

**TX\_COMMIT(3)**

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

# Contents

<a href="#">1 SYNOPSIS</a>	<a href="#">1</a>
<a href="#">2 DESCRIPTION</a>	<a href="#">2</a>
<a href="#">3 RETURN VALUE</a>	<a href="#">3</a>
<a href="#">4 ERRORS</a>	<a href="#">4</a>
<a href="#">5 EXAMPLE</a>	<a href="#">5</a>
<a href="#">6 BUGS</a>	<a href="#">6</a>
<a href="#">7 SEE ALSO</a>	<a href="#">7</a>
<a href="#">8 COPYING</a>	<a href="#">8</a>

## Chapter 1

# SYNOPSIS

```
#include <tx.h>
```

```
int tx_commit(void);
```

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

Function commits global transaction associated with current thread previously started by **tx\_begin(3)**. If transaction control was changed by **tx\_set\_transaction\_control(3)** flag **TX\_CHAINED**, the new transaction is started immediately after the commit.

If commit return was altered by **tx\_set\_commit\_return(3)** and setting flag was set to **TX\_COMMIT\_DECISION\_LOGGED**, the function will return after transaction is prepared by resource managers, and final commit (second phase of transaction) is left for transaction manager (**tmsrv(8)**) to complete by background thread, executed periodically (configured by **-s** scan time CLI parameter to **tmsrv**).

Function use following attributes set by:

- **tx\_set\_transaction\_control(3)** - if applied with **TX\_CHAINED**, new transaction is started. The default is **TX\_UNCHAINED** - meaning after commit, control thread is no more in global transaction.
- **tx\_set\_commit\_return(3)** - if applied with **TX\_COMMIT\_DECISION\_LOGGED** function returns when first phase of two phase commit is completed (and second phase is done by background thread). The default is **TX\_COMMIT\_COMPLETED** meaning that function returns when commit is completed.

TX API is base on TP API. This function is based on **tpcommit(3)** and it is possible to mix these two API kinds.

## Chapter 3

# RETURN VALUE

On success, **tx\_commit()** return **TX\_OK**; on error, error code is returned

## Chapter 4

# ERRORS

**TX\_NO\_BEGIN** The transaction committed successfully; however, a new transaction could not be started and the caller is no longer in transaction mode. This return value occurs only when the `transaction_control` characteristic is **TX\_CHAINED**.

**TX\_ROLLBACK** The transaction could not commit and has been rolled back. In addition, if the `transaction_control(3)` characteristic is **TX\_CHAINED**, a new transaction is started.

**TX\_ROLLBACK\_NO\_BEGIN** The transaction could not commit and has been rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control(3)` characteristic is **TX\_CHAINED**.

**TX\_MIXED** The transaction was partially committed and partially rolled back. In addition, if the `transaction_control(3)` characteristic is **TX\_CHAINED**, a new transaction is started.

**TX\_MIXED\_NO\_BEGIN** The transaction was partially committed and partially rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control(3)` characteristic is **TX\_CHAINED**.

**TX\_HAZARD** Due to a failure, the transaction may have been partially committed and partially rolled back. In addition, if the `transaction_control(3)` characteristic is **TX\_CHAINED**, a new transaction is started.

**TX\_HAZARD\_NO\_BEGIN** Due to a failure, the transaction may have been partially committed and partially rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control(3)` characteristic is **TX\_CHAINED**.

**TX\_PROTOCOL\_ERROR** The function was called in an improper context (for example, the caller is not in transaction mode). The caller's state with respect to the transaction is not changed.

**TX\_FAIL** Either the transaction manager or one or more of the resource managers encountered a fatal error. The nature of the error is such that the transaction manager and/or one or more of the resource managers can no longer perform work on behalf of the application. The exact nature of the error can be checked in process and/or transaction manager log files.

## Chapter 5

# EXAMPLE

See `atmitest/test021_xafull/atmict21tx.c` for sample code.

## Chapter 6

# BUGS

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

## Chapter 7

## SEE ALSO

`tx_begin(3)` `tpcommit(3)` `tx_rollback(3)` `tx_info(3)` `tx_close(3)` `tx_set_transaction_timeout(3)` `tx_set_transaction_control(3)`  
`tx_set_commit_return(3)`

## **Chapter 8**

# **COPYING**

© Mavimax, Ltd