TMSRV(8)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SYNOPSIS

tmsrv [*OPTIONS*]

Chapter 2

DESCRIPTION

This is special ATMI server which is used for distributed transaction coordination. For new transactions started with *tpbegin()*, *tmsrv* generates new XID and passed it back to transaction initiator. At the same time transaction is remembered by *tmsrv* as active transaction and time-out counter is checked by background thread.

In Enduro/X XA Resource Managers are numeric identifiers, which are allowed to be in range of 1..32. Enduro/X's Resource Manager ID (RMID) is same identifier as Group Number or grpno known in other ATMI systems.

If during distributed transaction processing new resource manager is associated with transaction, then process notifies initial transaction manager that new association must be made. In active phase this too is stored in process memory.

When transaction comes to commit() phase, then it is logged to disk. Every transaction is logged to separate file. File name contains resource manager ID and transaction XID. During the prepare phase status of every Resource Manager is logged. The same is with commit phase. Once all resources are completed, transaction file is removed. If *tmsrv* crashed for in-progress transaction, then transaction files are read from disk, and appropriate actions according to state machine are performed (aborted or rolled back).

If running transaction did time-out, then background thread will abort it automatically, and for caller process commit() will fail with abort-only message.

In cluster environment, other transaction manager of corresponding resource managers are involved for prepare/abort/commit actions. Also as every service doing as part of single transaction, must have a cluster link to initial transaction manager (for registration purposes). Thus means that all involved nodes, must have direct cluster visibility, otherwise transactions will fail.

Transaction managers can be load balanced with *ndrxconfig.xml* with min/max attributes. In load balance manner at tpbegin() corresponding free transaction manager will be selected. Later at transaction process this manager is responsible for full cycle of the transaction. Other resource managers for this transaction will help for prepare/commit/aborts of other RMs. These other TMs will be selected in load balanced mode.

Every instance of *tmsrv* will advertise following list of services

1. @TM-<Resource Manager ID>
2. @TM-<Cluster Node ID>-<Resource Manager ID>
3. @TM-<Cluster Node ID>-<Resource Manager ID>-<EnduroX Server ID>

For example for Resource Manager ID *1*, Cluster Node ID *6*, Enduro/X Server ID *10* services will look like:

1. @TM-1
2. @TM-6-1
3. @TM-6-1-10

Currently 1. format is used for starting the transaction, and accepting prepare/commit/abort calls from master TM of the transaction. Service Nr 3. is used by transaction initiator for subsequent calls of the *tpcommit()*/*tpabort()*. Also 3. is used by services involved in transaction to register new Resource Manager ID as part of the transaction.

For XA processing, resource manager drivers are loaded via dynamic loadable shared libs. Resource manager should expose *xa_switch* in shared lib. For every different resource manager, there is different Enduro/X server running. Enduro/X process gets associated with corresponding RMDI via *NDRX_XA_RES_ID* environment variable.

To configure different RMID's for different processes or *tmsrvs*. Use the Enduro/X build in facility of environment variable override. See the manpage of *ex_env(5)*.

Enduro/X support static and dynamic XA registration.

Chapter 3

OPTIONS

-t *DEFAULT_TIMEOUT*
DEFAULT_TIMEOUT is default transaction time-out in seconds.

-l *LOG_DIR*
LOG_DIR is full path to transaction log file directory. Process at startup scans the directory for transaction files. The format of the file name is following: *TRN-<Cluster Node ID>-<RMID>-<Server ID>*. To move transaction to different transaction manager. The log file should be renamed accordingly.

[-s *SCAN_TIME*]
Time in seconds for one cycle to perform transaction actions for background thread. I.e. the background thread does the sleep of this time on every loop. Default is set to *10*.

[-c *TIME_OUT_CHECK*]
This is periodic timer for doing active transactions time-out checks. Default is set to *1*

[-m *MAX_TRIES*]
Max tries to complete whole transaction by background thread. If the counter is reached, then no more attempts to complete the transaction are done. The counter is restarted at *tmsrv* reboot. Default is set to *100*.

[-r *XA_RETRIES*]
This is number of attempts on resource manager when it returns **XA_RETRY** during the commit. So lets say we have issued **tpcommit()** and some involved database is returning **XA_RETRY**. If *-r* is set above 2, then during the processing of *tpcommit()*, the xa commit to database will be retries one more time. If DB returns any other error than **XA_RETRY** or succeeds, the transaction is proceeded. If retry is returned again, then **TPEHAZARD** is returned to caller, transaction is moved to background thread, and will by processed by *-m* tries. But also here every *-m* try for **XA_RETRY** will be multiplied by *-r* attempts. Default value is seto to *3*.

[-p *THREAD_POOL_SIZE*]
This is the number of threads processing incoming requests. If all threads are busy, then job is internally queued. It is known that some databases slowly process some of the XA operations, for example *xa_rollback*. Thus multiple threads can handle this more efficiently. **Default threadpool size is set to 10**. For more load balancing it is recommended to start multiple *tmsrv* processes for same RMID. Note that **tmsrv** run with multiple threads, thus for Oracle DB flag *+Threads=true* **MUST** be set in **NDRX_XA_OPEN_STR**. Otherwise unexpected core dumps can be received from **tmsrv**.

[-P *PING_SECONDS*]
Number of seconds to perform database pings by either xa_start+TMJOIN flag or by xa_recover+TMSTARTRSCAN and TMENDRSCAN flags. The xa_recover is enabled by **-R** parameter. The **default** is xa_start. In case of xa_start from database it is expected error code XAER_NOTA (transaction not found) as the scan is performed for non existent XID, generated for each worker thread. For xa_recover it is expected that operation succeeds. If the operations goes out of the normal behavior, then reconnection procedure is set in **NDRX_XA_FLAGS** - tag **RECON** i.e. thread will perform xa_close() and xa_open() and retry operation. See the **ex_env(5)** manpage for the details. But for quick reference you may use value *RECON:*:3:100* which will perform 3x attempts on any error by sleeping 100 ms in between attempts. The

NDRX_XA_FLAGS must be set in CC config or environment and the attempts must be greater that 1. Other with the **tmsrv** will not boot with **-P** flag set.

[-R]

Enable xa_recover() call for PINGs instead of xa_start(). See **-P** flag description.

Chapter 4

XA RECOVER SETTINGS FOR ORACLE DB

The -R mode might not be enabled in database for user. I.e. user is not allowed to see open transactions. Thus must be enabled by following commands on DB user set in XA open string:

```
grant select on pending_trans$ to <database_user>;
grant select on dba_2pc_pending to <database_user>;
grant select on dba_pending_transactions to <database_user>;
grant execute on dbms_system to <database_user>;  (If using Oracle 10.2)
grant execute on dbms_xa to <database_user>; (If using Oracle 10.2)
```

Chapter 5

EXIT STATUS

0
Success

1
Failure

Chapter 6

BUGS

Report bugs to support@mavimax.com

Chapter 7

SEE ALSO

ex_env(5)

Chapter 8

AUTHOR

Enduro/X is created by Madars Vitolins.

Chapter 9

COPYING

© Mavimax, Ltd