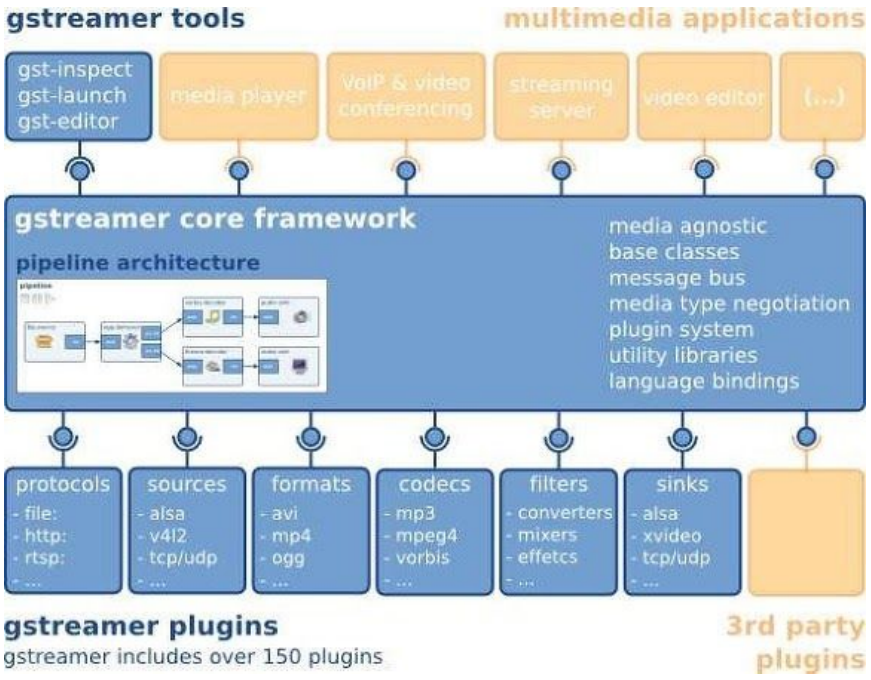
Gstreamer

https://gstreamer.freedesktop.org/documentation

파이프라인 기반의 영상처리 오픈소스.

핵심기능 : plugin, data flow, media type handling 및 negotiation을 위한 프레임워크 제공



하나의 파이프라인 구조에서 각기 다른 기능을 가진 Element들을 연결하여 결과물을 생성(src → sink)

```
 $ gst-launch-1.0 rtspsrc
location=rtsp://192.168.0.11/profile2/media.smp
! capsfilter caps="application/x-rtp,media=video" ! decodebin
! clockoverlay time-format="%D %H:%M:%S"
! x264enc ! mpegtsmux ! tee name = t
\t. ! queue ! hlssink location="/home/imr/media/file%03d.ts"
\t. ! queue ! hlssink location="/home/imr/nms/test%04d.ts"
```

Gst-launch-1.0 : Gstreamer상에서 파이프라인을 실행시켜주는 도구.

파이프라인을 parsing하여 영상처리

설치 :
리눅스상에서 Gstreamer 설치유무 확인:
$ which gst-launch-1.0
설치 경로가 출력됐을 경우, 설치 필요 無
ex) /usr/bin/gst-launch-1.0

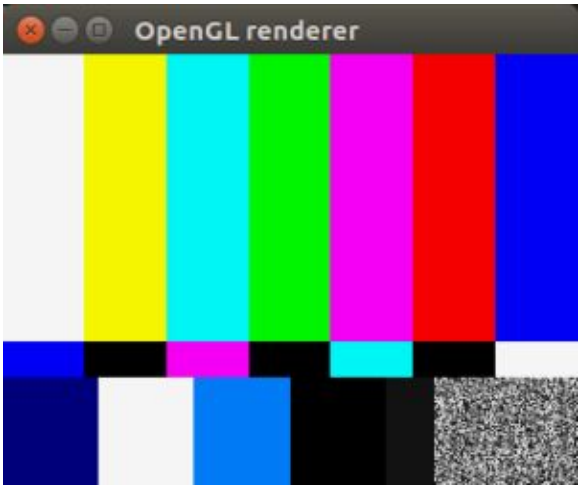
필요한 패키지 설치:
$ sudo apt-get install libgstreamer1.0-0 sudo apt-get install gstreamer1.0-plugins-base gstreamer1.0-plugins-good sudo apt-get install gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly sudo apt-get install gstreamer1.0-libav gstreamer1.0-doc gstreamer1.0-tools

에러 발생시 한줄씩 설치:
$ sudo apt-get install libgstreamer1.0-0 sudo apt-get install gstreamer1.0-plugins-base gstreamer1.0-plugins-good
$ sudo apt-get install gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly
$ sudo apt-get install gstreamer1.0-libav gstreamer1.0-doc gstreamer1.0-tools

파이썬 GStreamer 라이브러리 설치:
$sudo apt-get install python-gst-1.0 python3-gst-1.0

dev-packages 설치 :
$sudo apt-get install libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev sudo apt-get install libfontconfig1-dev libfreetype6-dev libpng-dev sudo apt-get install libcairo2-dev libjpeg-dev libgif-dev sudo apt-get install libgstreamer-plugins-base1.0-dev

기본 command line:
$ gst-launch-1.0 videotestsrc ! autovideosink



해당 화면 출력시 설치 및 실행 성공
※실행전 SSH -X 옵션 및 Xming(디스플레이 서버) 설치 必

Element : GstElement class 에서 파생된 object. 소스와 싱크는 패드를 통해 연결

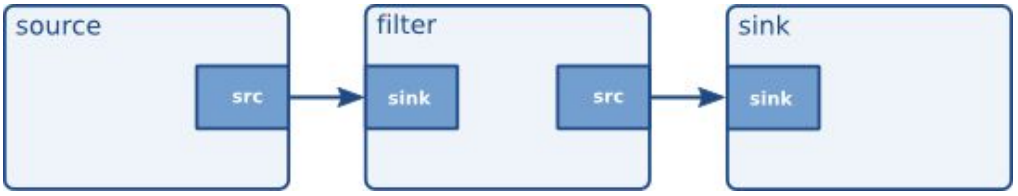
Element는 서로간 연결될때 특정 기능을 제공

- ex) source element는 stream에 data를 제공하고, filter element는 stream상의 data에 따라 행동한다.

Element는 plugin으로 감싸져야 Gstreamer 에서 사용

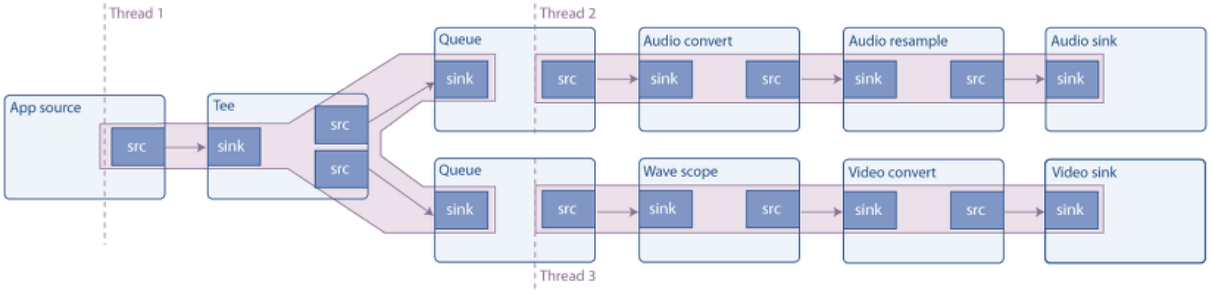
한 개의 plugin은 한 개 또는 다수 element 구현

데이터의 생산자(source or src)와 소비자(sink element) Elements



주요 Elements (RTSP streaming 기능 위주) :

- rtspsrc : RTSP로 생성된 영상 소스를 불러오는 Element
- filter : 현재 pad에 동작하는 미디어에 대해 제한을 걸기 위한 용도로 cap을 사용
  - ex) capsfilter caps="application/x-rtp,media=video"
- decodebin : 자동으로 알맞은 decoder와 demuxer기능을 구성
- encoder : 포맷에 맞는 결과물을 처리하기 위해 소스를 인코딩해주는 기술
  - ex) x264enc : H.264 코덱으로 인코딩
- hlssink : HLS형식으로 sink 생성.
  - 이외에도 다양한 sink elements와 mux를 사용하여 원하는 포맷으로 저장 가능
  - ex) filesink, appsink, autovideosink, ximagesink...etc
- autovideosink : Gstreamer상에서 내장된 시스템의 기본적인 sink를 자동 선택
- tee : 다양한 결과물을 처리하고자 할때 파이프라인 병렬처리 시켜주는 Element
- queue : 순서에 맞게 Element 간의 쓰레드를 분리.



Bus - application과 pipelin 사이의 데이터 교환 또는 통신을 위한 메커니즘
메시지를 생성하여 event를 확인하고 결과값을 파이프라인으로 전송.
메시지 핸들러 함수를 통해 element간 주고받는 메시지를 확인하고 처리
ex) ERROR, WARNING, EOS, STREAM_START...etc
State에 따라 파이프라인 작동 유무 판단.
ex) PLAYING, PAUSED, NULL...etc

GST_STATE_NULL:

this is the default state. This state will deallocate all resources held by the element.

GST_STATE_READY:

in the ready state, an element has allocated all of its global resources, that is, resources that can be kept within streams. You can think about opening devices, allocating buffers and so on. However, the stream is not opened in this state, so the stream positions is automatically zero. If a stream was previously opened, it should be closed in this state, and position, properties and such should be reset.

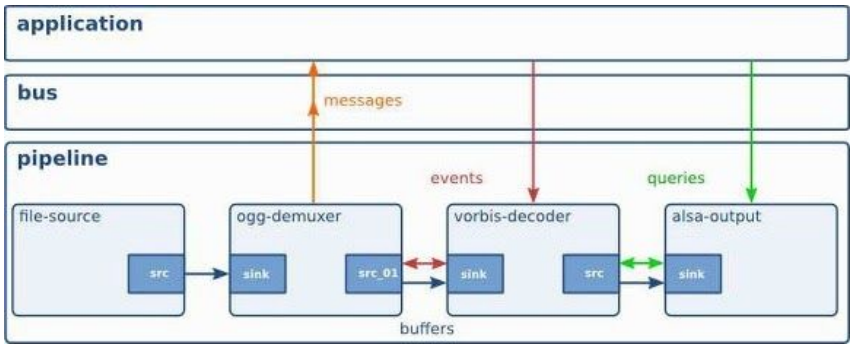
GST_STATE_PAUSED:

in this state, an element has opened the stream, but is not actively processing it. An element is allowed to modify a stream's position, read and process data and such to prepare for playback as soon as state is changed to LAYING, but it is not allowed to play the data which would make the clock run. In summary, PAUSED is the same as PLAYING but without a running clock. Elements going into the PAUSED state should prepare themselves for moving over to the PLAYING state as soon as possible. Video or audio outputs would, for example, wait for data to arrive and queue it so they can play it right after the state change. Also, video sinks can already play the first frame (since this does not affect the clock yet). Autopluggers could use this same state transition to already plug together a pipeline. Most other elements, such as codecs or filters, do not need to explicitly do anything in this state, however.

GST_STATE_PLAYING:

in the PLAYING state, an element does exactly the same as in the PAUSED state, except that the clock now runs.

출처 :
https://gstreamer.freedesktop.org/documentation/application-development/basics/elements.html?gi-language=c



메시지를 통해 실시간으로 파이프라인의 작업을 진행하고 영상물 처리.
필요한 메시지만 filtering하여 영상 처리 가능

프로그래밍 기본 구성(Python) :

1. gi.require_version('Gst', '1.0')
2. from gi.repository import GObject, Gst #파이썬 상에서 Gstreamer 프레임워크 적용
3.
4. Gst.init(None) #초기화
5.
6. pipeline = Gst.parse_launch(“””파이프라인 구성 요소 추가“””)
7. element = pipeline.get_by_name(“element name”)
8. element.set_property(‘option’, ‘value’)
9.
10. pipeline.set_state(Gst.State.PLAYING) #Gstreamer 기능 수행
11. bus = pipeline.get_bus() #파이프라인 element간 메시지 공유
12. pipeline.set_state(Gst.State.NULL) #작업완료, EOS, 또는 에러 발생시 기능종료

Http Live Streaming

스트리밍 : 네트워크 기반 비디오, 오디오 등의 멀티미디어 정보를 제공하는 기술로 다운로드없이 실시간으로 재생가능.
재생 시간이 단축되며 HDD 용량에도 영향을 받지 않는다.

HLS : m3u8의 확장자를 가진 재생목록 파일과 다수의 ts 영상을 HTTP를 통해 전송하는 방식

- m3u8 : UTF-8으로 인코딩된 m3u파일
- m3u : ts 파일의 재생목록을 관리하는 파일
- ts : MPEG-4 형식의 분할 저장 파일

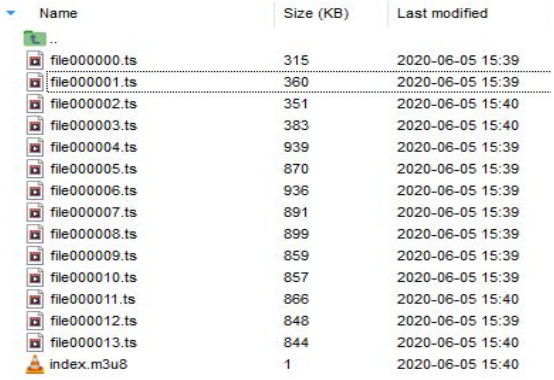
기본적으로 실시간 스트리밍 기능을 제공하지만 options를 통해 저장 기능 구현 가능
ex)

- playlist-length : 1~∞의 ts 재생목록에 기록 설정가능
- max-files : 최대 ts 저장 설정. 설정한 값이 넘어가면 FIFO순으로 파일 삭제

m3u8 재생목록과 ts파일 분할 예시 :

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-ALLOW-CACHE:NO
#EXT-X-MEDIA-SEQUENCE:1
#EXT-X-TARGETDURATION:3

#EXTINF:3.4995403289794922,
file000000.ts
#EXTINF:3.3863615989685059,
file000001.ts
#EXTINF:3.3931219577789307,
file000002.ts
```



Name	Size (KB)	Last modified
..		
file000000.ts	315	2020-06-05 15:39
file000001.ts	360	2020-06-05 15:39
file000002.ts	351	2020-06-05 15:40
file000003.ts	383	2020-06-05 15:40
file000004.ts	939	2020-06-05 15:39
file000005.ts	870	2020-06-05 15:39
file000006.ts	936	2020-06-05 15:39
file000007.ts	891	2020-06-05 15:39
file000008.ts	899	2020-06-05 15:39
file000009.ts	859	2020-06-05 15:39
file000010.ts	857	2020-06-05 15:39
file000011.ts	866	2020-06-05 15:40
file000012.ts	848	2020-06-05 15:39
file000013.ts	844	2020-06-05 15:40
index.m3u8	1	2020-06-05 15:40

Options를 사용하여 24시간 녹화 실행시 화면

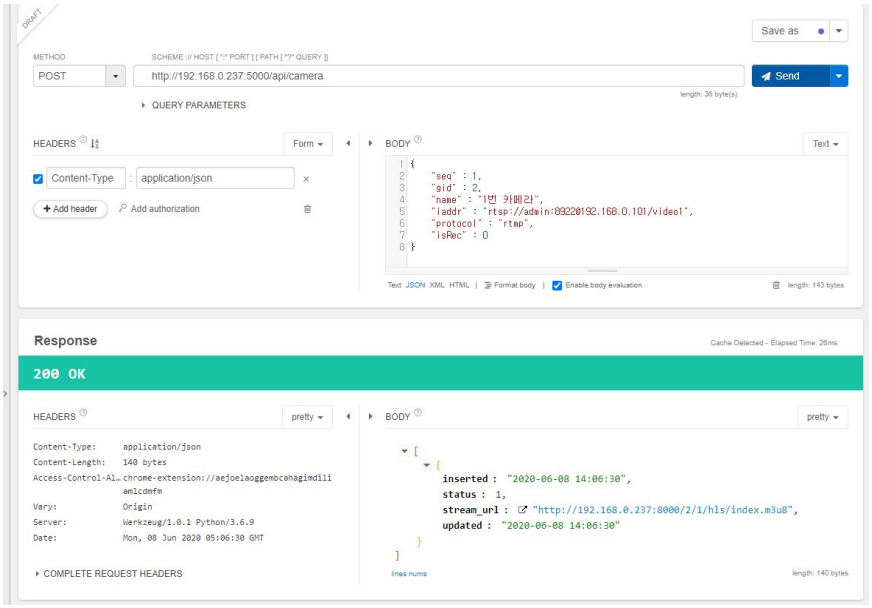
API(Application programming interface)
파이썬 상에서 만든 기능을 프론트에서 구현 할 수 있도록 JSON을 통해 필요한 정보를 주고 받는 인터페이스.
Flask를 사용하여 REST API형식으로 필요한 데이터를 요청하거나 받아서 기능을 실행.

API 설정 예시:
* 카메라 API 목록: 사용법 http://IP:PORT/api/camera
# 노드.JS To 파이썬
카메라 등록 및 실시간 스트림 시작
설명 : 개별 구동
주소 : /api/camera
메소드 : POST
송신 데이터 :

```
{
        "seq" : 1,
        "gid" : 2,
        "name" : "1번 카메라",
        "iaddr" : "rtsp://admin:8922@192.168.0.101/video1",
        "protocol" : "rtmp",
        "isRec" : 0
}
```

응답 데이터 :

```
{
  "status" : 1 or 0,
  "isRec" : 1 or 0,
  "inserted" : "2020-02-05 11:22:33",
  "updated" : "2020-02-05 11:22:33",
  "stream_url" : "http://192.168.0.237:8000/{gid}/{seq}/hls/index.m3u8"
}
```



API 테스트 실행 후 전송 성공시 Return 값 출력

```
imr@mainserver:~/nms/Node-Media-Server$ python3 test.py
 * Serving Flask app "test" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
Rec status : 0
[{'status': 1, 'inserted': '2020-06-08 14:06:30', 'updated': '2020-06-08 14:06:30', 'stream_url': 'http://192.168.
0.237:8000/2/1/hls/index.m3u8'}]
Streaming Registered on 2020-06-08 14:06:30
192.168.0.47 - - [08/Jun/2020 14:06:30] "POST /api/camera HTTP/1.1" 200 -
Bus message gets
Pipeline state changed from NULL to READY
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efda827ed80)>
Pipeline state changed from READY to PAUSED
<flags GST_MESSAGE_NEW_CLOCK of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x7efda827f810)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efda827fa90)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efda827ff10)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x12dd090)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x12dd110)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x12dd510)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x131ed40)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x7efda827fa10)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efda826bd00)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022ee8 (GstMessage at 0x12fcda0)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x12fcc20)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022ee8 (GstMessage at 0x131ee40)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efd90002cc0)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x131eb40)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022ee8 (GstMessage at 0x131ecc0)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x12fcf20)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x7efd84002b80)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022ee8 (GstMessage at 0x7efda827fa90)>
<flags GST_MESSAGE_PROGRESS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efd84002d00)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022ee8 (GstMessage at 0x13339b0)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x7efd74002830)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efd74002e30)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efd7401f9b0)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x7efd7401fb30)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022768 (GstMessage at 0x7efda827ff10)>
<flags GST_MESSAGE_STREAM_STATUS of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x131ed40)>
<flags GST_MESSAGE_LATENCY of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efda827eb80)>
<flags GST_MESSAGE_ELEMENT of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x7efd74002f30)>
<flags GST_MESSAGE_ELEMENT of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022e88 (GstMessage at 0x1333ab0)>
Stream Start
2020-06-08 14:06:30
<flags GST_MESSAGE_ASYNC_DONE of type Gst.MessageType>
Unknown message: <Gst.Message object at 0x7efdb0022ee8 (GstMessage at 0x7efd541522b0)>
```

API 테스트 전송 성공 후 GStreamer 기능 실행
(192.168.0.47 - - [08/Jun/2020 14:06:30] "POST /api/camera HTTP/1.1" 200 -)