tpsrvgetctxdata(3)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SYNOPSIS

#include <atmi.h>

char * tpsrvgetctxdata (void);

char * tpsrvgetctxdata2 (char *p_buf, long *p_len);

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

Chapter 2

DESCRIPTION

Get server context data. This function is intended to be used by XATMI server function. When main thread service dispatcher invokes the service function, by using **tpsrvgetctxdata()** it is possible to flip over the work load to thread. Function allocates memory block by **malloc()** and fills with the thread data. If global transaction is started, then it is suspended and transaction data is written to returned context data.

If the size of allocated memory block is needed to be know, then **tpsrvgetctxdata2()** can be used, for which *p_len* is mandatory attributed. In case if *p_buf* is NULL, the *p_len* will be set to block size allocated. In case of *p_buf* is not NULL i.e. using user buffer, then on input *p_len* indicates the buffer size. The function will validate the size, if shorter than context data needed, the function will terminate with error. In case of success **tpsrvgetctxdata2()** will either return user pointer passed in *p_buf* if was not NULL or memory block allocated by function.

Returned pointer if was allocated by **tpsrvgetctxdata()** or **tpsrvgetctxdata2()** must be freed by caller by using **tpsrvfreectxdata()** to avoid memory leak. Currently **tpsrvfreectxdata()** is just a wrapper for system function **free()**, but this can change in future.

Sample app, original request goes to *TESTSVFN* but it is servered by later by new thread on function *_TH_TESTSVFN*.

```
#include <unistd.h>     /* Symbolic Constants */
#include <sys/types.h>  /* Primitive System Data Types */
#include <errno.h>      /* Errors */
#include <stdio.h>      /* Input/Output */
#include <stdlib.h>     /* General Utilities */
#include <pthread.h>    /* POSIX Threads */


#include <ndebug.h>
#include <atmi.h>
#include <ndrstandard.h>
#include <ubf.h>
#include <test.fd.h>
#include <string.h>

struct thread_server
{
    char *context_data; /* malloced by enduro/x */
    char *buffer; /* buffer data, managed by enduro/x */
};
/* note we must malloc this struct too. */
typedef struct thread_server thread_server_t;

/* threaded function... */
void _TH_TESTSVFN (void *ptr)
{
    int ret=EXSUCCEED;
    double d;
```

```
    int i;
    thread_server_t *thread_data = (thread_server_t *)ptr;
    UBFH *p_ub = (UBFH *)thread_data->buffer;

    if (EXSUCCEED!=tpinit(NULL))
    {
        NDRX_LOG(log_error, "Failed to init worker client");
        exit(1);
    }

    /* restore context. */
    if (EXSUCCEED!=tpsrvsetctxdata(thread_data->context_data, SYS_SRV_THREAD))
    {
        NDRX_LOG(log_error, "Failed to set context");
        exit(1);
    }

    /* free up the transport data.*/
    tpsrvfreectxdata(thread_data->context_data);
    free(thread_data);

    NDRX_LOG(log_debug, "TESTSVFN got call");

    /* Just print the buffer
    Bprint(p_ub);*/

    tpreturn(  ret==SUCCEED?TPSUCCESS:TPFAIL,
                0L,
                (char *)p_ub,
                0L,
                0L);

   /* this is not the end of the life - do tpterm for this thread...! */
    tpterm();
    NDRX_LOG(log_debug, "Thread is done - terminating...!");
}


/* main server thread...
 * NOTE: p_svc - this is local variable of enduro's main thread (on stack).
 * but p_svc->data - is auto buffer, will be freed when main thread returns.
 *                      Thus we need a copy of buffer for thread.
 */
void TESTSVFN (TPSVCINFO *p_svc)
{
    int ret=SUCCEED;
    UBFH *p_ub = (UBFH *)p_svc->data; /* this is auto-buffer */
    pthread_t thread;
    pthread_attr_t attr;
    long size;
    char btype[16];
    char stype[16];

    pthread_attr_init(&attr);
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);

    if (0==(size = tptypes (p_svc->data, btype, stype)))
    {
        NDRX_LOG(log_error, "Zero buffer received!");
        exit(1);
    }
```

```
    thread_server_t *thread_data = malloc(sizeof(thread_server_t));

    /* not using sub-type - on tpreturn/forward for thread it will be auto-free
*/
    thread_data->buffer =  tpalloc(btype, NULL, size);


    if (NULL==thread_data->buffer)
    {
        NDRX_LOG(log_error, "tpalloc failed of type %s size %ld", btype, size);
        exit(1);
    }

    /* copy off the request data */
    memcpy(thread_data->buffer, p_svc->data, size);

    thread_data->context_data = tpsrvgetctxdata();

    if (EXSUCCEED!=pthread_create (&thread, &attr, (void *) &_TH_TESTSVFN,
thread_data))
    {
        ret=FAIL;
        goto out;
    }

out:
    if (SUCCEED==ret)
    {
        /* serve next.. */
        tpcontinue();
    }
    else
    {
        /* return error back */
        tpreturn(  TPFAIL,
                0L,
                (char *)p_ub,
                0L,
                0L);
    }
}
/*
 * Do initialization
 */
int tpsvrinit(int argc, char **argv)
{
    NDRX_LOG(log_debug, "tpsvrinit called");

    if (SUCCEED!=tpadvertise("TESTSV", TESTSVFN))
    {
        NDRX_LOG(log_error, "Failed to initialize TESTSV (first)!");
    }
}

/**
 * Do de-initialization
 */
void tpsvrdone(void)
{
    NDRX_LOG(log_debug, "tpsvrdone called");
}
```

This function is available only for XATMI servers.

Chapter 3

RETURN VALUE

On success, **tpsrvgetctxdata()** return pointer to context data block; on error, NULL 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 the *p_len* indicates buffer shorter than needed for context data.

TPEPROTO Global transaction was started and it was marked for abort-only, there was any open call descriptors with-in global transaction,

TPERMERR Resource Manager failed (failed to suspend global transaction). The **tpstrerror()** will provide more info from last call.

TPESYSTEM System failure occurred during serving. See logs i.e. user log, or debugs for more info. This could also be a problem with dynamical driver loading.

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

Chapter 5

EXAMPLE

See **atmitest/test017_srvthread/atmisv17.c** for sample code.

Chapter 6

BUGS

Report bugs to support@mavimax.com

Chapter 7

SEE ALSO

tpsetctxdata(3) tpsrvfreectxdata(3) tpcontinue(3) tpinit(3)

Chapter 8

COPYING