeue (char *qspace, char *qname, TPQCTL *ctl, char **data, long *len, long flags);
```

```
int tpdequeueex (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 *-latmisrvl -latmisrvnomainl -latmisrvinteg -latmi -lubf -lnstd -lpthread -lrt -lm*

## Chapter 2

# DESCRIPTION

Dequeue message from transactional persistent message queue. Message queue subsystem (see **tmqueue(8)** and **persistent\_message\_queue(8)**) 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 dequeue and additional meta data. Also *ctl* returns diagnostics information in case of error. *data* pair is XATMI buffer to which dequeued message must be written. Buffer will be reallocated accordingly to match the message type in queue. **tpdequeueex()** allows to specify exactly which **tmqueue** server will process the request, by giving the Enduro/X cluster node id in *nodeid* field and *srv*id 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 enqueue in the users global transaction. In this case **tmqueue** will open it's own transaction, and will commit transaction upon return.

**TPSIGRSTR** Restart the system call in progress if interrupted by signal handler. This affects only underlying mq\_\* function calls.

**TPNOTIME** Ignore timeout setting (**NDRX\_TOUT** env variable). Wait for reply for infinitely.

**TPNOCHANGE** Do not allow to change the buffer type if buffer type in message queue does not match the buffer type passed into *data* field.

**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[TMSGIDLEN];     /* 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.

**TPQGETBYCORRID** Use the correlator ID in *TPQCTL.corrid* to get the message by. Note that each byte in *corrid* is significant.

**TPQGETBYMSGID** Use the message ID in *TPQCTL.msgid* to get the message by. Note that each byte in *msgid* is significant.

**TPQPEEK** Return the message from queue by not actually removing it.

**TPNOABORT** Do not abort global transaction, even if queue failed (could be suitable in cases when for example polling on Q is done (request for next msg and queue is empty)).

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 set to client id string which enqueued the message.

## Chapter 3

# RETURN VALUE

On success, **tpdequeue()** 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.

**QMENOMSG** No messages in queue.

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



## Chapter 5

# EXAMPLE

See `atmitest/test028_tmq/atmict28.c` for sample code.

## Chapter 6

# BUGS

Report bugs to [support@mavimax.com](mailto:support@mavimax.com)

## Chapter 7

## SEE ALSO

[tpdenqueue\(3\)](#) [tpenqueueex\(3\)](#) [tmqueue\(8\)](#) [persistent\\_message\\_queues\\_overview\(guides\)](#)

## Chapter 8

# COPYING

© Mavimax, Ltd