Building Enduro/X On MacOS Platform

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
1.0	2016-06	Initial draft	UV

Contents

Chapter 1

About manual

This manual describes how to build *Enduro/X* on fresh installed OS X. Document is based on operating system version 10.11 (El Capitan). Generally version is considered as development time only and experimental for production use for low queue and message density setups. This is due to fact, that MAC OS X does not provide Posix queue API and emulation is in use. During the porting Enduro/X to OSX it have been monitored that system can freeze. How ever the emulated queues on GNU/Linux works good. So there might be a bug in OSX Posix thread mutex routines. Also OSX does not support robust mutexes which means, that if some process get the lock for emulated queue and process is being killed, then lock is never released and if this queue is **ndrxd(8)** main q, then system is stalled. In future release this might be fixed with advancing **tprecover(8)** to do pings to **ndrxd(8)** and doing kill and reboot the *ndrxd* in learning mode.

Chapter 2

Overview

Enduro/X normally use Operating System provided Posix kernel queues, however OS X does not have this subsystem, however Enduro/X provides queue emulation via POSIX Threads functionality. This includes process shared semaphores and memory mapped files. Enduro/X CMake scripts will automatically do the configuration and enable this functionality.

In the result queues will we listed on file system. The recommendation is to use separate folder this for example "/tmp/mq".

This document covers only core functionality of the Enduro/X building. That's mean that building of:

1. documentation;
2. gpg-me;

is out of the scope.

Chapter 3

Operating System Configuration

For OS configuration settings see ex_adminman(guides)(Enduro/X Administration Manual, Setup System chapter).

Chapter 4

Installation process

This chapter describes how to install system dependencies needed from build and how to build Enduro/X it self.

4.1 Packages to be installed

1. Apple Xcode (https://developer.apple.com/xcode/) install from Appstore or other sources
2. Git source code version control system
3. CMake (https://cmake.org/install/)

4.2 Getting the source code

For test purposes we will parepare new user for which Enduro/X will built.

```
$ sudo -s
# dscl . -create /Users/user1
# dscl . -create /Users/user1 UserShell /bin/bash
# dscl . -create /Users/user1 RealName "Test User"
# dscl . -create /Users/user1 UniqueID "510"
# dscl . -create /Users/user1 PrimaryGroupID 20
# dscl . -create /Users/user1 NFSHomeDirectory /Users/user1
# dscl . -passwd /Users/user1 password
# su - user1
$ cd /Users/user1
$ git clone https://github.com/endurox-dev/endurox endurox
```

To work with CMake, you need to put the *cmake* binary into the path. You may do that by

```
$ cat << EOF >> ~/.bashrc
export PATH=$PATH:/Applications/CMake.app/Contents/bin
EOF

$ chmod +x .bashrc
```

4.3 Enduro/X basic Environment configuration for HOME directory

This code bellow creates *ndrx_home* executable file which loads basic environment, so that you can use sample configuration provided by Enduro/X in *sampleconfig* directory. This also assumes that you are going to install to *$HOME/endurox/dist* folder.

Note
If we install the libaries in dist/lib64, OS X will still try for to get the the shared libraries for origin build folders. Thus it is recommended to configure environment file bit more advanced, to enable OS X library loader to search original (build) lib folders firstly.

Note
For test cases like 021 (XA Testing) and 028 (TMQUEUE) tests, the scripts uses *DYLD_FALLBACK_LIBRARY_PATH* environment variable. Hoever testing scripts are using */bin/bash*, which means that by enabled SIP (System Integrity Protection), the variable is dropped. Thus pass all ATMI tests, you have to disable SIP.

```
cat << EOF > $HOME/ndrx_home
#!/bin/bash

echo "Loading ndrx_home ..."
# Where app domain lives
export NDRX_APPHOME=/Users/user1/endurox
# Where NDRX runtime lives
export NDRX_HOME=/Users/user1/endurox/dist/bin
# Debug config too
export NDRX_DEBUG_CONF=/Users/user1/endurox/sampleconfig/debug.conf
# NDRX config too.
export NDRX_CONFIG=/Users/user1/endurox/sampleconfig/ndrxconfig.xml

export PATH=$PATH:/Users/user1/endurox/dist/bin
export FLDTBLDIR=/Users/user1/endurox/ubftest/ubftab

# For testing firstly use original folder where dynamic libraries were built:
for f in libthpool libexuuid libcgreen libubf libnstd libnetproto libatmi libpsstdlib libps ↩
     libexnet libatmisrv libatmiclt tmqueue; do

        export DYLD_FALLBACK_LIBRARY_PATH=$DYLD_FALLBACK_LIBRARY_PATH:/Users/user1/endurox/ ↩
            $f
        echo $f
done

# Speedup a bit q open & close
export NDRX_MSGMAX=100
export NDRX_MSGSIZEMAX=40000

# Increase stack size
ulimit -s 40000

##############################################################################
# In case if building with Postgresql DB database testing support
# or building endurox-java with Oracle DB tests (03_xapostgres), then
# configure bellow setting (demo values provided):
# If so - uncomment bellow
##############################################################################
#export EX_PG_HOST=localhost
#export EX_PG_USER=exdbtest
#export EX_PG_PASS=exdbtest1
# currently uses default port
#export EX_PG_PORT=5432
#export EX_PG_DB=xe

EOF

$ chmod +x $HOME/ndrx_home
```

4.4 Configuring PostgreSQL

If Enduro/X PostgreSQL driver is needed to be build for MacOS, the PostgreSQL needs to be installed for build and test purposes. For installation purposes *brew* is used.

```
$ su - user1
$ brew update
$ brew install postgresql
```

Once PostgreSQL is installed, update the configuration and create the database. Also needs to ensure that **user1** has write permissions to *var* folder:

```
--- Add user to admin groups so that it has access to /usr/local/var
sudo dseditgroup -o edit -a user1 -t user admin
sudo dseditgroup -o edit -a user1 -t user wheel

-- ensure that user1 has write permissions:
$ sudo chmod g+w /usr/local/var/

-- Create the DB for postgres. If installing Postgresql from other user, then
-- folder needs to be removed.
$ initdb /usr/local/var/postgres

$ pg_ctl -D /usr/local/var/postgres -l logfile start

$ createuser exdbtest

$ createdb xe

$ psql -d template1

> alter user exdbtest with encrypted password 'exdbtest1';
> grant all privileges on database xe to exdbtest;
> \q
```

Configuration files needs to be updated for authentication and distributed transactions must be enabled too.

Edit **/usr/local/var/postgres/postgresql.conf**, set "max_prepared_transactions" to 1000.

```
max_prepared_transactions = 1000                # zero disables the feature
```

For access permissions and network configuration, update **/usr/local/var/postgres/pg_hba.conf**, so that it contains following:

```
local   all             all                                     peer
host    all             all             127.0.0.1/32            md5
host    all             all             ::1/128                 md5
```

Restart PostgreSQL:

```
$ pg_ctl -D /usr/local/var/postgres stop
$ pg_ctl -D /usr/local/var/postgres start
```

To add PostgreSQL to system startup, execute:

```
$ brew services start postgresql
```

NOTE: Auto startup adding works only in case if brew was installed from *user1*.

4.5 Building the code

This install the binary version to /Users/user1/endurox:

```
$ cd /Users/user1/endurox
$ cmake -DDEFINE_DISABLEGPGME=ON -DDEFINE_DISABLEDOC=ON -DCMAKE_INSTALL_PREFIX:PATH=`pwd`/ ↩
    dist .
$ make
$ make install
```

Chapter 5

Unit Testing

Enduro/X basically consists of two parts: . XATMI runtime; . UBF/FML buffer processing. Each of these two sub-systems have own units tests.

5.1 UBF/FML Unit testing

```
$ cd /Users/user1/endurox/ubftest
$ ./ubfunit1 2>/dev/null
Running "main" (76 tests)...
Completed "ubf_basic_tests": 198 passes, 0 failures, 0 exceptions.
Completed "ubf_Badd_tests": 225 passes, 0 failures, 0 exceptions.
Completed "ubf_genbuf_tests": 334 passes, 0 failures, 0 exceptions.
Completed "ubf_cfchg_tests": 2058 passes, 0 failures, 0 exceptions.
Completed "ubf_cfget_tests": 2232 passes, 0 failures, 0 exceptions.
Completed "ubf_fdel_tests": 2303 passes, 0 failures, 0 exceptions.
Completed "ubf_expr_tests": 3106 passes, 0 failures, 0 exceptions.
Completed "ubf_fnext_tests": 3184 passes, 0 failures, 0 exceptions.
Completed "ubf_fproj_tests": 3548 passes, 0 failures, 0 exceptions.
Completed "ubf_mem_tests": 4438 passes, 0 failures, 0 exceptions.
Completed "ubf_fupdate_tests": 4613 passes, 0 failures, 0 exceptions.
Completed "ubf_fconcat_tests": 4768 passes, 0 failures, 0 exceptions.
Completed "ubf_find_tests": 5020 passes, 0 failures, 0 exceptions.
Completed "ubf_get_tests": 5247 passes, 0 failures, 0 exceptions.
Completed "ubf_print_tests": 5655 passes, 0 failures, 0 exceptions.
Completed "ubf_macro_tests": 5666 passes, 0 failures, 0 exceptions.
Completed "ubf_readwrite_tests": 5764 passes, 0 failures, 0 exceptions.
Completed "ubf_mkfldhdr_tests": 5770 passes, 0 failures, 0 exceptions.
Completed "main": 5770 passes, 0 failures, 0 exceptions.
```

5.2 XATMI Unit testing

ATMI testing might take some time. Also ensure that you have few Gigabytes of free disk space, as logging requires some space. To run the ATMI tests do following:

```
$ cd /Users/user1/endurox/atmitest
$ nohup ./run.sh &
$ tail -f /Users/user1/endurox/atmitest
...
************ FINISHED TEST: [test028_tmq/run.sh] with 0 ************
Completed "atmi_test_all": 28 passes, 0 failure, 0 exceptions.
Completed "main": 28 passes, 0 failure, 0 exceptions.
```

Chapter 6

Conclusions

At finish you have a configured system which is read to process the transactions by Enduro/X runtime. It is possible to copy the binary version (*dist*) folder to other same architecture machine and run it there with out need of building.

Chapter 7

Additional documentation

7.1 Resources

[1] [BINARY_INSTALL] See Enduro/X binary_install manual.