LIBNDRXXAECPG(8)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SYNOPSIS

libndrxxaecpg.so libndrxxaecpg.dylib

```
NDRX_XA_RES_ID=1
NDRX_XA_OPEN_STR={"url":"tcp:postgresql://${EX_PG_HOST}/${EX_PG_DB}"
        ,"user":"${EX_PG_USER}"
        ,"password":"${EX_PG_PASS}"
        ,"compat":"PGSQL|INFORMIX|INFORMIX_SE"
        }"
NDRX_XA_CLOSE_STR=$NDRX_XA_OPEN_STR
NDRX_XA_DRIVERLIB=libndrxxapq.so
NDRX_XA_RMLIB=-
NDRX_XA_LAZY_INIT=1
```

Chapter 2

DESCRIPTION

This is XA driver specifically written Enduro/X needs. It provides an XA switch emulation on top of PostgreSQL prepared transactions. PostgreSQL by default does not support XA switch. Also when transaction start, there is no possibility to identify the work unit performed. The identification of work done by some process on particular connection can be done by preparing the transaction. Thus there is no such thing as "active" transaction in terms of XA specification. Also there is no possibility for other processes to join the existing work and see work done by other session. Thus for example if one server process in same transaction performs some insert and other process tries insert on table which has foreign key to first insert, it will fail, as FK will not be seen. Thus Enduro/X needs to work on branch-transactions mode with out join feature. The mode of PostgreSQL driver is the same as enabled by **NDRX_XA_FLAG** with value **NOJOIN**.

The emulation of XA protocol is done by following steps and assumptions:

1. When process joins to global transaction, new branch-transaction-id is acquired. The TMSRV logs the branch with status *u*-unknown.
2. When process completes the work unit (server does tpreturn(3) or tpforward(3)) or initiating process performs tpcommit(3), the xa_end() is called which in-turn runs PostgreSQL "PREPARE TRANSACTION <XID_STR>". "XID_STR" is based on PosgreSQL JDBC driver format. Thus JDBC version on PQ versions of tmsrv drivers can be mixed. When transaction is prepared OK, the TMSRV is reported with *p*-prepared status. If prepare fails, this means that transaction is aborted by PosgreSQL, and in this case *a*-aborted status is reported to TMSRV.
3. When TMSRV tries to commit, and branch is in *u*-unknown or *a*-aborted status, the global transaction is aborted and tpcommit caller receives **TPEABORT** error. If transaction is in status *p*, the prepare phase of global transaction is skipped/no operation and then commit is executed.
4. In case if work unit performs too long operation and tries to xa_end() after when TMSRV is already timed-out/rolled back the global transaction, the xa_end() status call to TMSRV fails with error **TPEMATCH**. In this case local process rolls back the prepared transaction.
5. When tpabort is executed, the xa_end() within tpabort() process performs abort with out executing prepare.

The driver can be used for standard libpq programming and for using ECPG precompiler. Internally the libndrxxaecpg driver manages connection by help of ECPG routines. But thanks to their API, the connection handler can be extracted to used by libPQ.

The connection details are encoded in JSON based string which contains the database URL, user name and password.

To get connection handler for PG sessions, used **tpgetconn(3)** function. Which is available for this driver.

Chapter 3

CONNECTION PARAMETERS

url
: This is standard PostgreSQL connection URL. Typically it contains database host, port and database name. The Enduro/X standard environment variable substitution is used here.

user
: This is PosgreSQL connection user name. Enduro/X standard environment substitution applies. Used where applies.

password
: Password used for establishing connection. Enduro/X standard environment substitution applies. Used where applies.

compat
: Compatibility mode. The default is **PGSQL**. Other options are **INFORMIX** and **INFORMIX_SE**.

Chapter 4

EXTERNAL SYMBOLS

ndrx_G_PG_conname
This is connection name currently associated with thread. It is thread is stored in thread local storage (TLS). Definition is **__thread char ndrx_G_PG_conname[65]**.

Chapter 5

CONFIGURATION EXAMPLE

When starting to use Enduro/X PQ XA Driver, ensure that PosgreSQL PQ and ECPG libraries are installed. Particularity **libecpg** and **libpq** must be installed.

The typical configuration is done as a standard Enduro/X XA resource configuration, which can be set directly in environment variables or in **[@global]** section in application configuration (e.g. app.ini). This gives example of app.ini configuration.

Sample configuration **app.ini** for CCTAG **DB1_PQ**:

```
[@global/DB1_PQ]
NDRX_XA_RES_ID=1
NDRX_XA_OPEN_STR={"url":"tcp:postgresql://localhost:5432/testdb",
        "user":"testuser", "password":"testuser1"}
NDRX_XA_CLOSE_STR=${NDRX_XA_OPEN_STR}
NDRX_XA_DRIVERLIB=libndrxxaecpg.so
NDRX_XA_RMLIB=-
NDRX_XA_LAZY_INIT=1
```

Sample configuration of transaction manager in **ndrxconfig.xml** for CCTAG **DB1_PQ**:

```
    <servers>
        ...
        <server name="tmsrv">
            <max>1</max>
            <srvid>1650</srvid>
            <cctag>DB1_PQ</cctag>
            <sysopt>-e /tmp/tmsrv-dom1.log -r -- -t1 -l/tmp</sysopt>
        </server>
        ...
    </servers>
```

Chapter 6

LIBPQ C EXAMPLE

This is example of programming database with libpq.

File: test_expq.c

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#include <atmi.h>
#include <libpq-fe.h>

#define FAIL -1
#define SUCCEED 0

int main(int argc, char** argv)
{
    PGconn * conn;
    PGresult *res = NULL;
    ExecStatusType estat;
    int ret = SUCCEED;

    /* open connection */
    if (EXSUCCEED!=tpopen())
    {
        fprintf(stderr, "Failed to open: %s\n", tpstrerror(tperrno));
        ret = FAIL;
        goto out;
    }

    /* get the connection which was open by tpopen() */
    conn = (PGconn *)tpgetconn();

    /* create some table... */

    res = PQexec(conn, "CREATE TABLE manextest(userid integer UNIQUE NOT NULL);");

    estat = PQresultStatus(res);

    if (PGRES_COMMAND_OK != estat)
    {
        char *state = PQresultErrorField(res, PG_DIAG_SQLSTATE);
        char *msg = PQresultErrorField(res, PG_DIAG_MESSAGE_PRIMARY);

        fprintf(stderr, "Failed to create table: state: [%s]: %s\n", state, msg);
```

```
        if (0==strcmp(state, "42P07"))
        {
            fprintf(stderr, "Table already exist - ignore error\n");
        }
        else
        {
            ret = FAIL;
            goto out;
        }
    }

    /* start transaction */
    if (EXSUCCEED!=tpbegin(60, 0))
    {
        fprintf(stderr, "Failed to begin: %s\n", tpstrerror(tperrno));
        ret = FAIL;
        goto out;
    }


    /* insert data */

    PQclear(res);

    res = PQexec(conn, "insert into manextest(userid) values ((select COALESCE(max(userid), ↩
         1)+1 from manextest));");

    estat =PQresultStatus(res);

    if (PGRES_COMMAND_OK != estat)
    {
        char *state = PQresultErrorField(res, PG_DIAG_SQLSTATE);
        char *msg = PQresultErrorField(res, PG_DIAG_MESSAGE_PRIMARY);

        fprintf(stderr, "Failed to create table: state: [%s]: %s\n", state, msg);
        ret = FAIL;
        goto out;
    }

    if (SUCCEED!=tpcommit(0))
    {
        fprintf(stderr, "TESTERROR: Commit OK, must fail!\n");
        ret = FAIL;
        goto out;
    }

out:
    if (SUCCEED!=ret)
    {
        tpabort(0);
    }
    tpclose();
    tpterm();

}
```

Build the program with:

```
$ cc test_expq.c -o expqtest -I/usr/include/postgresql -lpq -lecpg -latmi -lnstd -lubf -lrt
```

Run and test:

```
$ ./expqtest
Failed to create table: state: [42P07]: relation "manextest" already exists
Table already exist - ignore error

$ psql -U testuser -d testdb -h localhost

testdb=> select * from manextest;
 userid

      2
(1 row)
```

```
ECPG C EXAMPLE
```

The same code example above can be written in PosgreSQL embedded C code. Assuming that table is already created by previous example.

File: test_exec.pgc

```
#include <stdio.h>
#include <stdlib.h>
#include <ndebug.h>
#include <atmi.h>
#include <ecpglib.h>

#define FAIL -1
#define SUCCEED 0

int main(int argc, char** argv)
{
    int ret = SUCCEED;
    EXEC SQL BEGIN DECLARE SECTION;
    long id;
    EXEC SQL END DECLARE SECTION;


    /* open connection */
    if (EXSUCCEED!=tpopen())
    {
        fprintf(stderr, "Failed to open: %s\n", tpstrerror(tperrno));
        ret = FAIL;
        goto out;
    }

    /* start transaction */
    if (EXSUCCEED!=tpbegin(60, 0))
    {
        fprintf(stderr, "Failed to begin: %s\n", tpstrerror(tperrno));
        ret = FAIL;
        goto out;
    }

    EXEC SQL SELECT COALESCE(MAX(USERID), 1)+1 into :id from manextest;

    if ((sqlca.sqlcode < 0) || (sqlca.sqlcode == 100))
    {

        fprintf(stderr, "failed to get max: error code [%ld] message [%s] rows %ld, warning ↩
            %c\n",
                   sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, sqlca.sqlerrd[2], sqlca.sqlwarn ↩
                       [0]);
```

```
        ret = FAIL;
        goto out;
    }


    EXEC SQL INSERT INTO manextest(USERID) VALUES (:id);

    if ((sqlca.sqlcode < 0) || (sqlca.sqlcode == 100))
    {

        fprintf(stderr, "Failed to insert: error code [%ld] message [%s] rows %ld, warning  ↩
            %c\n",
                    sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, sqlca.sqlerrd[2], sqlca.sqlwarn ↩
                        [0]);
        ret = FAIL;
        goto out;
    }

    if (SUCCEED!=tpcommit(0))
    {
        fprintf(stderr, "TESTERROR: Commit OK, must fail!\n");
        ret = FAIL;
        goto out;
    }

out:

    if (SUCCEED!=ret)
    {
        tpabort(0);
    }

    tpclose();
    tpterm();
}
```

Build program with:

```
$ ecpg test_exec.pgc
$ cc test_exec.c -o exectest -I/usr/include/postgresql -lecpg -latmi -lnstd -lubf -lrt
```

Execute the test:

```
$ ./exectest
Failed to create table: state: [42P07]: relation "manextest" already exists
Table already exist - ignore error

$ psql -U testuser -d testdb -h localhost

testdb=> select * from manextest;
 userid

      2
      3
(1 row)
```

```
For more unit tests please see 'atmitest/test067_postgres' unit test folder for
ECPG, PQ source examples and configuration.

BUGS
```

Report bugs to support@mavimax.com

Chapter 7

SEE ALSO

libndrxxpq(8) ndrxconfig.xml(5) tmsrv(8) libndrxxawsmqs(8) libndrxxaoras(8) tpgetconn(3)

Chapter 8

COPYING