NOTES FOR THE HPE NONSTOP PLATFORM

Requirement details

In addition to the requirements and instructions listed in INSTALL.md, the following are required as well:

- The TNS/X platform supports hardware randomization. Specify the `--with-rand-seed=rdcpu` option to the `./Configure` script. This is recommended but not required. `egd` is supported at 3.0 but cannot be used if FIPS is selected.
- The TNS/E platform does not support hardware randomization, so specify the `--with-rand-seed=egd` option to the `./Configure` script.

About c99 compiler

The c99 compiler is required for building OpenSSL from source. While c11 may work, it has not been broadly tested. c99 is the only compiler prerequisite needed to build OpenSSL 3.0 on this platform. You should also have the FLOSS package installed on your system. The ITUGLIB FLOSS package is the only FLOSS variant that has been broadly tested.

Threading Models

OpenSSL can be built using unthreaded, POSIX User Threads (PUT), or Standard POSIX Threads (SPT). Select the following build configuration for each on the TNS/X (L-Series) platform:

- `nonstop-nsx` or default will select an unthreaded build.
- `nonstop-nsx_put` selects the PUT build.
- `nonstop-nsx_64_put` selects the 64 bit file length PUT build.
- `nonstop-nsx_spt_floss` selects the SPT build with FLOSS. FLOSS is required for SPT builds because of a known hang when using SPT on its own.

TNS/E Considerations

The TNS/E platform is build using the same set of builds specifying `nse` instead of `nsx` in the set above.

You cannot build for TNS/E for FIPS, so you must specify the `no-fips` option to `./Configure`.

About Prefix and OpenSSLDir

Because there are many potential builds that must co-exist on any given NonStop node, managing the location of your build distribution is crucial. Keep each destination separate and distinct. Mixing any mode described in this document

can cause application instability. The recommended approach is to specify the OpenSSL version and threading model in your configuration options, and keeping your memory and float options consistent, for example:

- For 1.1 `--prefix=/usr/local-ssl1.1 --openssldir=/usr/local-ssl1.1/ssl`
- For 1.1 PUT `--prefix=/usr/local-ssl1.1_put --openssldir=/usr/local-ssl1.1_put/ssl`

As of 3.0, the NonStop configurations use the multilib attribute to distinguish between different models:

- For 3.0 `--prefix=/usr/local-ssl3.0 --openssldir=/usr/local-ssl3.0/ssl`

The PUT model is placed in `${prefix}/lib-put` for 32-bit models and `${prefix}/lib64-put` for 64-bit models.

Use the `_RLD_LIB_PATH` environment variable in OSS to select the appropriate directory containing `libcrypto.so` and `libssl.so`. In GUARDIAN, use the `=_RLD_LIB_PATH` search define to locate the GUARDIAN subvolume where OpenSSL is installed.

Float Considerations

OpenSSL is built using IEEE Float mode by default. If you need a different IEEE mode, create a new configuration specifying `tfloat-x86-64` (for Tandem Float) or `nfloat-x86-64` (for Neutral Float).

Memory Models

The current OpenSSL default memory model uses the default platform address model. If you need a different address model, you must specify the appropriate c99 options for compile (`CFLAGS`) and linkers (`LDFLAGS`).

Cross Compiling on Windows

To configure and compile OpenSSL, you will need to set up a Cygwin environment. The Cygwin tools should include bash, make, and any other normal tools required for building programs.

Your `PATH` must include the bin directory for the c99 cross-compiler, as in:

```
export PATH=/cygdrive/c/Program\ Files\ \(x86\)/HPE\ NonStop/L16.05/usr/bin:$PATH
```

This should be set before Configure is run. For the c99 cross-compiler to work correctly, you also need the `COMP_ROOT` set, as in:

```
export COMP_ROOT="C:\Program Files (x86)\HPE NonStop\L16.05"
```

`COMP_ROOT` needs to be in Windows form.

`Configure` must specify the `no-makedepend` option otherwise errors will result when running the build because the c99 cross-compiler does not support the `gcc`

`-MT` option. An example of a `Configure` command to be run from the OpenSSL directory is:

```
./Configure nonstop-nsx_64 no-makedepend --with-rand-seed=rdcpu
```

Do not forget to include any OpenSSL cross-compiling prefix and certificate options when creating your libraries.

The OpenSSL test suite will not run on your workstation. In order to verify the build, you will need to perform the build and test steps in OSS in your NonStop server. You can also build under gcc and run the test suite for Windows but that is not equivalent.

Note: In the event that you are attempting a FIPS-compliant cross-compile, be aware that signatures may not match between builds done under OSS and under cross-compiles as the compilers do not necessarily generate identical objects. Anything and everything to do with FIPS is outside the scope of this document. Refer to the FIPS security policy for more information.

The following build configurations have been successfully attempted at one point or another. If you are successful in your cross-compile efforts, please update this list:

- nonstop-nsx_64
- nonstop-nsx_64_put

Note: Cross-compile builds for TNS/E have not been attempted, but should follow the same considerations as for TNS/X above. SPT builds generally require FLOSS, which is not available for workstation builds. As a result, SPT builds of OpenSSL cannot be cross-compiled.

Also see the NSDEE discussion below for more historical information.

Cross Compiling with NSDEE

Note: None of these builds have been tested by the platform maintainer and are supplied for historical value. Please submit a Pull Request to OpenSSL should these need to be adjusted.

If you are attempting to build OpenSSL with NSDEE, you will need to specify the following variables. The following set of compiler defines are required:

```
# COMP_ROOT must be a full path for the build system (e.g. windows)
COMP_ROOT=$(cygpath -w /path/to/comp_root)
# CC must be executable by your shell
CC=/path/to/c99
```

Optional Build Variables

```
DBGFLAG="--debug"
CIPHENABLES="enable-ssl3 enable-ssl3-method enable-weak-ssl-ciphers enable-rc4"
```

Internal Known TNS/X to TNS/E Cross Compile Variables

The following definition is required if you are building on TNS/X for TNS/E and have access to a TNS/E machine on your EXPAND network - with an example node named `\CS3`:

```
SYSTEMLIBS="-L/E/cs3/usr/local/lib"
```

Version Procedure (VPROC) Considerations

If you require a VPROC entry for platform version identification, use the following variables:

For Itanium

```
OPENSSL_VPROC_PREFIX=T0085H06
```

For x86

```
OPENSSL_VPROC_PREFIX=T0085L01
```

Common Definition

```
export OPENSSL_VPROC=${OPENSSL_VPROC_PREFIX}_$(
    . VERSION.dat
    if [ -n "$PRE_RELEASE_TAG" ]; then
        PRE_RELEASE_TAG="-$PRE_RELEASE_TAG"
    fi
    echo "$MAJOR.$MINOR.$PATCH$PRE_RELEASE_TAG$BUILD_METADATA" |\
        sed -e 's/[-.+]/_/g'
    )
```

Example Configure Targets

For OSS targets, the main DLL names will be `libssl.so` and `libcrypto.so`. For GUARDIAN targets, DLL names will be `ssl` and `crypto`. The following assumes that your PWD is set according to your installation standards.

```
./Configure nonstop-nsx           --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=rdcpu ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nsx_g         --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=rdcpu ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nsx_put       --prefix=${PWD} \
    --openssldir=${PWD}/ssl threads "-D_REENTRANT" \
    --with-rand-seed=rdcpu ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nsx_spt_floss --prefix=${PWD} \
```

```
    --openssldir=${PWD}/ssl threads "-D_REENTRANT" \
    --with-rand-seed=rdcpu ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nsx_64        --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=rdcpu ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nsx_64_put    --prefix=${PWD} \
    --openssldir=${PWD}/ssl threads "-D_REENTRANT" \
    --with-rand-seed=rdcpu ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nsx_g_tandem  --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=rdcpu ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}

./Configure nonstop-nse           --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=egd ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nse_g         --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=egd ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nse_put       --prefix=${PWD} \
    --openssldir=${PWD}/ssl threads "-D_REENTRANT" \
    --with-rand-seed=egd ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nse_spt_floss --prefix=${PWD} \
    --openssldir=${PWD}/ssl threads "-D_REENTRANT" \
    --with-rand-seed=egd ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nse_64        --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=egd ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nse_64_put    --prefix=${PWD} \
    --openssldir=${PWD}/ssl threads "-D_REENTRANT"
    --with-rand-seed=egd ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
./Configure nonstop-nse_g_tandem  --prefix=${PWD} \
    --openssldir=${PWD}/ssl no-threads \
    --with-rand-seed=egd ${CIPHENABLES} ${DBGFLAG} ${SYSTEMLIBS}
```