i.MX 6 Series Yocto Project Multimedia Gstreamer 1.x

User’s Guide

Rev. 4.0.0, 04.2014

By Freescale Semiconductor. Inc

Table of Contents

[Chapter 1 About This Book 3](#_Toc385424658)

[1.1 Audience 3](#_Toc385424659)

[1.2 Conventions 3](#_Toc385424660)

[1.3 References 3](#_Toc385424661)

[Chapter 2 Build Multimedia packages 4](#_Toc385424662)

[2.1 Build Yocto Project 4](#_Toc385424663)

[2.2 Build Freescale Multimedia components 4](#_Toc385424664)

[2.2.1 Freescale Muitimedia packages 4](#_Toc385424665)

[2.2.2 Build Standard packages 4](#_Toc385424666)

[2.2.3 Build Special and Excluded packages 4](#_Toc385424667)

[Chapter 3 Multimedia User Cases 5](#_Toc385424668)

[3.1 Playback 5](#_Toc385424669)

[3.1.1 Audio only Playback 5](#_Toc385424670)

[3.1.2 Video only Playback 5](#_Toc385424671)

[3.1.3 Audio/Video file Playback 5](#_Toc385424672)

[3.1.4 Other methods for playback 5](#_Toc385424673)

[3.2 Camera 6](#_Toc385424674)

[3.2.1 Camera preview 6](#_Toc385424675)

[3.3 HTTP streaming 6](#_Toc385424676)

[3.4 RTSP Streaming Playback 6](#_Toc385424677)

[3.5 RTP/UDP MPEGTS streaming 8](#_Toc385424678)

[3.6 RTSP Streaming Server 9](#_Toc385424679)

[Appendix 1 : Install gstreamer1.0-libav into rootfs image 9](#_Toc385424680)

[Appendix 2 : pulseaudio input/output setting 10](#_Toc385424681)

Chapter 1 About This Book

This document describes how to build Freescale Multimedia components (Gstreamer plugins, decoder/encoder, demuxer) with Yocto Project, and provides how to run various multimedia use cases with Gstreamer command lines. Customers can refer to these command lines to create their multimedia products.

## Audience

This document is intended for software, hardware, and system engineers who are planning to use multimedia codecs with GStreamer architecture and for anyone who wants to understand more about multimedia codecs. The document assumes that the user has a basic understanding of GStreamer and Linux architecture.

## Conventions

This document uses the following conversions:

* Courier New font: This font is used to identify commands, explicit command parameters, code examples, expressions, data types, and directives.
* $ Sign: It is used to specify replaceable command parameters.

## References

The following documents were referenced to build this document:

* i.MX 6Dual/6Quad/6SoloLite/6SX SABRE-SD/SABRE-AUTO Linux User’s Guide.
* Freescale Yocto Project User Guide.
* i.MX6 Multimedia Gstreamer1.x Yocto Project Release Notes.

# Chapter 2 Build Multimedia packages

This chapter describes how to setup the Yocto Project build environment and how to build multimedia packages into Yocto Project image.

## 2.1 Build Yocto Project

Please refer to the “Freescale Yocto Project User Guide” for how to setup Yocto Project environment and how to build the Yocto Project image.

## 2.2 Build Freescale Multimedia components

### 2.2.1 Freescale Muitimedia packages

Due to the license limitation, Freescale Multimedia packages consist of three parts:

* Standard package: no license limitation packages
* Special package: license limitation packages
* Excluded package: license limitation package

For each package details, please refer to Yocto Project Multimedia Release Notes.

### 2.2.2 Build Standard packages

Standard multimedia packages are default built into Yocto Project image.

If you want to update certain package and build it, you can put it under downloads directory and do below steps:

$ bitbake -c cleanall $packagename

$ bitbake $packagename

The package name should be identical to the recipe name (under sources/meta-fsl-arm/recipes-multimedia/$component/$packagename\_$version.bb)

For example:

$ bitbake gst1.0-fsl-plugin

### 2.2.3 Build Special and Excluded packages

Please place the special/excluded packages into downloads directory and refer to the readme in each package.

For example, README-microsoft in the package libflmscodec-$version.bin

# Chapter 3 Multimedia User Cases

## 3.1 Playback

### 3.1.1 Audio only Playback

gst-launch-1.0 filesrc location=$clip\_name [typefind=true] ! $audio\_parser\_plugins ! $audio\_decoder\_plugin ! $audio\_sink\_plugin

MP3 playback example:

gst-launch-1.0 filesrc location=test.mp3 [typefind=true] ! mpegaudioparse ! beepdec ! pulsesink

### 3.1.2 Video only Playback

gst-launch-1.0 filesrc location=test.video typefind=true ! $capsfilter ! $demuxer\_plugin ! queue max-size-time=0 ! $video\_decoder\_plugin ! $video\_sink\_plugin

MP4 file video only playback example:

gst-launch-1.0 filesrc location= H264\_BP11\_352x240\_30\_1248\_AAC\_48\_192\_2\_friendsr.mp4 typefind=true ! video/quicktime ! aiurdemux ! queue max-size-time=0 ! vpudec ! imxv4l2sink

### 3.1.3 Audio/Video file Playback

gst-launch-1.0 filesrc location=test\_file typefind=true ! $capsfilter ! $demuxer\_plugin name=demux demux. ! queue max-size-buffers=0 max-size-time=0 ! $video\_decoder\_plugin ! $video\_sink\_plugin demux. ! queue max-size-buffers=0 max-size-time=0 ! $audio\_decoder\_plugin ! $audio\_sink\_plugin

AVI file playback example:

gst-launch-1.0 filesrc location=test.avi typefind=true ! aiurdemux name=demux demux. ! queue max-size-buffers=0 max-size-time=0 ! vpudec ! imxv4l2sink demux. ! queue max-size-buffers=0 max-size-time=0 ! beepdec ! pulsesink

**Note**

For the plattforms without VPU hardware (6SL/6SX),   
$video\_deocder\_plugin could be software decoder plugin like ffdec\_h264.

### 3.1.4 Other methods for playback

You can use playbin plugin or FSL gplay-1.0 command line player for media file playback.

gst-launch-1.0 playbin uri=file:///mnt/sdcard/test.avi

gplay-1.0 /mnt/sdcard/test.avi

## 3.2 Camera

### 3.2.1 Camera preview

gst-launch-1.0 imxv4l2src ! 'video/x-raw,format=(string)$FORMAT,width=$WIDTH,height=$HEIGHT,framerate=(fraction)30/1' ! imxv4l2sink

Camera preview example:

gst-launch-1.0 imxv4l2src device=/dev/video1 ! 'video/x-raw,format=(string)UYVY,width=640,height=480,framerate=(fraction)30/1' ! imxv4l2sink

**Note**

* Get the camera support format and resolution using gst-inspect-1.0 imxv4l2src.
* Set caps filter according camera supported capabilities if user need other format or resolution.
* Ensure set right caps filter which also need supported by imxv4l2sink.

## 3.3 HTTP streaming

* manually pipeline

gst-launch-1.0 souphttpsrc location= <http://SERVER/test.avi> ! typefind ! aiurdemux name=demux demux. ! queue max-size-buffers=0 max-size-time=0 ! vpudec ! imxv4l2sink demux. ! queue max-size-buffers=0 max-size-time=0 ! beepdec ! $audiosink\_plugin

* playbin

gst-launch-1.0 playbin uri=http://SERVER/test.avi

* gplay-1.0

gplay-1.0 http://SERVER/test.avi

## 3.4 RTSP Streaming Playback

For h264 high bit rate playback, need to set access-unit property to true to let the depay plugin output h264 in complete frames to avoid performance downgrade in vpu decoder plugin in low latency mode.

* manually pipeline

gst-launch-1.0 rtspsrc location=$RTSP\_URI name=source ! queue ! $video\_rtp\_depacketize\_plugin ! vpudec ! imxv4l2sink source. ! queue ! $audio\_rtp\_depacketize\_plugin ! $audio\_parse\_plugin ! beepdec ! $audiosink\_plugin

For example (h264 + aac):

gst-launch-1.0 rtspsrc location=rtsp://10.192.241.11:8554/test name=source ! queue ! rtph264depay ! vpudec ! imxv4l2sink source. ! queue ! rtpmp4gdepay ! aacparse ! beepdec ! pulsesink

Audio parse plugin is required before beepdec plugin to let beepdec working in low latency mode.

You can input below command to show the gstreamer rtp depacketize plugins,

gst-inspect-1.0 | grep depay

Two properties of rtspsrc are useful for rtsp streaming:

**latency:** This is the extra added latency of the pipeline, default value is 200ms, if you need low latency rtsp streaming playback, you can set this property with smaller value.

**buffer-mode:** This property is used to control the buffering algorithm in use, 4 modes are provided:

* none: Outgoing timestamps are calculated directly from the RTP timestamps, not good for real-time applications.
* slave: Calculate the skew between sender and receiver and produce smoothed adjusted outgoing timestamps, good for low latency communications.
* buffer: Buffer packets between low/high watermarks, good for streaming communication.
* auto: Choose above 3 modes depending on the stream

Default setting is auto.

* playbin

Need to set vpudec low-latency to true if play with playbin.

gst-launch-1.0 playbin uri=$RTSP\_URI

**Note**: If you need to pause/resume rtsp streaming playback, you need to use slave/none buffer-mode for rtspsrc, as in buffer buffer-mode, after resume, the timestamp is forced to start from 0, this will cause buffers are dropped after resume.

The playback will not exit automatically if buffer-mode of rtspsrc is ‘buffer’ .

## 3.5 RTP/UDP MPEGTS streaming

* UDP MPEGTS Streaming commands

gst-launch-1.0 udpsrc do-timestamp=false uri=$UDP\_URI caps="video/mpegts" ! aiurdemux streaming\_latency=400 name=d d. ! queue ! vpudec ! queue ! imxv4l2sink sync=true d. ! queue ! beepdec ! $audiosink\_plugin sync=true

For example:

gst-launch-1.0 udpsrc do-timestamp=false uri=udp://10.192.241.255:10000 caps="video/mpegts" ! aiurdemux streaming\_latency=400 name=d d. ! queue ! vpudec ! queue ! imxv4l2sink sync=true d. ! queue ! beepdec ! pulsesink sync=true

* RTP MPEGTS Streaming commands

gst-launch-1.0 udpsrc do-timestamp=false uri=$RTP\_URI caps="application/x-rtp" ! rtpmp2tdepay ! aiurdemux streaming\_latency=400 name=d d. ! queue ! vpudec ! queue ! imxv4l2sink sync=true d. ! queue ! beepdec ! $audiosink\_plugin sync=true

For example:

gst-launch-1.0 udpsrc do-timestamp=false uri=udp://10.192.241.255:10000 caps="application/x-rtp" ! rtpmp2tdepay ! aiurdemux streaming\_latency=400 name=d d. ! queue ! vpudec ! queue ! imxv4l2sink sync=true d. ! queue ! beepdec ! pulsesink sync=true

* Note:

Source file that udp/rtp server sent must be in ts format.

Recommends that server starts one second earlier than the time client start.

One property of aiurdemux is useful for udp/rtp ts streaming:

**streaming-latency:** This is the extra added latency of the pipeline, default value is 400ms. This value is designed for situation that client starts first, if the value is too small, the whole pipeline may not run since lack of audio or video buffers, in that case, you should cancel current command and restart the pipeline; if the value is too large, you will wait a long time to see the video after starting the server.

## 3.6 RTSP Streaming Server

The RTSP streaming server use case is based on open source gst-rtsp-server package, it uses Freescale aiurdemux plugin to demux the file to audio/video elementary streams and send them out via RTP. You can start RTSP streaming server in one board, play it in another board with RTSP streaming playback commands.

gst-rtsp-server package is not installed default in the Yocto Project release, you can follow below steps to build and install it.

1. Enable the layer – meta-openembedded/meta-multimedia:

Adding the line BBLAYERS += " ${BSPDIR}/sources/meta-openembedded/meta-multimedia" to the configure file <yocto\_project\_root>/build/conf/bblayers.conf .

1. Include gst-rtsp-server into the image build:

Adding the line IMAGE\_INSTALL\_append += " gst-rtsp-server" to the configure file <yocto\_project\_root>/build/conf/local.conf

1. Run the command bitbake fsl-image-test/fsl-image-gui/fsl-image-x11/test-internal-x11 to build the image with gst-rstp-server
2. After that, you can find test-uri binary in the folder <yocto\_project\_root>/build/tmp/work/cortexa9hf-vfp-neon-poky-linux-gnueabi/gst-rtsp-server/1.2.3-r0/gst-rtsp-server-1.2.3/ examples/.libs/
3. Flash the image.

Copy test-uri into /usr/bin on board and make it have executable permission.

* commands

test-url $RTSP\_URI

For example:

test-uri [file:///home/root/temp/TestSource/mp4/1.mp4](file:///D:\home\root\temp\TestSource\mp4\1.mp4)

* server address:

rtsp://$SERVER\_IP:8554/test

For example:

rtsp://10.192.241.106:8554/test

* client operation supported:

Play, Stop, Pause/Resume, Seek

# Appendix 1 : Install gstreamer1.0-libav into rootfs image

Step1: Add the lines followd in the configuration file conf/local.conf

IMAGE\_INSTALL\_append = “gstreamer1.0-libav”

LICENSE\_FLAGS\_WHITELIST = “commercial”

Step2: Build gstreamer1.0-libav

$ bitbake gstreamer1.0-libav

Step3: Build rootfs image

$ bitbake <image\_name>

# Appendix 2 : pulseaudio input/output setting

If the pulseaudio is installed in the rootfs, the pulseaudio input /out setting may need set.

Note: Pulse audio only available for X11back-end Yocto Project rootfs.

#### Audio output Setting

Use ‘pactl’ command to list all available audio sinks:

$ pactl list sinks

A list of available audio sinks will be displayed:

**Sink #0**

**State: SUSPENDED**

**Name: alsa\_output.platform-soc-audio.1.analog-stereo**

**Description: sgtl5000-audio Analog Stereo  
 ...**

**...**

**Sink #1**

**State: SUSPENDED**

**Name: alsa\_output.platform-soc-audio.4.analog-stereo**

**Description: imx-hdmi-soc Analog Stereo**

**...**

**...**

Use ‘pacmd’ command to set the default audio sink accordingly as the sink number in list showed above:

$ pacmd set-default-sink $sink-number

e.g. $sink-number could be 0 or 1 in above list

After setting the default sink, use below command to verify the audio path:

$ gst-launch-1.0 audiotestsrc ! plusesink

#### Audio input Setting

Use ‘pactl’ command to list all available audio sources:

$ pactl list sources

A list of available audio sources will be displayed:

**Source #0**

**State: SUSPENDED**

**Name: alsa\_output.platform-soc-audio.1.analog-stereo.monitor**

**Description: Monitor of sgtl5000-audio Analog Stereo**

**...**

**...**

**Source #1**

**State: SUSPENDED**

**Name: alsa\_input.platform-soc-audio.1.analog-stereo**

**Description: sgtl5000-audio Analog Stereo ...**

**...**

**...**

Use ‘pacmd’ command to set the default audio source accordingly as the source number in list showed above:

$ pacmd set-default-source $sink-number

e.g. $sink-number could be 0 or 1 in above list

Note: if not need record and playback at the same time, there is no need to set to monitor mode.

The pulseaudio I/O path setting status can be checked by

$ pactl stat