Raspberry Pi cross-compilation

This guide will be applicable starting with the upcoming Kaa 0.8.0 release.

This guide explains how to build Kaa C/C++ endpoint SDKs for Raspberry Pi on a Linux machine.

Verified against:

Host OS: Ubuntu 14.04 LTS 32-bit LTS

Target OS: Raspbian Jessie (2015-11-21)

The further instructions must be executed **on the host machine**.

Cross-compiling Kaa C SDK

1. Download and install toolchain.

```
mkdir rpi_root && cd rpi_root && export RPI_ROOT=`pwd`
git clone https://github.com/raspberrypi/tools.git
export INSTALLDIR=`pwd`/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian
export PATH=$PATH:$INSTALLDIR/bin
```

2. Download and install openssl.

```
wget https://www.openssl.org/source/openssl-1.0.2e.tar.gz
tar -xf openssl-1.0.2e.tar.gz
cd openssl-1.0.2e && ./Configure --openssldir=${INSTALLDIR}/arm-linux-gnueabihf/libc/usr shared os
/compiler:arm-linux-gnueabihf-gcc
make && make install
```

3. Go to the Kaa c-client folder and run the following commands.

```
mkdir build && cd build && cmake -DCMAKE_TOOLCHAIN_FILE=../toolchains/rpi.cmake .. && make
```

Cross-compiling Kaa C++ SDK

1. Download and install toolchain, create necessary folders to keep all build environment in a certain place.

```
mkdir rpi_root && cd rpi_root && export RPI_ROOT=`pwd`
git clone https://github.com/raspberrypi/tools.git
export INSTALLDIR=`pwd`/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian
export PATH=$PATH:$INSTALLDIR/bin && export CROSS=arm-linux-gnueabihf
export CC=${CROSS}-gcc && export LD=${CROSS}-ld && export AS=${CROSS}-as && export AR=${CROSS}-ar
```

2. Download and install zlib.

```
wget http://zlib.net/zlib-1.2.8.tar.gz
tar -xvzf zlib-1.2.8.tar.gz
cd zlib-1.2.8
./configure --prefix=${INSTALLDIR}/arm-linux-gnueabihf/libc/usr
make && make install && cd $RPI_ROOT
```

3. Download and install bzib2.

```
wget http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz
tar -xvzf bzip2-1.0.6.tar.gz
cd bzip2-1.0.6
sed -e "/^all:/s/ test//" Makefile > Makefile-libbz2_so
make -f Makefile-libbz2_so CC="${CC}" AR="${AR}"
make PREFIX=${INSTALLDIR}/arm-linux-gnueabihf/libc/usr install && cd $RPI_ROOT
```

4. Download and install boost.

```
wget -O boost_1_59_0.tar.bz2 http://sourceforge.net/projects/boost/files/boost/1.59.0/boost_1_59_0.
tar.bz2/download
tar -xvjf boost_1_59_0.tar.bz2 && cd boost_1_59_0 && ./bootstrap.sh
```

Edit the project-config.jam file. Add 'using gcc : arm : arm-linux-gnueabihf-c++ ;' instead of 'using gcc ;':

```
./bjam install toolset=gcc-arm --prefix=${INSTALLDIR}/arm-linux-gnueabihf/libc/usr
cd $RPI_ROOT
```

5. Install Avro.

```
wget https://archive.apache.org/dist/avro/avro-1.7.5/cpp/avro-cpp-1.7.5.tar.gz
tar -xvzf avro-cpp-1.7.5.tar.gz
cd ./avro-cpp-1.7.5
wget http://docs.kaaproject.org/download/attachments/15796115/avro-1.7.5_build.patch?api=v2
patch < avro-1.7.5_build.patch?api=v2
mkdir build && cd build
cmake -DCMAKE_INSTALL_PREFIX:PATH=${INSTALLDIR}/arm-linux-gnueabihf/libc/usr -DCMAKE_TOOLCHAIN_FILE=..
/rpi.cmake ..
make && make install && cd $RPI_ROOT
```

6. Install Botan.

```
wget http://botan.randombit.net/releases/Botan-1.11.27.tgz
tar -xvzf Botan-1.11.27.tgz
cd Botan-1.11.27 && python configure.py --cpu=arm --cc-bin=${CROSS}-g++ --prefix=${INSTALLDIR}/arm-
linux-gnueabihf/libc/usr
make && make install
mv ${INSTALLDIR}/arm-linux-gnueabihf/libc/usr/include/botan-1.11/botan/ ${INSTALLDIR}/arm-linux-
gnueabihf/libc/usr/include
rm -r ${INSTALLDIR}/arm-linux-gnueabihf/libc/usr/include/botan-1.11
```

7. Go to the kaa cpp-client directory and create the build folder.

```
cd /path/to/kaa/kaa/client/client-multi/client-cpp
mkdir build && cd build && cmake -DCMAKE_TOOLCHAIN_FILE=../toolchains/rpi.cmake .. && make
```