Qualcomm Technologies, Inc.

Qualcomm Intelligent Multimedia Product (QIMP) SDK Quick Start Guide

80-70018-51 AD

May 1, 2025

Contents

1 QIMP SDK overview

The Qualcomm® Intelligent Multimedia Product (QIMP) SDK is a collection of four standalone function SDKs, namely, Qualcomm® Intelligent Multimedia SDK (IM SDK), Qualcomm® Neural Processing SDK, Qualcomm® AI Engine direct SDK, and TensorFlow Lite (currently known as Lite Runtime or LiteRT). It also includes reference applications that you can use to develop use cases.

This guide explains how to:

- Explore the inbuilt sample applications and command-line use cases.
- Create applications using the QIMP SDK workflow.
- Compile the Extensible SDK (eSDK) from source and upgrade function SDKs.

Get started

→ QIMP SDK workflow

→ Download and install eSDK

Develop applications

→ Develop your first application

→ Customize existing sample applications

→ Troubleshoot common issues

Customize eSDK

→ Build platform image with QIMP layer

→ Build a custom eSDK

→ Upgrade QIMP SDK

→ Upgrade other SDKs

Important:

- Ensure that the host machine uses Ubuntu 22.04.

- The commands in this guide are compatible with Qualcomm® Linux® 1.4. Verify your Qualcomm Linux release version, and if it isn't 1.4, update the software.
- The sample applications and AI procedures in this guide are compatible with Qualcomm AI® Runtime SDK v2.32 and LiteRT (or TFLite) v2.16.1. Ensure that you download the matching SDKs to your host machine before starting AI/ML development..
- See hardware SoCs that are supported on Qualcomm Linux.

1.1 Architecture

The figure shows the various software components of the QIMP SDK.

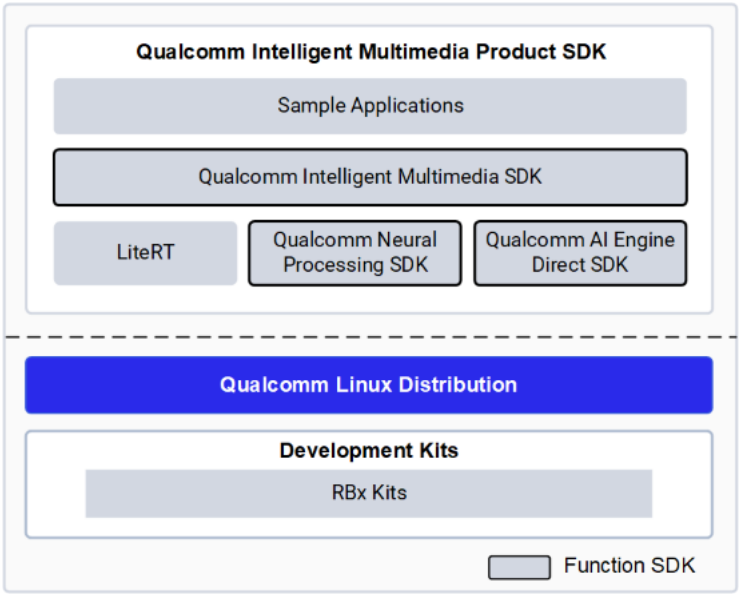


Figure : QIMP SDK software stack

1.2 QIMP SDK components

The QIMP SDK integrates specialized components called function SDKs, listed as follows:

Function SDKs in QIMP SDK

Function SDK	Description
Qualcomm IM SDK	Provides Qualcomm hardware-accelerated Gstreamer-based plugins for optimized application development.

Function SDK	Description
Qualcomm Neural Processing SDK	• It is also known as Snapdragon Neural Processing Engine (SNPE) • It is a software-accelerated runtime for executing deep neural networks. • SNPE offers tools to convert and quantize neural networks, and accelerate them on hardware accelerators including CPU, GPU, and Qualcomm® Hexagon™ Tensor Processor.
Qualcomm AI Engine direct SDK	• It provides a unified Qualcomm Neural Network (QNN) API and tools to accelerate the AI/ML models, use cases on Qualcomm chipsets and AI acceleration cores. • It provides a unified API and modular and extensible per-accelerator libraries, which form a reusable basis for full-stack AI solutions. • It supports runtimes such as Qualcomm Neural Processing SDK and LiteRT AI Engine direct delegate.
LiteRT	LiteRT is an open-source deep learning framework designed for on-device inference. QIMP includes delegates and tools from the compatible LiteRT package.

1.3 Sample applications

The QIMP SDK sample applications show how to use the SDKs and develop end-to-end edge analytics use cases. For more information, see Sample applications.

1.4 Qualcomm Linux distribution

Qualcomm Linux is a Linux distribution that supports the QIMP SDK. For more information on Qualcomm Linux, see Software Overview.

1.5 Development kits

Development kits are Qualcomm reference devices built to support Qualcomm Linux. They're used to develop and test software applications. A development kit includes necessary hardware and interfaces such as sensors, inertial measurement units (IMU), and other peripherals. You can use these development kits to test the capabilities of Qualcomm Linux. For more information, see Development Kits.

1.6 Supported component versions

The current release of the QIMP SDK integrates the following components:

Components	Version
Qualcomm Neural Processing SDK	2.32.0.250228
Qualcomm AI Engine direct SDK	2.32.0.250228
LiteRT	2.16.1
GStreamer	1.22.12

1.7 QIMP eSDK

The eSDK is a package generated from the Qualcomm Linux image. It's installed on an Ubuntu host and provides a Yocto-based environment that you can use for application development.

The eSDK uses the devtool mechanism of Yocto Project to download, compile, and install reference applications and plugins.

The following figure shows the components of the QIMP eSDK.

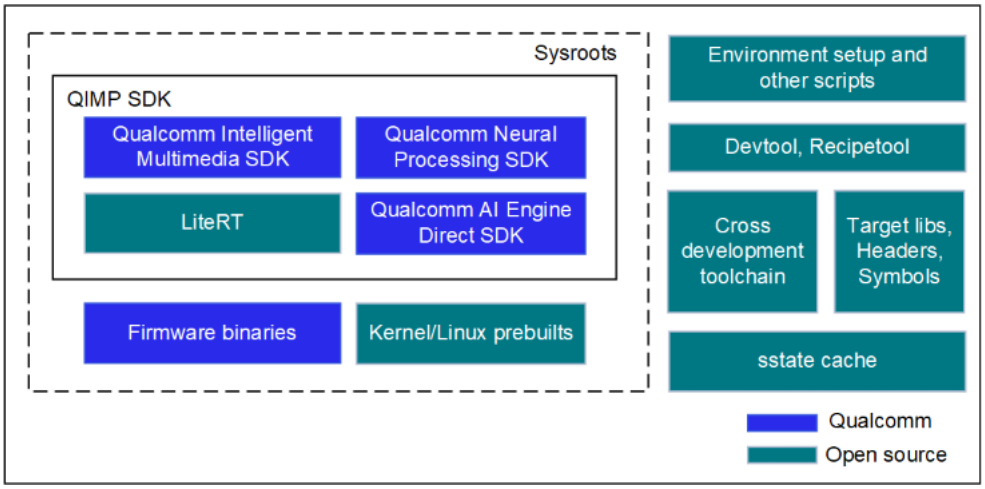


Figure : QIMP eSDK

2 Get started with the QIMP SDK

This section provides instructions on how to start developing your software using the QIMP SDK.

2.1 QIMP SDK workflow

The figure shows the SDK workflow from setting up the device to developing your first application. For more information, click each component.

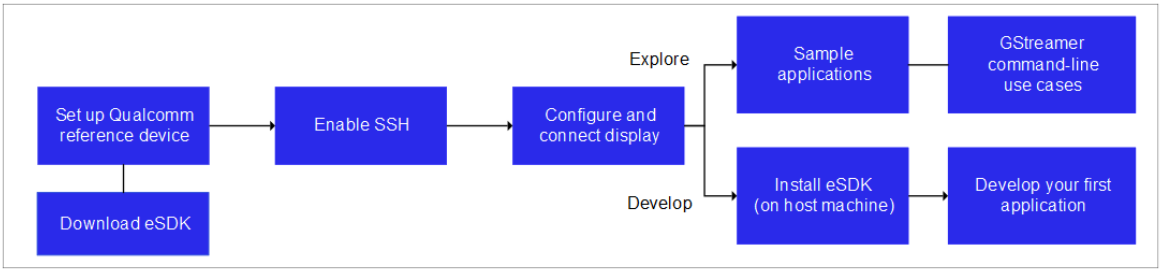


Figure : QIMP SDK workflow

2.2 Prerequisites

- Ensure that your development kit has the latest Qualcomm Linux software. Verify the Qualcomm Linux version for the latest software.
- Else, upgrade your development kit following the steps described in Set up the device.

2.3 Explore sample applications with Qdemo

You can explore sample applications using the QDemo application, which is part of the software installation. You can run this GUI-based application directly on the target device without needing to set up a host. For more details, see Run sample applications using Qdemo.

2.4 Explore eSDK

Run the sample applications that are part of the QIMP package and gst command-line use cases to understand the eSDK capabilities.

2.5 Download and install eSDK

Note: For the QCS8275, a downloadable eSDK is not available. However, if you are a registered user with access to the QCS8275 software, you have the option to build a custom eSDK.

After upgrading the development kit to the latest Qualcomm Linux software, download the corresponding eSDK.

This release supports separate eSDKs for the Qualcomm® RB3 Gen 2 Vision Development Kit, the Qualcomm® RB3 Gen 2 Core Development Kit, and Qualcomm Dragonwing™ IQ-9075 Evaluation Kit (EVK). You can download the eSDK that corresponds to your specific development kit.

Prerequisites

- An Ubuntu 22.04 host machine with at least 100 GB of free space. To set up the virtual machine running Ubuntu 22.04 OS on a Microsoft Windows or an Apple® Mac®, see Qualcomm Linux Virtual Machine Setup Guide.
- `sudo` permission to run the commands.

Procedure

1. Download the eSDK from Qualcomm public archive.
   a. Make directory:

   ```
   mkdir <workspace_path>
   ```

   b. Change directory:

   ```
   cd <workspace_path>
   ```

   c. Download the zipped file:

   - For RB3 Gen 2 Vision Kit:
     – Ubuntu x86 architecture-based host machines:

     ```
     wget https://artifacts.codelinaro.org/artifactory/
     qli-ci/flashable-binaries/qimpsdk/qcs6490-rb3gen2-
     vision-kit/x86-qcom-6.6.65-QLI.1.4-Ver.1.1_qim-
     product-sdk-1.1.2.zip
     ```

  - Arm® architecture-based host machines:

```
wget https://artifacts.codelinaro.org/artifactory/qli-ci/flashable-binaries/qimpsdk/qcs6490-rb3gen2-vision-kit/arm-qcom-6.6.65-QLI.1.4-Ver.1.1_qim-product-sdk-1.1.2.zip
```

- For Dragonwing IQ-9075 EVK:
  - Ubuntu x86 architecture-based host machines:

```
wget https://artifacts.codelinaro.org/artifactory/qli-ci/flashable-binaries/qimpsdk/qcs9075-rb8-core-kit/x86-qcom-6.6.65-QLI.1.4-Ver.1.1_qim-product-sdk-1.1.2.zip
```

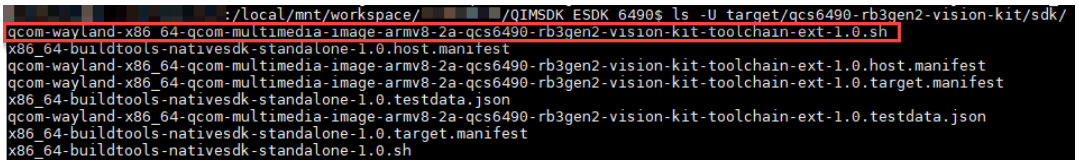
  - Arm architecture-based host machines:

```
wget https://artifacts.codelinaro.org/artifactory/qli-ci/flashable-binaries/qimpsdk/qcs9075-rb8-core-kit/arm-qcom-6.6.65-QLI.1.4-Ver.1.1_qim-product-sdk-1.1.2.zip
```

d. Unzip the QIMP SDK to a directory of your choice:

```
unzip x86-qcom-6.6.65-QLI.1.4-Ver.1.1_qim-product-sdk-1.1.2.zip
```

e. Ensure that the eSDK installer is at `<unzip_location>/target/qcs6490-rb3gen2-vision-kit/sdk/`.



f. If you don't have the required write permissions to install the eSDK in the directory, the installer alerts you and then terminates the installation. Ensure that you set up the required permissions in the directory and rerun the installer.

```
umask a+rx
```

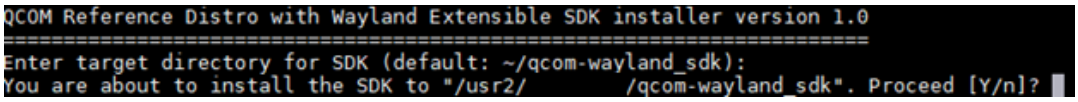
2. To install the eSDK, run the installer script from `<unzip_location>/target/qcs6490-rb3gen2-vision-kit/sdk/`:

```
sh ./qcom-wayland-x86_64-qcom-multimedia-image-armv8-2a-<machine>-toolchain-ext-1.0.sh
```

For example:

```
sh ./qcom--wayland-x86_
64-qcom-multimedia-image-armv8-2a-qcs6490-rb3gen2-vision-kit-toolchain-ext-1.
0.sh
```

3. To install the eSDK in your host machine, follow the instructions on the console:
   - On the console, you will be prompted to specify the directory path for the eSDK installation. If you accept the default path, the eSDK will be installed in the user directory, as shown in the following screenshot:

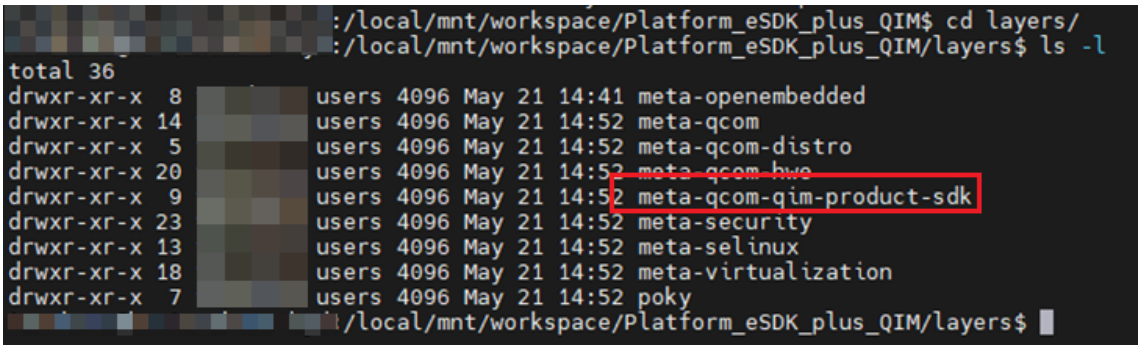


   - Alternatively, if you specify a custom path, such as `/local/mnt/workspace/Platform_eSDK_plus_QIM`, the eSDK will be installed in that directory.

Note: If packages such as diffstat, gawk, gcc, and g++ are missing during the eSDK installation, see Build with standalone commands.

---

4. To verify if the installation is successful, check if the QIMP SDK layers are available at `<workspace_path>/layers`.



Note: If you are an advanced user, you can build a custom eSDK.

---

5. Set up the `ESDK_ROOT` variable.

```
export ESDK_ROOT=<path of installation directory>
```

For example,

```
export ESDK_ROOT=/local/mnt/workspace/Platform_eSDK_plus_QIM
```

The QIMP SDK installation is now complete. Next, you can develop an application using the SDK, see Develop your first application.

3 Develop applications with the QIMP SDK

The figure shows the workflow for creating an application or a plugin using the QIMP SDK.

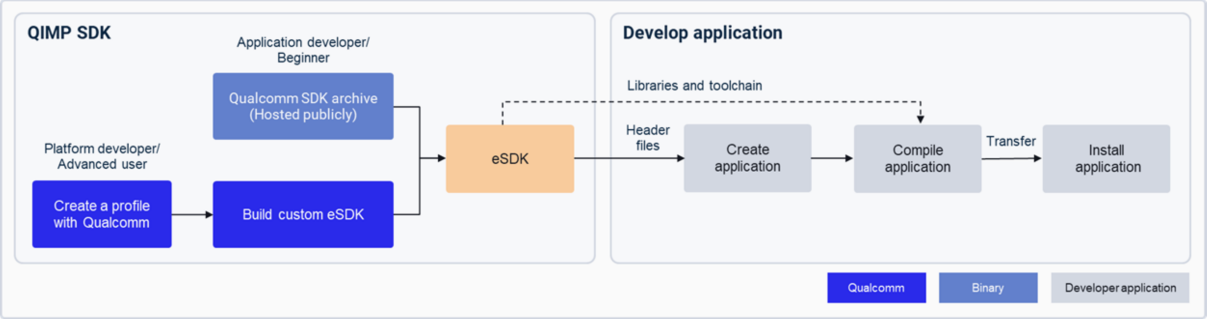


Figure : Application development using QIMP SDK

- Download and install eSDK on the host machine.

  It's recommended that you install eSDK from the Qualcomm public archive or compile it independently.

- Develop applications:

  1. Compile the application in an installable package (`.ipk`) using the following from the QIMP SDK:
     - Cross-compiler
     - Dependent libraries
     - Header files
  2. Transfer this `.ipk` to the target device.
  3. Use opkg utility from the Yocto project to install the ipk.

3.1 Develop your first application

You can create applications using one of the following methods:

- Qualcomm® Visual Studio Code
- A standalone Makefile
- The Yocto devtool utility

Develop using Qualcomm Visual Studio Code Extension

The recommended approach for creating applications is to use the Qualcomm Visual Studio Code Extension, which integrates application development for QLI platforms.

Before you begin, ensure that both the Host and Target are configured for the Qualcomm Visual Studio Code Extension, as outlined in the Quick Start.

Follow the steps provided in Projects to configure and create applications using the Qualcomm Visual Studio Code Extension.

Develop using Makefile

Use the Hello-QIM sample application to experience the capabilities of Qualcomm Linux. The sample application is hosted on GitHub.

Note: The Hello-QIM sample application requires a working MIPI camera sensor on the Qualcomm reference devices.

This procedure gets buffer samples from a camera pipeline with the `gst-appsink-example` GStreamer application, which uses the `app-sink` plugin.

1. After installing the eSDK, set the `ESDK_ROOT`:

   ```
   export ESDK_ROOT=<path of installation directory>
   ```

   For example,

   ```
   export ESDK_ROOT=/local/mnt/workspace/Platform_eSDK_plus_QIM
   ```

2. Go to the directory where the SDK was installed:

   ```
   cd $ESDK_ROOT
   ```

3. Set up the source environment:

   ```
   source environment-setup-armv8-2a-qcom-linux
   ```

4. Download the Hello-QIM sample application from GitHub:

```
git clone https://github.com/quic/sample-apps-for-qualcomm-linux
```

5. Go to the Hello-QIM application:

```
cd sample-apps-for-qualcomm-linux/Hello-QIM
```

6. Set the environment variables:

   - `export SDKTARGETSYSROOT=<path to Installation directory of platform SDK>/tmp/sysroots`

     For example:

     ```
     export SDKTARGETSYSROOT=$ESDK_ROOT/tmp/sysroots
     ```

Note: For Arm architecture-based host machine, change `Hello-QIM/Makefile` with `aarch64`:

```
CXX=${SDKTARGETSYSROOT}/aarch64/usr/bin/aarch64-qcom-linux/
aarch64-qcom-linux-g++
```

   - `export MACHINE=<Machine name>`

     For example, for the RB3 Gen 2 Vision Development Kit, add `qcs6490-rb3gen2-vision-kit`.

     ```
     export MACHINE=qcs6490-rb3gen2-vision-kit
     ```

   - `export GST_APP_NAME=<appname>`

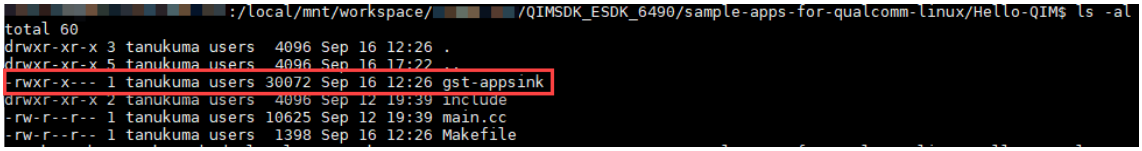
     For example:

     ```
     export GST_APP_NAME=gst-appsink
     ```

7. Compile the application:

```
make
```

After successful compilation, the application binary is generated.



8. To run the compiled program, do the following:

   a. Transfer the program to the Qualcomm Linux development kit:

   ```
   scp -r gst-appsink root@[IP Address of the device]:/opt/
   ```

   Enable SSH to access your Qualcomm Linux development kit securely. For instructions, see Sign in using SSH.

   b. Sign in to the SSH shell and run the sample application:

   ```
   ssh root@[IP Address of the device]
   ```

   Note: If prompted for a password, enter `oelinux123`.

   ```
   chmod 777 /opt/gst-appsink
   ```

   In this command, `gst-appsink` is the name of the sample application.

   ```
   cd /opt/
   ```

   ```
   ./gst-appsink -w 1280 -h 720
   ```

After the application is executed, the following message is displayed:

```
Hello-QIM: Success creating pipeline and received camera frame.
```

Develop using devtool

The eSDK provides the devtool utility to create or customize applications.

The following steps outline how to compile and generate sample applications, using your own, such as `gst-xx-xx-app`.

The existing `qcom-gst-ai-classification` application is used to build and generate the reference application `gst-ai-example-classification`, assuming it is `gst-xx-xx-app`.

To create the `gst-ai-example-classification` sample application, integrate it as part of the devtool workflow, and generate an installable IPK, do the following:

Note: The source code must be available for the `gst-ai-example-classification` sample application.

---

1. Set up the environment:

   ```
   cd $ESDK_ROOT
   ```

   ```
   source environment-setup-armv8-2a-qcom-linux
   ```

   ```
   [redacted]:/local/mnt/workspace/AI_Workflow$ source environment-setup-armv8-2a-qcom-linux
   SDK environment now set up; additionally you may now run devtool to perform development tasks.
   Run devtool --help for further details.
   ```

2. Add a BitBake file under `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer-sample-apps`.

   For example:

   ```
   $ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/
   gstreamer-sample-apps/qcom-gst-ai-example-classification.bb
   ```

   a. Copy the contents from an existing sample application such as `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer-sample-apps/qcom-gst-ai-classification.bb` to the BitBake file of `qcom-gst-ai-example-classification.bb`.

   b. Update the following variables as shown in the following code snippet:

   ```
   SUMMARY = "My custom ref sample apps for GStreamer pipelines"
   ```

   ```
   SRC_URI = "file://gst-ai-example-classification"
   ```

```
S = "${WORKDIR}/gst-ai-example-classification"
```

```
inherit cmake pkgconfig

SUMMARY = "My custom ref sample apps for GStreamer pipelines"
SECTION = "multimedia"
DEPENDS += "gstreamer1.0-plugins-base"
DEPENDS += "qcom-gst-sample-apps-utils"
DEPENDS += "qcom-camera-server"

#SRCPROJECT = "git://git.codelinaro.org/clo/le/platform/vendor/qcom-opensource/gst-plugins-qti-oss.git;protocol=https"
#SRCBRANCH  = "imsdk.lnx.2.0.0.r2-rel"
#SRCREV     = "043ee4e6f6b43989fd100614b1fdc99e616edc1c"

#SRC_URI = "${SRCPROJECT};branch=${SRCBRANCH};rev=${SRCREV};subpath=gst-sample-apps/gst-ai-example-classification"
#S = "${WORKDIR}/gst-ai-example-classification"
SRC_URI = "file://gst-ai-example-classification"
S = "${WORKDIR}/gst-ai-example-classification"

# Install directries.
INSTALL_BINDIR := "${bindir}"
INSTALL_LIBDIR := "${libdir}"
INSTALL_CONFIG := "${sysconfdir}/configs/"

EXTRA_OECMAKE += "-DGST_VERSION_REQUIRED=1.20.7"
EXTRA_OECMAKE += "-DSYSROOT_INCDIR=${STAGING_INCDIR}"
EXTRA_OECMAKE += "-DSYSROOT_LIBDIR=${STAGING_LIBDIR}"
EXTRA_OECMAKE += "-DGST_PLUGINS_QTI_OSS_INSTALL_BINDIR=${INSTALL_BINDIR}"
EXTRA_OECMAKE += "-DGST_PLUGINS_QTI_OSS_INSTALL_LIBDIR=${INSTALL_LIBDIR}"
EXTRA_OECMAKE += "-DGST_PLUGINS_QTI_OSS_INSTALL_CONFIG=${INSTALL_CONFIG}"

FILES:${PN} += "${INSTALL_BINDIR}"
FILES:${PN} += "${INSTALL_LIBDIR}"
FILES:${PN} += "${INSTALL_CONFIG}"

SOLIBS = ".so*"
FILES_SOLIBSDEV = ""
```

3. Create a directory called `files` at `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer-sample-apps/files`.

   a. Create the `gst-ai-example-classification` directory under `files`.

   b. Copy the existing `qcom-gst-ai-classification` application source code to the `gst-ai-example-classification` directory.

   Note: The `qcom-gst-ai-classification` source code can be downloaded using `devtool modify qcom-gst-ai-classification`.

```
                    :/local/mnt/workspace/         /GA.1.3.ESDK/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer-sample-apps$ tree files
files
└── gst-ai-example-classification
    ├── CMakeLists.txt
    ├── config_classification.json
    └── main.c

1 directory, 3 files
```

   c. Modify the `CMakeLists.txt` file to set the binary name to `gst-ai-example-classification`.

```
# Get the pkgconfigs exported by the automake tools
pkg_check_modules(GST
  REQUIRED gstreamer-1.0>=${GST_VERSION_REQUIRED})

pkg_check_modules(GST_JSON
  REQUIRED json-glib-1.0)

include_directories(${CMAKE_CURRENT_BINARY_DIR})

set(GST_EXAMPLE_BIN gst-ai-example-classification)

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall -Wextra -Werror")
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wno-unused-parameter")
```

4. To include `qcom-gst-ai-example-classification` in the packagegroup, add it under `RDEPENDS` in the `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/packagegroups/packagegroup-qcom-gst-sample-apps.bb` directory.

```
inherit packagegroup

PACKAGES = "${PN}"

RDEPENDS:${PN}:qcom-custom-bsp = " \
    qcom-gst-sample-apps-utils \
    qcom-gst-activate-deactivate-streams-runtime \
    qcom-gst-add-remove-streams-runtime \
    qcom-gst-add-streams-as-bundle-example \
    qcom-gst-ai-classification \
    qcom-gst-ai-daisychain-detection-classification \
    qcom-gst-ai-daisychain-detection-pose \
    qcom-gst-ai-monodepth \
    qcom-gst-ai-multi-input-output-object-detection \
    qcom-gst-ai-multistream-inference \
    qcom-gst-ai-object-detection \
    qcom-gst-ai-parallel-inference \
    qcom-gst-ai-pose-detection \
    qcom-gst-ai-segmentation \
    qcom-gst-ai-superresolution \
    qcom-gst-appsink-example \
    qcom-gst-audio-decode-example \
    qcom-gst-audio-encode-example \
    qcom-gst-audio-video-encode \
    qcom-gst-audio-video-playback \
    qcom-gst-camera-burst-capture-example \
    qcom-gst-camera-metadata-example \
    qcom-gst-camera-shdr-ldc-eis-example \
    qcom-gst-camera-single-stream-example \
    qcom-gst-camera-switch-example \
    qcom-gst-concurrent-videoplay-composition \
    qcom-gst-multi-camera-example \
    qcom-gst-multi-stream-example \
    qcom-gst-smartcodec-example \
    qcom-gst-snapshot-stream-example \
    qcom-gst-transform-example \
    qcom-gst-usb-single-camera-app \
    qcom-gst-videocodec-concurrent-playback \
    qcom-gst-video-playback-example \
    qcom-gst-video-transcode-example \
    qcom-gst-webrtc-sendrecv-example \
    qcom-gst-weston-composition-example \
    qcom-gst-ai-multistream-batch-inference \
    qcom-gst-ai-example-classification \
```

5. To include `qcom-gst-ai-example-classification:do_package_write_ipk` in packagegroup bbclass, add it under `GST_SAMPLE_APPS` in the `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/classes/qimsdk-pkg.bbclass` directory.

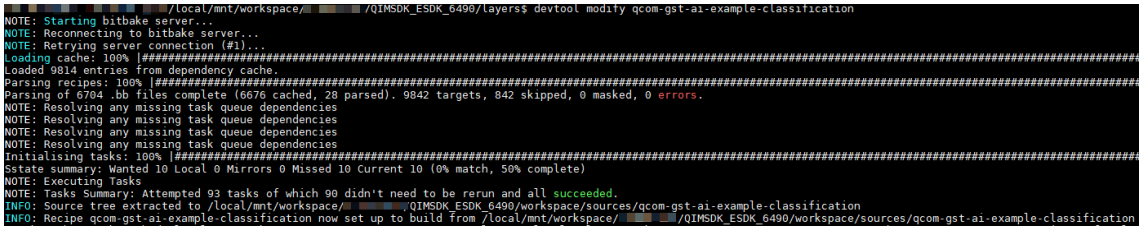
```
GST_SAMPLE_APPS = " \
    qcom-gst-sample-apps-utils:do_package_write_ipk \
    qcom-gst-activate-deactivate-streams-runtime:do_package_write_ipk \
    qcom-gst-add-remove-streams-runtime:do_package_write_ipk \
    qcom-gst-add-streams-as-bundle-example:do_package_write_ipk \
    qcom-gst-ai-classification:do_package_write_ipk \
    qcom-gst-ai-daisychain-detection-classification:do_package_write_ipk \
    qcom-gst-ai-daisychain-detection-pose:do_package_write_ipk \
    qcom-gst-ai-monodepth:do_package_write_ipk \
    qcom-gst-ai-multi-input-output-object-detection:do_package_write_ipk \
    qcom-gst-ai-multistream-inference:do_package_write_ipk \
    qcom-gst-ai-object-detection:do_package_write_ipk \
    qcom-gst-ai-parallel-inference:do_package_write_ipk \
    qcom-gst-ai-pose-detection:do_package_write_ipk \
    qcom-gst-ai-segmentation:do_package_write_ipk \
    qcom-gst-ai-superresolution:do_package_write_ipk \
    qcom-gst-appsink-example:do_package_write_ipk \
    qcom-gst-audio-decode-example:do_package_write_ipk \
    qcom-gst-audio-encode-example:do_package_write_ipk \
    qcom-gst-audio-video-encode:do_package_write_ipk \
    qcom-gst-audio-video-playback:do_package_write_ipk \
    qcom-gst-camera-burst-capture-example:do_package_write_ipk \
    qcom-gst-camera-metadata-example:do_package_write_ipk \
    qcom-gst-camera-shdr-ldc-eis-example:do_package_write_ipk \
    qcom-gst-camera-single-stream-example:do_package_write_ipk \
    qcom-gst-camera-switch-example:do_package_write_ipk \
    qcom-gst-concurrent-videoplay-composition:do_package_write_ipk \
    qcom-gst-multi-camera-example:do_package_write_ipk \
    qcom-gst-multi-stream-example:do_package_write_ipk \
    qcom-gst-smartcodec-example:do_package_write_ipk \
    qcom-gst-snapshot-stream-example:do_package_write_ipk \
    qcom-gst-transform-example:do_package_write_ipk \
    qcom-gst-usb-single-camera-app:do_package_write_ipk \
    qcom-gst-videocodec-concurrent-playback:do_package_write_ipk \
    qcom-gst-video-playback-example:do_package_write_ipk \
    qcom-gst-video-transcode-example:do_package_write_ipk \
    qcom-gst-webrtc-sendrecv-example:do_package_write_ipk \
    qcom-gst-weston-composition-example:do_package_write_ipk \
    qcom-gst-ai-multistream-batch-inference:do_package_write_ipk \
    qcom-gst-ai-example-classification:do_package_write_ipk \
```

6. Download the Qualcomm IM SDK sample application sources.

```
cd $ESDK_ROOT
```

```
devtool modify qcom-gst-ai-example-classification
```

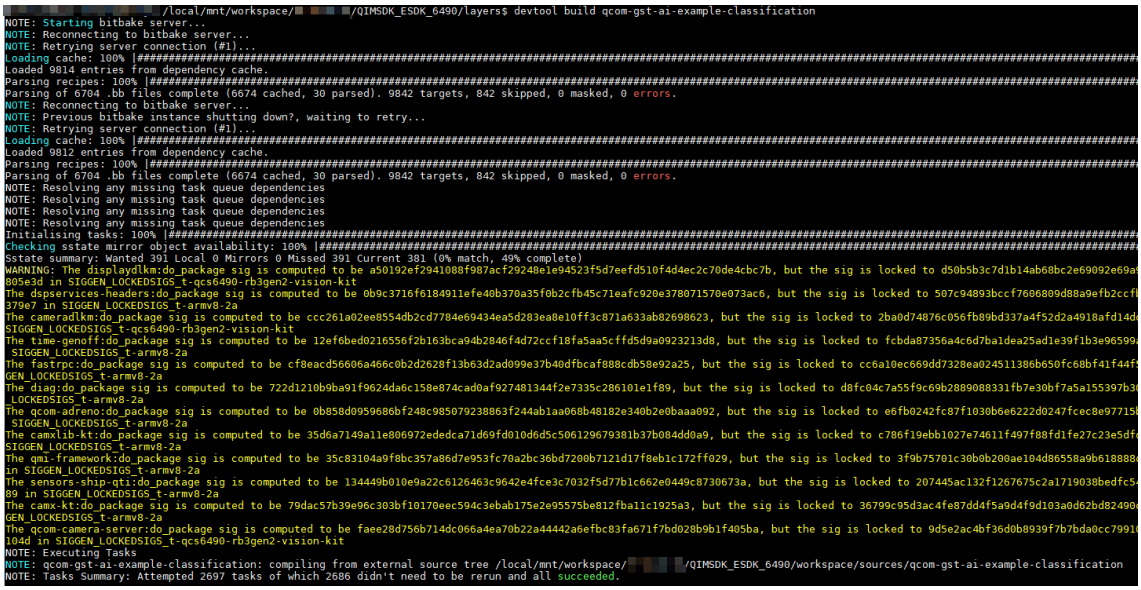
In this command, `qcom-gst-ai-example-classification` is the BitBake recipe that downloads the sources of the sample application.



The sample application source code is downloaded to `$ESDK_ROOT/workspace/sources/qcom-gst-ai-example-classification`.

7. Perform the required customizations and run the following command to rebuild the sample application:

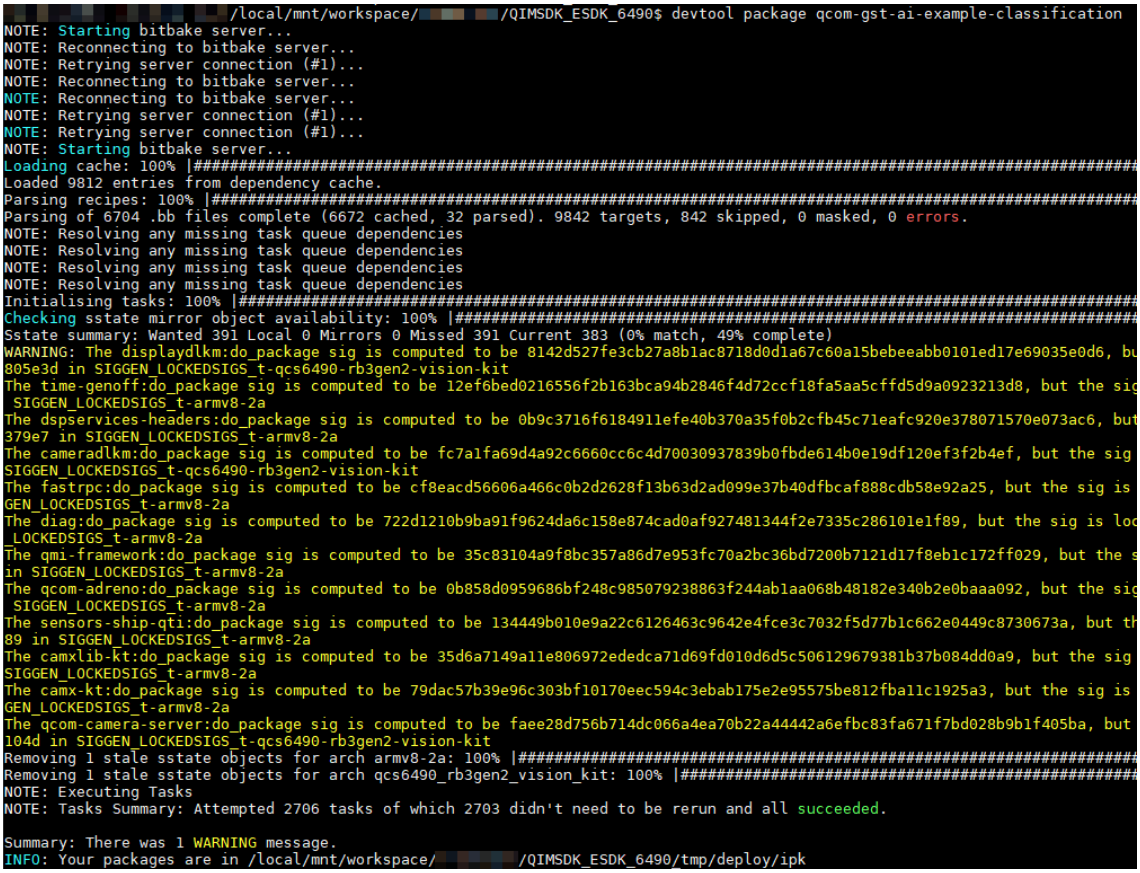
```
devtool build qcom-gst-ai-example-classification
```



8. To install the application on the Qualcomm Linux development kit, generate the sample application installer (IPK):

```
devtool package qcom-gst-ai-example-classification
```

The installable package is at `$ESDK_ROOT/tmp/deploy/ipk/armv8-2a/`.



9. Install compiled reference applications by copying the `qcom-gst-ai-example-classification_1.0-r0_armv8-2a.ipk` file to the Qualcomm Linux development kit.
10. Set up a connection with the Qualcomm Linux development kit through SSH as described in Sign in using SSH.

    The Qualcomm Linux development kit is accessible through its configured IP address.
11. Sign in to the SSH shell:

    ```
    ssh root@[ip-addr]
    ```

    Note: If prompted for a password, enter `oelinux123`.
12. On the Qualcomm Linux development kit, reconfigure the file system partition to support read and write:

    ```
    mount -o remount,rw /
    ```

```
exit
```

13. On the host machine, transfer the application installer to the Qualcomm Linux development kit:

```
cd $ESDK_ROOT/tmp/deploy/ipk/armv8-2a/
```

```
scp qcom-gst-ai-example-classification_1.0-r0_armv8-2a.ipk root@
<ip_address>:/opt/
```

14. Sign in to the SSH shell:

```
ssh root@[ip-addr]
```

15. Install the application on the Qualcomm Linux development kit:

```
opkg --force-depends --force-reinstall --force-overwrite install
/opt/qcom-gst-ai-example-classification_1.0-r0_armv8-2a.ipk
```

16. Verify if the IPK installation is successful.

```
Installing qcom-gst-ai-example-classification (1.0) on root
Configuring qcom-gst-ai-example-classification.
sh-5.1#
```

You can run the new application by starting it from the terminal.

For example:

- Enable the display:

```
export XDG_RUNTIME_DIR=/dev/socket/weston && export
WAYLAND_DISPLAY=wayland-1
```

- Run the application:

```
/usr/bin/gst-ai-example-classification
```

17. Verify if the application is executed and the screen displays the expected output.

3.2 Customize existing sample applications

To modify the sample applications, use the `devtool` utility from the eSDK to get the source code for sample applications. After making the required changes, use the `devtool` utility to compile and package the applications into an installable format (`ipk`).

Download and compile the existing sample application using the `devtool` utility of Yocto.

1. Download the source:

   a. Go to the directory where the eSDK was installed:

   ```
   cd <workspace root>
   ```

   In the command, `<workspace_root>` is the file system path where the eSDK is installed.

   b. Set up the source environment:

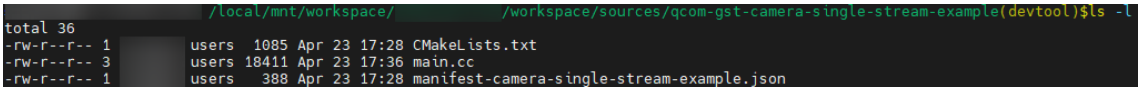
   ```
   source environment-setup-armv8-2a-qcom-linux
   ```

   c. Download the sample application sources:

   ```
   devtool modify qcom-gst-camera-single-stream-example
   ```

   The command `devtool modify` initializes the workspace and downloads the sample applications from the repository.

   The sources are at `<workspace_root>/workspace/sources/qcom-gst-camera-single-stream-example`.



2. Build the application:

   a. After you do the necessary customizations, run the following command to recompile:

   ```
   devtool build qcom-gst-camera-single-stream-example
   ```

   b. Generate an installable IPK file:

   ```
   devtool package qcom-gst-camera-single-stream-example
   ```

   The installable packages are at `<workspace_root>/tmp/deploy/ipk/armv8-2a/`.

3. Install the updated application:

   a. Push the application on the target device:

   ```
   opkg --force-reinstall install <.ipk that was generated >
   ```

   For example:

   ```
   opkg --force-reinstall install
   ```

```
qcom-gst-camera-single-stream-example_1.0-r0_armv8-2a.ipk
```

3.3 Compile and install Qualcomm IM SDK plugins

This section explains how to compile a Qualcomm IM SDK plugin and install it on the device, using the `mlvdetection` plugin as an example.

1. Set up the environment:

```
cd $ESDK_ROOT
```

```
source environment-setup-armv8-2a-qcom-linux
```

```
SDK environment now set up; additionally you may now run devtool
to perform development tasks.
Run devtool --help for further details.
[/local/mnt/workspace/QIMSDK_ESDK_6490/workspace/sources/qcom-
gst-ai-classification]$
```

2. Download the Qualcomm IM SDK plugin source code.

```
cd $ESDK_ROOT
```

```
devtool modify qcom-gstreamer1.0-plugins-oss-mlvdetection
```

Where `qcom-gstreamer1.0-plugins-oss-mlvdetection` is the BitBake recipe that downloads the source code for the plugin.

```
NOTE: Starting bitbake server...
Loading cache: 100% |#####################################|
Time: 0:00:00
Loaded 9816 entries from dependency cache.
Parsing recipes: 100% |###################################|
Time: 0:00:01
Parsing of 6700 .bb  files complete (6678 cached, 22 parsed).
9838 targets, 842 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |################################|
Time: 0:00:55
Sstate summary: Wanted 10 local 0 mirrors 0 Missed 10 Current 10
(0% match, 50% complete)
NOTE: Executing Tasks
```

```
NOTE: Tasks Summary: Attempted 93 tasks of which 90 didn't need
to be rerun and all succeeded.
INFO: Source tree extracted to /local/mnt/workspace/QIMSDK_ESDK_
6490/workspace/sources/qcom-gstreamer1.0-plugins-oss-
mlvdetection
INFO: Recipe qcom-gstreamer1.0-plugins-oss-mlvdetection now
setup to build from /local/mnt/workspace/QIMSDK_ESDK_6490/
workspace/sources/qcom-gstreamer1.0-plugins-oss-mlvdetection
```

The plugin source code is downloaded to `$ESDK_ROOT/workspace/sources/qcom-gstreamer1.0-plugins-oss-mlvdetection`.

```
[/local/mnt/workspace/QIMSDK_ESDK_6490/workspace/sources/qcom-
gstreamer1.0-plugins-oss-mlvdetection]$ ls
CMakeLists.txt    config.h.in    mlvdetection.h    modules
[/local/mnt/workspace/QIMSDK_ESDK_6490/workspace/sources/qcom-
gstreamer1.0-plugins-oss-mlvdetection]$
```

3. After customizing, use the following commands to rebuild the plugin.

```
devtool build qcom-gstreamer1.0-plugins-oss-mlvdetection
```

```
NOTE: Starting bitbake server...
Loading cache: 100% |#########################################
#####| Time: 0:00:00
Loaded 9816 entries from dependency cache.
Parsing recipes: 100% |#######################################
#####| Time: 0:00:01
Parsing of 6700 .bb  files complete (6678 cached, 22 parsed).
9838 targets, 842 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |####################################
#####| Time: 0:00:55
Checking sstate mirror object availability: 100% |############
#####| Time: 0:01:06
Sstate Summary: Wanted 264 Local 0 Mirrors 0 Missed 264 Current
255 (0% match, 49% complete)
Removing 1 stale sstate objects for arch qcs6490_rb3gen2_vision_
kit: 100% |###############################################|
Time: 0:00:12
Removing 1 stale sstate objects for arch armv8-2a: 100% |######
#########################################################|
Time: 0:00:16
NOTE: Executing Tasks
```

```
NOTE: qcom-gstreamer1.0-plugins-oss-mlvdetection: compiling from
external source tree /local/mnt/workspace/QIMSDK_ESDK_6490/
workspace/sources/qcom-gstreamer1.0-plugins-oss-mlvdetection
NOTE: Task Summary: Attempted 1836 tasks of which 1827 didn't
need to be rerun and all succeeded.
```

4. Generate the plugin installer (`.ipk`) to install the application on the device.

```
devtool package qcom-gstreamer1.0-plugins-oss-mlvdetection
```

   Verify that the installable package is present at the following path.

```
ls $ESDK_ROOT/tmp/deploy/ipk/armv8-2a | grep qcom-gstreamer1.0-
plugins-oss- mlvdetection
```

```
qcom-gstreamer1.0-plugins-oss-mlvdetection_1.0-r0_armv8-2a.ipk
qcom-gstreamer1.0-plugins-oss-mlvdetection_dbg_1.0-r0_armv8-2a.
ipk
qcom-gstreamer1.0-plugins-oss-mlvdetection_dev_1.0-r0_armv8-2a.
ipk
qcom-gstreamer1.0-plugins-oss-mlvdetection_src_1.0-r0_armv8-2a.
ipk
[/local/mnt/workspace/QIMSDK_ESDK_6490/workspace/sources/qcom-
gstreamer1.0-plugins-oss-mlvdetection]$
```

5. Set up an SSH connection with the device by following the steps provided in Sign in using SSH.

   When connected, the RB3 Gen 2 device is accessible through its configured IP address.

6. Copy the `qcom-gstreamer1.0-plugins-oss-mlvdetection_1.0-r0_armv8-2a.ipk` to the RB3 Gen 2 device to install the compiled reference apps.

7. Connect to the device through the SSH shell:

```
ssh root@<IP addr of the target device>
```

   Note: If prompted, enter `oelinux123` as the password for the SSH shell.

8. Run the following command on the target device.

```
mount -o remount,rw /
exit
```

9. Run the following commands on the host machine:

```
cd $ESDK_ROOT/tmp/deploy/ipk/armv8-2a/
```

```
scp qcom-gstreamer1.0-plugins-oss-mlvdetection_1.0-r0_armv8-2a.
ipk root@<IP addr of the target device>:/opt
```

10. Connect to the device through the SSH shell:

```
ssh root@<IP addr of the target device>
```

11. Run the following command on the target device:

```
opkg --force-depends --force-reinstall --force-overwrite install
/opt/qcom- gstreamer1.0-plugins-oss-mlvdetection_1.0-r0_armv8-
2a.ipk
```

12. Verify that the IPK installation is successful as shown below:

```
Installing qcom-gstreamer1.0-plugins-oss-mlvdetection (1.0) on
root
Configuring qcom-gstreamer1.0-plugins-oss-mlvdetection.
```

3.4 Create your Qualcomm IM SDK plugin

This section explains how to create a new plugin and compile it as part of the Qualcomm Intelligent Multimedia Product SDK. Each ML plugin is integrated into an independent BitBake recipe.

Update BitBake files for custom plugin compilation

1. Set up the environment.

```
cd $ESDK_ROOT
```

```
source environment-setup-armv8-2a-qcom-linux
```

```
SDK environment now set up; additionally you may now run devtool
to perform development tasks.
Run devtool --help for further details.
[/local/mnt/workspace/Platform_eSDK_plus_QIM]$
```

2. Add a new BitBake file under `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer/`

   For example, `$ESDK_`

`ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer/qcom-gstreamer1.0-plugins-oss-sample-mlvclassification.bb`

a. Copy the contents from an existing plugin like `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer/qcom-gstreamer1.0-plugins-oss-mlvclassification.bb` to this new file.

b. Update the following variables as shown below.

```
SUMMARY = "My custom plugin for ML image categorization"
SRC_URI = "file://gst-plugin-sample-mlvclassification"
S = "${WORKDIR}/gst-plugin-sample-mlvclassification"
```

```
...
inherit cmake pkgconfig

SUMMARY = "My custom plugin for ML image categorization"**
SECTION= "multimedia"

LICENSE = "BSD-3-Clause-Clear"
LIC_FILES_CHKSUM - "file://${COMMON_LICENSE_DIR}/${LICENSE};
md5=7a43440b65lf4a472ca93716d01033a"

# Dependencies.
DEPENDS := "gstreamer1.0"
DEPENDS += "gstreamer1.0-plugins-base"
DEPENDS += "qcom-gstreamer1.0-plugins-oss-base" DEPENDS +=
"cairo"

SRC_URI += "file://gst-plugin-sample-mlvclassification"
S ="${WORKDIR}/gst-plugin-sample-mlvclassification"

# Install directories.
INSTALL_INCDIR := "${includedir}"
...
```

3. Create a directory named files in `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/gstreamer/`.

4. Create the `gst-plugin-sample-mlvclassification` directory under the files directory and copy the Qualcomm IM SDK plugin source code inside the directory.

```
files/
|
+-- gst-plugin-sample-mlvclassification/
   |
```

```
   +-- CMakeLists.txt
   +-- config.h.in
   +-- mlvclassification.c
   +-- mlvclassification.h
   +-- modules/
      |
      +-- CMakeLists.txt
      +-- ml-vclassification-ocr.c
      +-- ml-vclassification-qfr.c
      +-- README
```

Note: This assumes that source code is available for the `sample-mlvclassification` plugin.

5. Update `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/recipes-gst/packagegroups/packagegroup-qcom-gst.bb` to add `qcom-gstreamer1.0-plugins-oss-sample-mlvclassification` in `RDEPENDS`.

```
RDEPENDS:${PN}:qcom-custom-bsp = " \
   ${PN}-dependencies \
   ${PN}-basic \
   ...
   qcom-gstreamer1.0-plugins-oss-sample-mlvclassification \
"
```

6. Update `$ESDK_ROOT/layers/meta-qcom-qim-product-sdk/classes/qimsdk-pkg.bbclass` to add `qcom-gstreamer1.0-plugins-oss-sample-mlvclassification:do_package_write_ipk`.

```
GST_PLUGINS = " \
   gstd:do_package_write_ipk \
   gstreamer1.0:do_package_write_ipk \
   ...
   qcom-gstreamer1.0-plugins-oss-sample-mlvclassification:do_package_write_ipk \
"
```

7. Download the Qualcomm IM SDK plugin source code.

```
cd $ESDK_ROOT
```

```
devtool modify qcom-gstreamer1.0-plugins-oss-sample-
mlvclassification
```

Where `qcom-gstreamer1.0-plugins-oss-sample-mlvclassification` is the BitBake recipe that downloads the plugin source code.

```
NOTE: Starting bitbake server ...
Loading cache: 100% |########################################
########################################| Time 0:00:02
Loaded 9814 entries from dependency cache.
Parsing recipes: 100% |######################################
########################################| Time 0:00:03
Parsing of 6702 .bb files complete (6670 cached, 32 parsed).
9840 targets, 842 skipped, 0 masked, 0 errors.

Summary: There were 0 WARNING messages.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |###################################
########################################| Time 0:00:08
Sstate summary: Wanted 10 Local 0 Mirrors 0 Missed 10 Current 10
(0% match, 50% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 93 tasks of which 90 didn't need
to be rerun and all succeeded.
INFO: Source tree extracted to /local/mnt/workspace/QIMSDK_ESDK_
6490/workspace/sources/qcom-gstreamer1.0-plugins-oss-sample-
mlvclassification
INFO: Recipe qcom-gstreamer1.0-plugins-oss-sample-
mlvclassification now set up to build from /local/mnt/workspace/
QIMSDK_ESDK_6490/workspace/sources/qcom-gstreamer1.0-plugins-
oss-sample-mlvclassification
```

8. After customizing, use the following command to rebuild the plugin.

```
devtool build qcom-gstreamer1.0-plugins-oss-sample-
mlvclassification
```

```
NOTE: Starting bitbake server ...
Loading cache: 100% |                                        | ETA
-:--:--
Loaded 0 entries from dependency cache.
...
```

9. Generate the plugin installer (.ipk) to install the application on the device.

```
devtool package qcom-gstreamer1.0-plugins-oss-sample-
mlvclassification
```

```
NOTE: Starting bitbake server ...
Loading cache: 100% |#########################################
########################| ETA 0:00:14
Loaded 9812 entries from dependency cache.
Parsing recipes: 100% |#######################################
#####################################| Time 0:00:03
Parsing of 6702 .bb files complete (6674 cached, 28 parsed).
9840 targets, 842 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |####################################
#####################################| Time 0:00:10
Checking sstate mirror object availability: 100% |############
#####################################| Time 0:00:00
Sstate summary: Wanted 262 Local 0 Mirrors 0 Missed 262 Current
259 (0% match, 49% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 1847 tasks of which 1847 didn't
need to be rerun and all succeeded.

Summary: There were 0 WARNING messages.
INFO: Your packages are in /local/mnt/workspace/QIMSDK_ESDK_
6490/tmp/deploy/ipk
```

10. Verify that the installable package is present at the following path.

```
ls $ESDK_ROOT/tmp/deploy/ipk/armv8-2a | grep qcom-gstreamer1.0-
plugins-oss-sample-mlvclassification
```

```
qcom-gstreamer1.0-plugins-oss-sample-mlvclassification_1.0-r0_
armv8-2a.ipk
qcom-gstreamer1.0-plugins-oss-sample-mlvclassification_dbg_1.0-
r0_armv8-2a.ipk
qcom-gstreamer1.0-plugins-oss-sample-mlvclassification_dev_1.0-
r0_armv8-2a.ipk
qcom-gstreamer1.0-plugins-oss-sample-mlvclassification_src_1.0-
r0_armv8-2a.ipk
[/local/mnt/workspace/Platform_eSDK_plus_QIM]$
```

11. Set up an SSH connection with the device by following the steps provided in Sign in using SSH.

When successful, the RB3 Gen 2 device is accessible through its configured IP address.

12. Copy the `qcom-gstreamer1.0-plugins-oss-sample-mlvclassification_1.0-r0_armv8-2a.ipk` to the RB3 Gen 2 device to install the compiled reference apps.
13. Connect to the device through the SSH shell:

```
ssh root@<IP addr of the target device>
```

Note: If prompted, enter `oelinux123` as the password for the SSH shell.

14. Run the following command on the target device.

```
mount -o remount,rw /
exit
```

15. Run the following commands on the host machine:

```
cd $ESDK_ROOT/tmp/deploy/ipk/armv8-2a/
```

```
scp qcom-gstreamer1.0-plugins-oss-sample-mlvclassification_1.0-r0_armv8-2a.ipk root@< IP addr of the target device>:/opt/
```

16. Connect to the device through the SSH shell:

```
ssh root@<IP addr of the target device>
```

17. Run the following command on the target device:

```
opkg --force-depends --force-reinstall --force-overwrite install /opt/qcom-gstreamer1.0-plugins-oss-sample-mlvclassification_1.0-r0_armv8-2a.ipk
```

The new plugin is now installed and can be used by applications that need the plugin.

3.5 Troubleshoot common issues

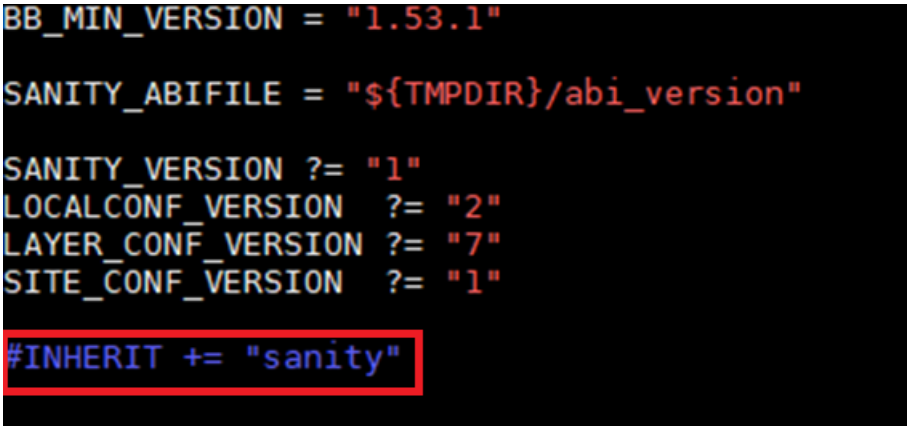
Occasionally, the `devtool` might produce sanity errors. Ensure that you have `sudo` access on the host machine.

If the error continues, do the following:

1. Set the default permissions for newly created files and directories:

```
umask a+rx
```

2. Disable the BitBake sanity checking in the `$ESDK_ROOT/layers/poky/meta/conf/sanity.conf` file.



4 Advanced procedures

This section provides procedures to customize the eSDK. Additionally, it explains the process to upgrade the individual function SDKs.

4.1 Build platform image with QIMP layer

Build the QIMP SDK using either the Qualcomm® Software Center (QSC), or the Yocto build.

Before you begin, set up your infrastructure as described in the Qualcomm Linux Build Guide.

Build using QSC

QSC enables you to download the Qualcomm software distribution and the required host machine dependencies.

QSC provides you with the appropriate software distribution to download the QIMP SDK and compile the software to create the flashable binaries with a single command.

Use any of the following interfaces to build the QIMP SDK through QSC:

- Command-line interface (CLI): Download and compile the QIMP SDK, and generate flashable images using the QSC CLI.
- Graphical user interface (GUI): Use the QSC Launcher GUI to download and compile the QIMP SDK.

Use CLI method

1. Download and compile the QIMP SDK using the instructions available at Build with QSC CLI.
2. During compilation, update the product, release, and distribution tags based on the type of distribution.
   - For product tag, update `--product '<Product_ID>'`.

     For example:

     ```
     --product 'QCM6490.LE.1.0'
     ```

- For distribution tag, update `--distribution '<Distribution>'`.

  For example:

```
--distribution 'Qualcomm_Linux.SPF.1.0|TEST|DEVICE|PB_
QIMPSDK'
```

- For release tag, update `--release '<Release_ID>'`.

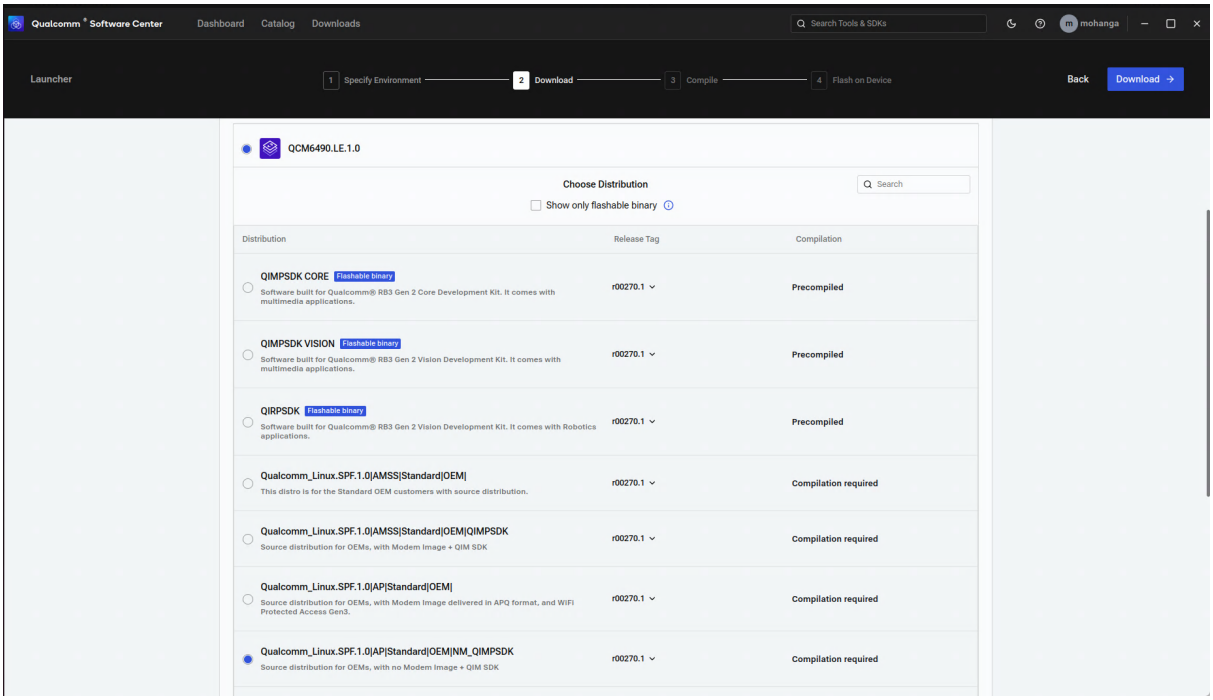
  For example:

```
--release 'r00270.1'
```

Use GUI method

For instructions on the QSC Launcher, see Build with QSC Launcher.

The figure shows a sample selection for a registered user from a verified organization:



The release distribution information is updated based on the user access level.

For example:

- If you are a registered user with any email address, select the distribution as **Qualcomm_Linux.SPF.1.0|TEST|DEVICE|PB_QIMPSDK**.

- If you are a registered user from a verified organization, select the distribution as **Qualcomm_Linux.SPF.1.0|AP|Standard|OEM|NM_QIMPSDK**.

Build using GitHub workflow

To download the QIMP SDK and build using the `meta-qcom-qim-product-sdk` Yocto layer, see GitHub workflow.

4.2 Build a custom eSDK

To enhance the eSDK with custom features and additions or if a compatible prebuilt eSDK isn't available for your platform, build a custom eSDK.

Prerequisites

1. Ensure that the platform image is created. For instructions, see Use CLI method or Build using GitHub workflow.
2. Create an installable eSDK from the platform image.

Generate eSDK

Note: For Arm architecture-based host machines, before generating the eSDK, change the environmental variable for the SDK machine `SDKMACHINE='x86_64'` with `SDKMACHINE="aarch64"`. The variable is at `qcom-6.6.13-QLI.1.0-Ver.1.2_qim-product-sdk\layers\meta-qcom-distro\set_bb_env.sh`. As an example, see the following figure:

```
171
172   if [ -z "${SDKMACHINE}" ]; then
173   |    SDKMACHINE='x86_64'
174   fi
175
176   BUILDDIR="${WS}/build-$DISTRO"
177   DISTRO_VERSION='1.0'
178
```

1. Create the eSDK:

   ```
   MACHINE=<machine> DISTRO=<distro> source setup-environment
   ```

   For example:

   ```
   MACHINE=qcs6490-rb3gen2-vision-kit DISTRO=qcom-wayland source
   setup-environment
   ```

   ```
   bitbake -c do_populate_sdk_ext qcom-multimedia-image
   ```

   For example: `/local/mnt/workspace/qcom-6.6.13-QLI.1.0-Ver.1.2_qim-product-sdk/build-qcom-wayland$ bitbake -c do_populate_sdk_ext qcom-multimedia-image`

   The eSDK is created at `<WORKSPACE_DIR>/build-qcom-wayland/tmp-glibc/deploy/sdk/`. It's a standalone installer that can be shared with you if you're interested in application development without doing a complete BSP build.

2. Install the eSDK:

   ```
   sh qcom-wayland-x86_64-qcom-multimedia-image-armv8-2a-<machine>-
   toolchain-ext-1.0.sh
   ```

For example:

```
sh qcom--wayland-x86_
64-qcom-multimedia-image-armv8-2a-qcs6490-rb3gen2-vision-kit-toolchain-ext-1
0.sh
```

3. When prompted, enter the file system path to install the eSDK.

   For example, `/local/mnt/workspace/Platform_eSDK_plus_QIM`.

Note: For information about `<machine>`, see the Release Notes.

For example, `qcs6490-rb3gen2-vision-kit` is `<machine>` for the RB3 Gen 2 Vision Development Kit.

The host machine is ready for application development. Develop your first application using the eSDK.

4.3 Upgrade QIMP SDK

This section explains the steps to upgrade the QIMP SDK to a new version.

Before you begin, ensure that the QIMP SDK artifacts are generated at `<absolute_workspace_path>/build-qcom-wayland/qim-prod-sdk`. For more information, see Build using GitHub workflow.

Customize and install the QIMP SDK on the device:

1. Sign in to the SSH shell:

   ```
   ssh root@[IP Address of the device]
   ```

   Note: If prompted for a password, enter `oelinux123`.

   ---

2. On the target device, reconfigure the file system partition to support read and write operations:

   ```
   mount -o remount,rw /
   ```

   ```
   exit
   ```

3. Push the `qim-prod-sdk` artifacts on the target device:

```
scp -r qim-prod-sdk_1.0.0.tar.gz root@[IP-ADDR]:/opt
```

4. Extract the tarfile of artifacts inside the target device:

```
cd /opt/
```

```
tar -xvf qim-prod-sdk_1.0.0.tar.gz
```

5. Install the artifacts on the target device:

```
cd qim-prod-sdk
```

```
bash install.sh
```

```
Installing gstreamer1.0-plugins-bad-meta (1.20.7) on root
Configuring gstreamer1.0-plugins-bad-meta.
Installing qcom-gstreamer1.0-plugins-oss-tools-src (1.0) on root
Configuring qcom-gstreamer1.0-plugins-oss-tools-src.
Installing gstreamer1.0-plugins-bad-asfmux (1.20.7) on root
Configuring gstreamer1.0-plugins-bad-asfmux.
Installing gstreamer1.0-locale-hu (1.20.7) on root
Configuring gstreamer1.0-locale-hu.
Installing gstreamer1.0-plugins-bad-proxy (1.20.7) on root
Configuring gstreamer1.0-plugins-bad-proxy.
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>> Installation done for QIM_PRODUCT_SDK at root
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

sh-5.1#
```

4.4 Upgrade other SDKs

The function SDKs are integrated into the QIMP SDK as individual layers that fetch the source or proprietary binaries. Though it's recommended to use the versions that are integrated with a specific QIMP SDK, you can update the individual function SDKs if necessary.

The SDK versions are configured in the `[qcom-ml.inc]` file.

Upgrade Qualcomm Neural Processing SDK or Qualcomm AI Engine direct SDK

To upgrade the SDK, update the required version of SDK in the **Version** field.

To find the version string:

- For Qualcomm Neural Processing SDK, go to QSC > **Qualcomm Neural Processing SDK** and select the required version from the **Version** drop-down list.
- For Qualcomm AI Engine direct SDK, go to QSC > **Qualcomm AI Engine direct SDK** and select the required version from the **Version** drop-down list.

5 References

5.1 Related documents

Title	Document number
Qualcomm Technologies, Inc.	
Qualcomm Intelligent Multimedia Software Development Kit (IM SDK) Reference	80-70018-50
RB3 Gen 2 Quick Start Guide	80-70018-253
Qualcomm Linux Build Guide	80-70018-254
Qualcomm Linux Virtual Machine Setup Guide	80-70018-41
Qualcomm Linux Landing Page	80-70018-115
Qualcomm RB3 Gen 2 Development Kit Guide	80-70018-251
Resources	
https://gstreamer.freedesktop.org/documentation/application-development/introduction/gstreamer.html	
https://wiki.yoctoproject.org/wiki/Extensible_SDK	
https://docs.yoctoproject.org/ref-manual/devtool-reference.html	
https://wiki.yoctoproject.org/wiki/Extensible_SDK#Introduction	
https://git.yoctoproject.org/opkg/tree/README.md	
https://github.com/quic/sample-apps-for-qualcomm-linux	
https://docs.yoctoproject.org/ref-manual/devtool-reference.html#getting-help	
https://github.com/quic-yocto/meta-qcom-qim-product-sdk/blob/kirkstone/recipes-qcom-ml/qcom-ml.inc	
https://softwarecenter.qualcomm.com/#/	

5.2 Acronyms and terms

Acronym	Definition
CLI	Command-line interface
eSDK	Extensible SDK
GUI	Graphical user interface
IM SDK	Qualcomm Intelligent Multimedia SDK
IMU	Inertial measurement units
IoT	Internet of Things
MIPI	Mobile industry processor interface
QIMP	Qualcomm Intelligent Multimedia Product
QNN	Qualcomm Neural Network
QSC	Qualcomm Software Center
SDK	Software development kit
SNPE	Snapdragon Neural Processing Engine

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this "Material"), is subject to your (including the corporation or other legal entity you represent, collectively "You" or "Your") acceptance of the terms and conditions ("Terms of Use") set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. ("**Qualcomm Technologies**"), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as "**Qualcomm Internal Use Only**", no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies' prior written approval, nor may it be used for any machine learning or artificial intelligence development purpose which results, whether directly or indirectly, in the creation or development of an automated device, program, tool, algorithm, process, methodology, product and/or other output. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com, the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, or other legal statements or notices found on prior pages of the Material, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.