ROS2 Documentation for QNX

Release 0.0.1

Ahmed Sobhy

Dec 16, 2022

CONTENTS

1 Distribution: Foxy **1**
1.1 Guide 1

CHAPTER

ONE

DISTRIBUTION: FOXY

1.1 Guide

1.1.1 License

Creative Commons Legal Code

CC0 1.0 Universal

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS DOCUMENT DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED HEREUNDER, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED HEREUNDER.

Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer exclusive Copyright and Related Rights (defined below) upon the creator and subsequent owner(s) (each and all, an "owner") of an original work of authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for the purpose of contributing to a commons of creative, cultural and scientific works ("Commons") that the public can reliably and without fear of later claims of infringement build upon, modify, incorporate in other works, reuse and redistribute as freely as possible in any form whatsoever and for any purposes, including without limitation commercial purposes. These owners may contribute to the Commons to promote the ideal of a free culture and the further production of creative, cultural and scientific works, or to gain reputation or greater distribution for their Work in part through the use and efforts of others.

For these and/or other purposes and motivations, and without any expectation of additional consideration or compensation, the person associating CC0 with a Work (the "Affirmer"), to the extent that he or she is an owner of Copyright and Related Rights in the Work, voluntarily elects to apply CC0 to the Work and publicly distribute the Work under its terms, with knowledge of his or her Copyright and Related Rights in the Work and the meaning and intended legal effect of CC0 on those rights.

1. Copyright and Related Rights. A Work made available under CC0 may be protected by copyright and related or neighboring rights ("Copyright and Related Rights"). Copyright and Related Rights include, but are not limited to, the following:

a. the right to reproduce, adapt, distribute, perform, display, communicate, and translate a Work;

b. moral rights retained by the original author(s) and/or performer(s);

c. publicity and privacy rights pertaining to a person's image or likeness depicted in a Work;

d. rights protecting against unfair competition in regards to a Work, subject to the limitations in paragraph 4(a), below;

e. rights protecting the extraction, dissemination, use and reuse of data in a Work;

f. database rights (such as those arising under Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, and under any national implementation thereof, including any amended or successor version of such directive); and

g. other similar, equivalent or corresponding rights throughout the world based on applicable law or treaty, and any national implementations thereof.

2. Waiver. To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

3. Public License Fallback. Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "License"). The License shall be deemed effective as of the date CC0 was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's express Statement of Purpose.

4. Limitations and Disclaimers.

a. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.

b. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.

c. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.

d. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CC0 or use of the Work.

1.1.2 Building ROS2 Foxy for QNX

Note: Use the menu at the bottom left corner of the page to select the distribution.

Table of Contents

- *Overview of the build process*
- *System requirements*
- *System setup*
  - *Set locale*
  - *Add the ROS 2 apt repository*
  - *Install development tools and ROS tools*
- *Building steps*

The following instructions go over the steps for building ROS2 foxy for QNX including FastRTPS and CycloneDDS RMW implementations.

Overview of the build process

Starting with a QNX SDP7.1 installation along with the required cross compiled dependencies, the build process will cross compile ROS 2's source code against SDP7.1 and the cross compiled dependencies. Binaries will be generated for the two architectures below:

- aarch64le
- x86_64

The generated files can then be transferred to the required target and used. The following document will go over the steps needed to cross compile the dependencies and ROS 2.

System requirements

HOST:

- Ubuntu 20.04
- QNX SDP7.1

For instructions to install SDP7.1 please follow the link: http://www.qnx.com/developers/docs/7.1/index.html#com.qnx.doc.qnxsdp.quickstart/topic/about.html

TARGET:

- A QNX supported architecture running QNX SDP7.1

System setup

Set locale

Make sure to set a locale that supports UTF-8.

The following is an example for setting locale. However, it should be fine if you're using a different UTF-8 supported locale.

```
sudo apt-get update && sudo apt-get install -y locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
```

Add the ROS 2 apt repository

You will need to add the ROS 2 apt repository to your system. Make sure the Ubuntu Universe repository is enabled first by checking the output of this command.

```
$ apt-cache policy | grep universe
 500 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 Packages
     release v=20.04,o=Ubuntu,a=focal,n=focal,l=Ubuntu,c=universe,b=amd64
```

If you don't see output like the above, then enable the Universe repository with these instructions.

```
sudo apt install software-properties-common
sudo add-apt-repository universe
```

Now add the ROS 2 apt repository to your system. First authorize our GPG key with apt.

```
sudo apt update && sudo apt install curl gnupg lsb-release
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key  -o /usr/
↪share/keyrings/ros-archive-keyring.gpg
```

Then add the repository to your sources list.

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
↪keyring.gpg] http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" | sudo tee /
↪etc/apt/sources.list.d/ros2.list > /dev/null
```

Install development tools and ROS tools

```
sudo apt update && sudo apt install -y \
  build-essential \
  git \
  python3-colcon-common-extensions \
  python3-flake8 \
  python3-pip \
  python3-pytest-cov \
  python3-rosdep \
  python3-setuptools \
```

(continues on next page)

(continued from previous page)

```
  python3-vcstool \
  wget

# install some pip packages needed for testing
python3 -m pip install -U \
  argcomplete \
  flake8-blind-except \
  flake8-builtins \
  flake8-class-newline \
  flake8-comprehensions \
  flake8-deprecated \
  flake8-docstrings \
  flake8-import-order \
  flake8-quotes \
  pytest-repeat \
  pytest-rerunfailures \
  pytest

 # Install additional tools needed for building the dependencies for QNX
 sudo apt update && sudo apt install -y \
   bc \
   subversion \
   autoconf \
   libtool-bin \
   libssl-dev \
   zlib1g-dev \
   rsync \
   rename

 python3 -m pip install -U \
   Cython \
   numpy \
   lark-parser

 # Optional: If CycloneDDS is needed then it has to be build for host first to use␣
↪dssconf tool required when cross compiling
 sudo apt install -y bison
 cd ~/
 git clone -b iceoryx https://github.com/eclipse-cyclonedds/cyclonedds.git
 cd cyclonedds
 mkdir build
 cd build
 cmake ..
 cmake --build . --target ddsconf idlc
 export DDSCONF_EXE=$(find ~/cyclonedds -type f -name ddsconf)
 export IDLC_EXE=$(find ~/cyclonedds -type f -name idlc)
```

```
cd /opt && sudo wget https://cmake.org/files/v3.18/cmake-3.18.0-Linux-x86_64.sh
sudo mkdir /opt/cmake-3.18.0-Linux-x86_64
yes | sudo sh cmake-3.18.0-Linux-x86_64.sh --prefix=/opt/cmake-3.18.0-Linux-x86_64 --
↪skip-license
sudo ln -s /opt/cmake-3.18.0-Linux-x86_64/bin/cmake /usr/local/bin/cmake
```

Building steps

1- From withing the directory ~/ros2_foxy, clone additional files necessary for building ROS 2 and the dependencies then merge them with your ROS 2 directory.

```
mkdir ~/ros2_foxy
cd ~/ros2_foxy
git clone -b foxy https://gitlab.com/qnx/frameworks/ros2/ros2_qnx.git /tmp/ros2
rsync -haz /tmp/ros2/* .
rm -rf /tmp/ros2
```

2- Import ROS 2 code and apply patches.

```
mkdir -p ~/ros2_foxy/src
cd ~/ros2_foxy
vcs import src < ros2.repos
./patch.sh
```

3- Import the required QNX build files for each dependency by importing QNX dependencies repositories.

```
mkdir -p src/qnx_deps
vcs import src/qnx_deps < qnx_deps.repos
```

4- Run a script to automatically embed <build_depend> in the packages that depends on qnx_deps.

```
./patch-pkgxml.py --path=src
```

5- Before building ROS 2, some packages will need to be ignored first. Which are as following.

```
./colcon-ignore.sh
```

6- Export CPU variable according to your target architecture:

Please note: If no CPU is set all architectures are going to be built.

options for CPU: aarch64, x86_64

```
export CPU=aarch64
```

7- Source qnxsdp-env.sh script.

```
. ~/qnx710/qnxsdp-env.sh
```

Optional: Add the sourcing command to the end of ~/.bashrc if you would like the environment to be set every time for you.

8- Build ROS 2.

```
./build-ros2.sh
```

1.1.3 Target Setup and Testing

Setup the Target:

1- ssh to your target or run the following commands on your target directly.

2- make sure libffi is included with your image otherwise copy it over from your sdp

```
scp ~/qnx710/target/qnx7/x86_64/usr/lib/libffi.so.6 root@<target_ip>:/usr/lib/
ln -s /usr/lib/libffi.so.6 /usr/lib/libffi.so
```

3- Download CA certificates bundle on your PC then copy it over to your target:

On your target:

```
mkdir -p /etc/curl
```

On your PC:

```
curl --time-cond cacert.pem https://curl.se/ca/cacert.pem
scp cacert.pem root@<target_ip_address>:/etc/curl/
```

4- Add the following line to end of your /etc/profile on target and restart or logout and back in for the change to take effect.

```
export CURL_CA_BUNDLE=/etc/curl/cacert.pem
```

5- If a /tmp directory does not exist, add one on your target. Please note that this will require having a writable / partition, otherwise you can create another partition and mount it on top of / or /tmp

```
mkdir /tmp
```

6- Update system time with ntpdate (on target). **Please use the appropriate time server for your region.** The following time server is for Canada, but others can be found at https://www.ntppool.org/zone.

```
ntpdate 0.ca.pool.ntp.org
```

7- Install pip on your target

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python3 get-pip.py
```

8- Install python dependencies on your target.

```
pip install -U \
colcon-common-extensions \
importlib-metadata \
importlib-resources \
lark-parser
```

9- create a directory for ROS2's installation.

```
mkdir -p /opt/ros/foxy
```

10- Get the ip address of your target

```
ifconfig
```

11- Check the amount of space available on your target and make sure you have enough space to copy the files over.

```
df -h
```

12- Copy ROS 2 to your target.

Note: you will have to replace "your_target_architecture" with your target architecture (e.g: "aarch64le" or "x86_64).

On host:

```
cd ~/ros2_foxy/install/<your_target_arch>/
tar -czvf ros2_foxy.tar.gz *
scp ros2_foxy.tar.gz root@target_ip_address:/opt/ros/foxy/
```

On target:

```
cd /opt/ros/foxy
tar -xzvf ros2_foxy.tar.gz
```

All the necessary files to run ROS 2 are now on your target.

13- Add the following lines to the end of your /etc/profile file

```
export COLCON_CURRENT_PREFIX=/opt/ros/foxy
export PYTHONPATH=/opt/ros/foxy/usr/lib/python3.8/site-packages
. /opt/ros/foxy/local_setup.sh
```

14- Logout and login or reboot. On QNX, a reboot can be done using *shutdown*.

Test the installation

1- ssh to your target and on one terminal run the following.

```
ros2 run demo_nodes_cpp talker
```

2- On another terminal run the following.

```
ros2 run demo_nodes_py listener
```

You should see the demos running on both terminals if the installation went successful.

1.1.4 Setting up a Workspace for ROS2 & QNX

Preferable host OS: Ubuntu 20.04

1- Clone the template workspace:

```
git clone http://gitlab.com/qnx/frameworks/ros2/dev_ws.git
```

This workspace contains the necessary setup, toolchain file and build script to cross compile for QNX.

2- Add your packages inside dev_ws/src

3- Set the value of ROS2_HOST_INSTALLATION_PATH inside build.sh according to the location of ROS2 installation is on your pc

4- Run the build command:

```
./build.sh
```

5- On target create a new directory for your group of packages:

```
mkdir /opt/dev_ws
```

6- Copy your packages over to the new location

```
scp -r ~/dev_ws/install/x86_64/* <user_name>@<ip_address>:/opt/dev_ws
```

7- Add the following commands at the end of the file /etc/.profile on your target:

```
export COLCON_CURRENT_PREFIX=/opt/ros/foxy
. /opt/ros/foxy/local_setup.sh
export COLCON_CURRENT_PREFIX=/opt/dev_ws
. /opt/dev_ws/local_setup.sh
```

8- Log out and log in back into new a terminal

9- Run your newly installed packages.

```
ros2 run my_new_package my_new_package_executable
```

1.1.5 Building a Docker Image for ROS2 & QNX Development

Table of Contents

- *Requirements*
- *Steps*

Docker is a tool that can be used to easily create environments that are reproducible and lightweight. In this tutorial we will use it to set up a development environment for building ROS2 for QNX.

Requirements

- Docker-CE
  - Can be setup on Ubuntu using Docker's official convenience script:

```
curl https://get.docker.com | sh && sudo systemctl --now enable docker
```

- QNX SDP 7.1.0
  - Official QNX SDP 7.1.0 install instructions

Steps

1- Clone the Dockerfile.

```
git clone https://gitlab.com/qnx/frameworks/ros2/docker
```

2- Prepare the docker build context with the QNX SDP. If the SDP is located in your home directory, run the following.

```
rsync -havz ~/qnx710 ./docker/
```

We use `rsync -havz` rather than a regular `cp` to preserve the symbolic links inside the SDP. If this is not done, the size of the copied SDP will be significantly larger.

3- Build the docker image with the included script.

```
cd docker
./docker-build-qnxros2-image.sh foxy
```

4- Run the image with the included script to create a container. This will open an interactive terminal into the Docker container.

```
./docker-create-container.sh foxy
```

5- Inside of the container's interactive terminal, set the CPU environment variable to the target cpu architecture for the build of ROS2. For example:

```
export CPU=x86_64
# or
export CPU=aarch64
# or
unset CPU # Build for all supported architectures
```

6- Perform the rest of the operations displayed by the welcome message to build ROS2.

```
cd ros2_foxy
./build-ros2.sh
```