QNX Neutrino RTOS

This guide explains how to build applications for QNX Neutrino RTOS 6.6 based on the Kaa C endpoint SDK (further, the C SDK).

NOTE: The further instructions are expected to be executed **on the host machine**.

Host OS: Ubuntu 14.04 LTS 64-bit.

Target OS: QNX Neutrino RTOS 6.6.

Installing QNX® Software Development Platform

To install QNX Software Development Platform (SDP), proceed as follows:

1. Register as a developer and download the following components:
    - QNX Software Development Platform 6.6
    - QNX® Software Development Platform 6.6 Applypatch Patch [Patch ID 4024]
    - QNX® Software Development Platform 6.6 Header Files Patch [Patch ID 3851]
2. Install SDP. **Note:** It is recommended to install SDP in the default directory (for Linux platforms, it is */opt/qnx660*).
3. Install Apply Patch and Header Files Patch.

Configuring the environment

1. Set the path to the root directory of SDP.

    ```
    export QNX_SDK_HOME="<path_to_qnx_sdk_home>"
    ```

    Default: */opt/qnx660*

2. Set the target architecture.

    ```
    export QNX_TARGET_ARCH=<architecture>
    ```

    Supported architectures:

    - gcc_ntoarmv7le_cpp-ne
    - gcc_ntox86_cpp-ne
    - gcc_ntox86_gpp
    - gcc_ntoarmv7le
    - gcc_ntox86
    - gcc_ntoarmv7le_cpp
    - gcc_ntoarmv7le_gpp
    - gcc_ntox86_cpp

    Default: *gcc_ntox86*

3. Set paths to host and target SDP files.

    ```
    export QNX_HOST="$QNX_SDK_HOME/host/linux/x86"
    export QNX_TARGET="$QNX_SDK_HOME/target/qnx6"
    export PATH="$PATH:$QNX_HOST/usr/bin"
    ```

Installing third-party components for C SDK

The following third-party components must be installed before building the C SDK.

- **Mandatory:** OpenSSL.

To install these libraries, proceed as follows:

1. Set the path to the directory where third-party dependencies will be put after build.

```
export THIRDPARTY_ROOT=<path>
export THIRDPARTY_PREFIX=$THIRDPARTY_ROOT/usr
mkdir -p $THIRDPARTY_PREFIX
```

2. Install OpenSSL.

```
wget ftp://ftp.openssl.org/source/openssl-1.0.2d.tar.gz
tar -zxf openssl-1.0.2d.tar.gz
cd openssl-1.0.2d/
CC="qcc -lsocket" && ./Configure -fPIC --prefix=$THIRDPARTY_PREFIX os/compiler:qcc
make install
unset TARGETMACH && unset CROSS  && unset CC && unset LD && unset AS && unset AR
```

Creating applications based on C SDK

Building C SDK

Before creating applications based on the C SDK, you should obtain the C SDK and build it into a static library. To do so, generate the C SDK in Admin UI, then extract the archive, `cd` to it and execute the following steps:

```
mkdir -p build
cd build
cmake -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchains/qnx.cmake -
DOPENSSL_ROOT_DIR=$THIRDPARTY_PREFIX -DOPENSSL_LIBRARIES=$THIRDPARTY_PREFIX/lib
make
```

For more details on building the C SDK for Linux, please refer to this page.

Example

To quickly start with the Kaa IoT platform, you can download one of Kaa demo applications from the Kaa Sandbox and run it on on a target machine with QNX Neutrino RTOS.
For this example, you need to download the notification demo from the Kaa Sandbox to your host machine. Build it and export an executable file to the target machine. Then run the demo, as follows

1. Build the demo.

Host

```
tar -zxf notification_demo.tar.gz
cd CNotificationDemo/
sed -i 's/kaac crypto/kaac_s libcrypto.a libsocket.a/g' CMakeLists.txt
sed -i 's/cmake -DKAA_DEBUG_ENABLED=1/cmake -G "Unix Makefiles" -DKAA_DEBUG_ENABLED=1 -
DCMAKE_TOOLCHAIN_FILE=..\/toolchains\/qnx.cmake -DOPENSSL_ROOT_DIR=\$THIRDPARTY_PREFIX -
DOPENSSL_LIBRARIES=$THIRDPARTY_PREFIX\/lib/g' build.sh
sed -i 's/cmake -DAPP_NAME=\$APP_NAME/cmake -G "Unix Makefiles" -DAPP_NAME=$APP_NAME -
DCMAKE_TOOLCHAIN_FILE=..\/libs\/kaa\/toolchains\/qnx.cmake -DOPENSSL_ROOT_DIR=$THIRDPARTY_PREFIX -
DOPENSSL_LIBRARIES=$THIRDPARTY_PREFIX\/lib/g' build.sh
./build.sh clean build
```

2. Export the executable file to the target machine.

Host

```
scp build/demo_client <user>@<ip_of_target_machine>:demo_client
```

3. Run the demo.

Target
./demo_client

If the build is successful you will see the following output on the target terminal:

```
Notification demo started
--= Press Enter to exit =--
Topic list is empty
Topic list was updated
Topic: id '603', name 'Sample mandatory topic', type 'MANDATORY_SUBSCRIPTION'
Topic: id '604', name 'Sample optional topic', type 'OPTIONAL_SUBSCRIPTION'
Subscribing to optional topic '604'
Notification for topic id '603' received
Notification body: 'This is sample notification for mandatory topic. Client automatically receive
notifications for manadatory topics. More details here: https://docs.kaaproject.org/display/KAA
/Using+notifications'
Notification for topic id '604' received
Notification body: 'This is sample notification for optional topic. Client should subscribe to optional
topics in order to receive notifications. This application subscribes to optional topics automatically.'
Notification demo stopped
```
