# TPSUBSCRIBE(3)

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| | | | |

# Contents

# Chapter 1

# SYNOPSIS

#include <atmi.h>

long tpsubscribe (char **eventexpr*, char **filter*, TPEVCTL **ctl*, long *flags*);

Link with *-latmisrv|-latmisrvnomain|-latmisrvinteg -latmi -lubf -lnstd -lpthread -lrt -lm*

# Chapter 2

# DESCRIPTION

Subscribe to event. This is API function for Enduro/X event broker services. In order to get the functionality working, then event server must be started and configured. See **tpevsrv(8)** manpage. *eventexpr* is regular expression syntax based expression which is registered into event broker, so that if it is matched then event is posted to service, written in *ctl→name1* field. Note that many servers with the same service can subscribe to same event. All server's services will receive the event. *filter* is boolean expression for **UBF** or regular expression for **STRING** buffer which are extra expression to be tested, when event name matched. If *filter* is true or **NULL**, then event is posted to service.

The **TPEVCTL** structure is following:

```
struct tpevctl_t
{
    long flags;
    char name1[XATMI_SERVICE_NAME_LENGTH];
    char name2[XATMI_SERVICE_NAME_LENGTH];
};
typedef struct tpevctl_t TPEVCTL;
```

*ctl→name2* is reserved for future use.

**NOTE:** that if multiple server binaries are advertising some service lets say "MYSVC" and event is subscribed for any of the server binaries, then if event is triggered, it will be delivered to "MYSVC" service as a one message. There is guarantee to which copy message will pass. Thus if all servers needs to be notified with the event, each server shall advertise unique server name and subscribe it to particular event. In that case matching event will be delivered to all interested executables. For example:

```
int tpsvrinit(int argc, char **argv)
{
    TPEVCTL evctl;
    char svcnm[XATMI_SERVICE_NAME_LENGTH+1];

    snprintf(svcnm, sizeof(svcnm), "MYSVC_%d", tpgetsrvid())

    if (EXSUCCEED!=tpadvertise(svcnm, MYSVC))
    {
        NDRX_LOG(log_error, "Failed to advertise %s!", svcnm);
        ret=EXFAIL;
        goto out;
    }

    memset(&evctl, 0, sizeof(evctl));

    NDRX_STRCPY_SAFE(evctl.name1, svcnm);
    evctl.flags|=TPEVSERVICE;

    /* Subscribe to event server */
```

```
    if (EXFAIL==tpsubscribe("EVTEST", NULL, &evctl, 0L))
    {
        NDRX_LOG(log_error, "Failed to subscribe to EVTEST: %s",
                                        tpstrerror(tperror));
        ret=EXFAIL;
        goto out;
    }

out:
    return ret;
}
```

**Valid flags**

**TPNOTRAN** Do not call service in transaction mode. This is effective in case if caller process is running in transaction mode, but destination process shall not run in the same global transaction

**TPSIGRSTRT** 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.

**Valid ctl→flags**

**TPEVSERVICE** Call the service. Note that service will be called with flags which are present at *ctl→flags* | **TPNOREPLY**.

**TPEVPERSIST** Do not remove service for event brokers registry, if service is not available any more (detected at point **tpcall**() to service failed).

This function is available only for XATMI servers.

# Chapter 3

# RETURN VALUE

On success, **tpsubscribe()** return subscription id; on error, -1 is returned, with **tperrno** set to indicate the error.

# Chapter 4

# ERRORS

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

**TPEINVAL** Invalid parameter is given to function. Event expression is **NULL** or empty. Event expression is longer than 255 bytes, *ctl* is NULL, or *clt→name1* is empty or filter is set and longer than 255 bytes.

**TPENOENT** Event server is not available.

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

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

**TPESVCERR** System level service failure. Server died during the message presence in service queue.

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

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

# Chapter 5

# BUGS

Report bugs to support@mavimax.com

# Chapter 6

# EXAMPLE

See **atmitest/test004_basicevent/atmisv4_1ST.c** for sample code.

# Chapter 7

# SEE ALSO

**tpevsrv(8) tpunsubscribe(3) tppost(3)**

# Chapter 8

# COPYING