HDDL-S user guide

# 1. How to run default use cases

## 1.1 Install HDDL-S software

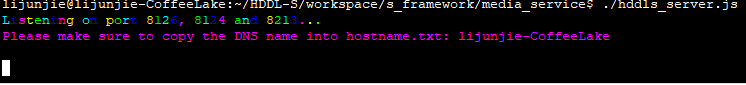
Detail see document of HDDL-S software installation guide.

## 1.2 Use MO toolset to convert caffe/tensorflow/other trained models into IR files

Detail see MO user guide document

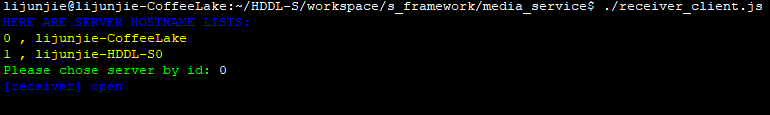
## 1.3 Start HDDL-S web server in HDDL-S module machines

#./hddls\_server.js

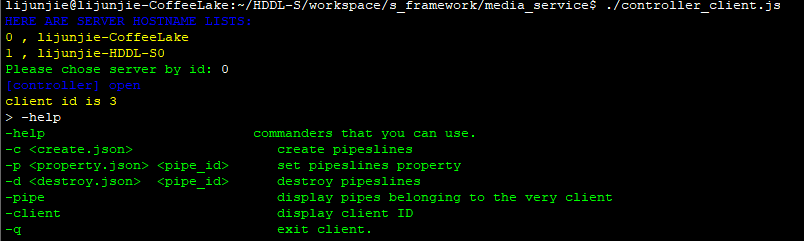


## 1.4 Start receiver client in host machine to receive the processing result

#./receiver\_client.js



## 1.5 Start a controller client in host machine

#./controller\_client.js 

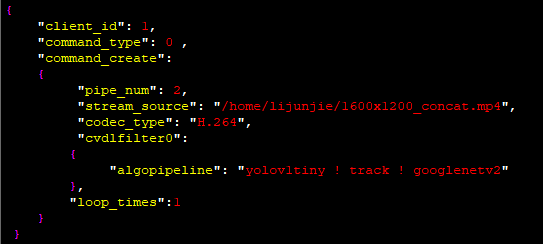
### 1.5.1 create hddl pipeline

#### 1) Edit create.json file to configure the hddlpipe parameters

The json file format is:

|  |
| --- |
| {      "client\_id":<int>,     // every client has an identity id, which is used for web server to distinguish the command is from which client      "command\_type": <int> ,  //  3 types: create or destroy or set property, choose one of the 3 for each command      "command\_create": // header of create pipe command      {           "pipe\_num": <int>,          // how many pipes to be created           "stream\_source": "<string>",  // input stream source uri           "codec\_type": "<string>",     //stream codec type: H.264 or H.265           "cvdlfilter0":  // cvdlfilter name, sub-item are its properties list          {               "algopipeline”: "<string>" // a property to set different algo chain for different task: detection, track, classification/recognition, etc          }      },     "command\_destroy":   // header of destroy pipe command     {          "pipe\_id":<int>,   // which pipeline will be destroy     },     "command\_set\_property": // header of set property command     {          "pipe\_id":<int>,   // which pipeline will be set property          "cvdlfilter0":     // filter name          {              <property>: <string>,  // property name and property value          }     }  } |

An example is below:



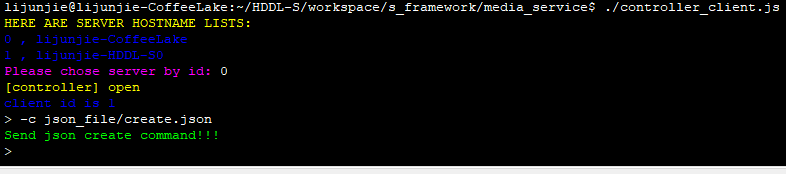
By default, it supports 3 use cases: road barrier, cross road and indoor person count, we can specified any use case for one hddl pipeline by set “algopipeline” property:

Road barrier: *algopipeline = "yolov1tiny ! track ! googlenetv2"*

Cross road: *algopipeline = “mobilenetssd ! tracklp ! lprecognize”*

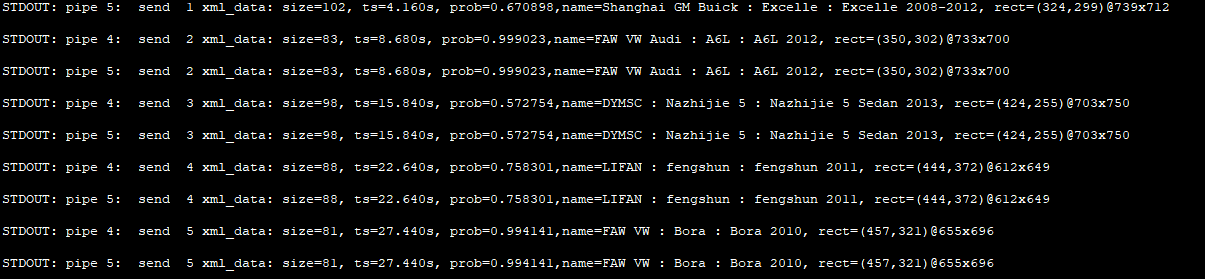
Indoor person count: *algopipeline = “yolov2tiny ! reid”*

2) Send this json format command to hddl web server:

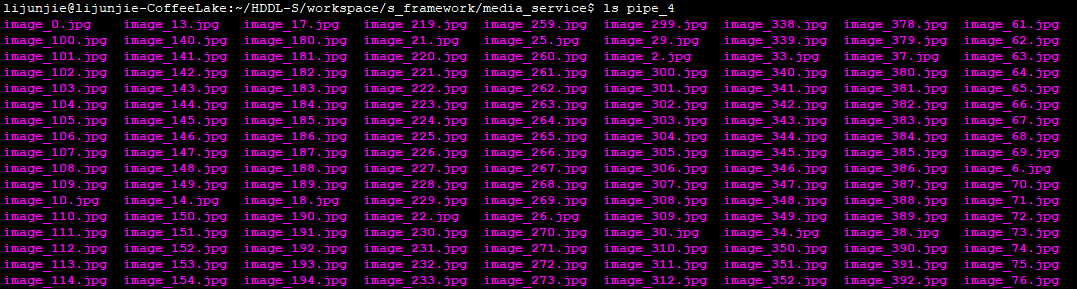


3) HDDL-S web server will receive this command and create hddl pipeline as command describes, and then receiver client will begin to receive pipeline processing result.

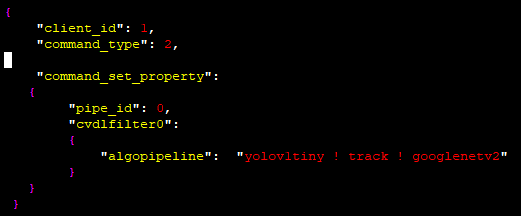
We can see some logs in web server(2 pipeline are running, pipe 4 and 5):



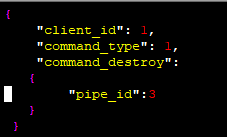
And get result in receiver client (for this example, I put receiver client in media\_server directory, and below show result of pipe 4):



### 1.5.2 Set properties for any gstreamer plugins



### 1.5.3 Stop a specified pipeline



# 2. How to deploy new customer models

## 2.1 Use MO toolset to convert caffe/tensorflow/other trained models into IR files

Detail see MO user guide document

## 2.2 Implement 5 APIs to parse inference result and set input parameters

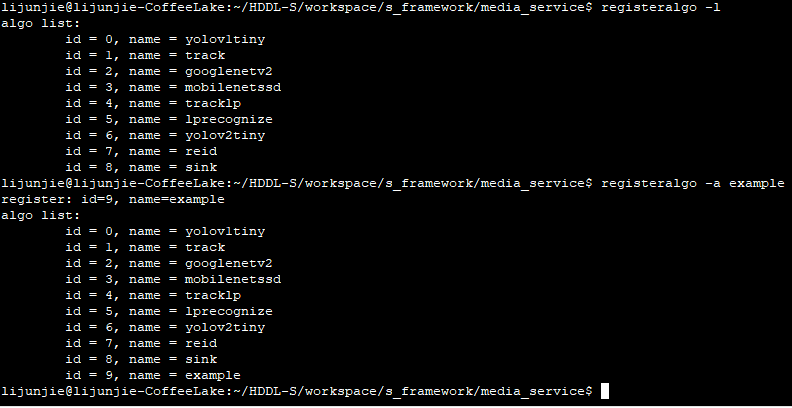
A reference is shown in gstreamer\_plugin\_openVINO/customer/example

1). Implement 5 APIs and build into a dynamical library, named lib<algoname>.so

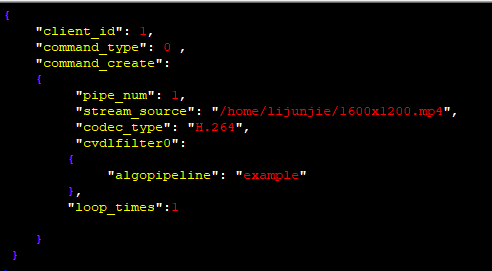
2). export CVDL\_MODEL\_FULL\_PATH=<cvdl\_model\_full\_path>

3). Cp lib<algoname>.so $ CVDL\_MODEL\_FULL\_PATH/<algoname>/.

## 2.3 Register this new model



## 2.4 configure hddl pipeline command to run this new models



## 2.5 send command to HDDL-S web server