TPSETUNSOL(3)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SYNOPSIS

#include <atmi.h>

void (**tpsetunsol (void (***disp*) (char **data*, long *len*, long *flags*))) (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

Functions sets unsolicited message handler. On the return function returns previous handler. When system is initialized, the handler is set to **NULL** thus no callback function is registered for receiving unsolicited messages. When unsolicited message handling is needed, then *disp* argument must be set to non NULL function. The signature for the function is:

```
void notification_callback (char *data, long len, long flags);
```

The notifications are received by this function, when somebody sends **tpnotify(3)** or **tpbroadcast(3)** to this client, and the **tpsetunsol()** handler is set. In that case the notification_callback callback function is invoked with XATMI allocated buffer in variable *data* and *len*. The user shall not free the buffer, as it is automatically freed when function finishes processing. The *flags* parameter currently is not used.

Callbacks are processed in case if in progress **tpcall(3)** or **tpgetrply(3)** receives the notification. In that case callback is invoked. The other option to process notification is that user process manually calls **tpchkunsol(3)** at some time interval.

The processing that can be done inside the callback is limited to following functions:

1. tpalloc()
2. tpfree()
3. tpgetlev()
4. tprealloc()
5. tptypes()

So basically if some significant processing is required within the callback, the user shall allocate new XATMI buffer, copy the data to it and create new thread. The copied data shall be passed to the thread for processing. And in the end callback can do return.

Enduro/X does not strictly control what API is used within the callback. It is up to user to follow these rules, otherwise unexpected behavior might occur.

Chapter 3

RETURN VALUE

On success, **tpsetunsol()** return previous function pointer, might be NULL too; on error, TPUNSOLERR 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 Environment variables not configured, see **ex_env(5)** page.

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

EXAMPLE

See **atmitest/test038_tpnotify/atmiclt38.c** for sample code.

Chapter 6

BUGS

Report bugs to support@mavimax.com

Chapter 7

SEE ALSO

tpnotify(3) tpbroadcast(3) tpchkunsol(3) tpinit(3)

Chapter 8

COPYING