pyforkurento

Kelvin Gakuo

Nov 25, 2020

SETUP

pyforkurento is a Python client for Kurento Media Server (KMS). This SDK was built because (currently) only Node, Angular and Bower clients exist.

The documentation for this SDK project is organized into different sections:

- *Setup*
- *Recipes*
- *API Reference*

CHAPTER

ONE

SETUP

1. Install Kurento Media Server

```
sudo docker run --name kms -d -p 8888:8888 kurento/kurento-media-server
```

2. Install pyforkurento

```
pip install pyforkurento
```

3. pyforkurento runs as an application server. You'll need to install relevant packages for the web or mobile client

For Node, Angular etc.

```
npm install kurento-utils
```

For vanilla Javascript, the steps are:

I. Install Node and NPM

II. Install Bower

III. Unpack Kurento-Utils using Bower. In any dir:

```
bower install kurento-utils
```

Inside `bower_components/kurento_utils/js` find the file `kurento-utils.min.js`. Copy it to your working area

CHAPTER

TWO

OPTIONAL SETUP

1. To use GStreamer filters

```
sudo apt-get install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-plugins-
↪good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav␣
↪gstreamer1.0-doc gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-
↪gl gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-pulseaudio
```

2. To develop custom filters

```
sudo apt-get update && sudo apt-get install --yes kurento-media-server-dev
```

If you get the error: *Unable to locate package kurento-media-server-dev* try the following:

1. Open this link in your browser
2. Extract the downloaded folder to a location of choice
3. `cd` into that folder then test using:

```
sh kurento-module-scaffold.sh TestKMSFilter ../custom_kurento_module opencv_filter
```

CHAPTER

THREE

RECIPES

Before you start using `pyforkurento`, make sure you have a pretty good understanding of KMS. Go through this refresher first.

Follow the instructions in the recipes dir here.

3.1 Client

class pyforkurento.client.**KurentoClient**(*kurento_server_url*)

pyforkurento entry point

create_media_pipeline()

Create a Media Pipeline. This HAS TO BE the first operation when dealing with KMS. Every other element is created by a pipeline

Returns

- MediaPipeline object

ping()

Prints 'pong' if a connection to KMS is available. Otherwise, an expection is thrown

3.2 MediaPipeline

class pyforkurento.pipeline.**MediaPipeline**(*session_id*, *pipeline_id*, *client_class*)

Base class for adding Media Elements to a Media Pipeline

add_endpoint(*endpoint*, ***kwargs*)

Adds an Endpoint Media Element to the pipeline

Params:

- **endpoint (str): String representing the type of endpoint to create. Accepts:**
  - WebRtcEndpoint
  - RtpEndpoint
  - HttpPostEndpoint
  - PlayerEndpoint
  - RecorderEndpoint
- **kwargs: Optional named arguments for different endpoints:**

- webrtc_recv_only (bool): Sets a WebRtcEndpoint to be a receiver only
- webrtc_send_only (bool): Sets a WebRtcEndpoint to be a sender only
- buffer_size (int): Milliseconds to buffer an RTSP stream in PlayerEndpoint
- **uri (str): Media URI for RecorderEndpoint & PlayerEndpoint only.**
  - **For PlayerEndpoint, it's the media to be played**

    Accepted URI schemas are:
    - file:///path/to/file (File on local file system)
    - rtsp://<server-ip> (IP camera RTSP URLs)
    - http(s)://<server-ip>/path/to/file (File on HTTP server)
  - **For RecorderEndpoint, it's the location to record to**

    Accepted URI schemas are:
    - file:///path/to/file (File on local file system)
    - http(s)://<server-ip>/path/to/file (File on HTTP server)

Returns
- Object of the requested endpoint

add_hub (*hub*, ***kwargs*)

Adds a Hub Media Element to the pipeline

Params:
- **hub (str): String representing the type of hub to create. Accepts:**
  - Composite
  - Dispatcher
  - DispatcherOneToMany

Returns
- Object of the requested hub

apply_filter (*filter*, ***kwargs*)

Applies a Filter to the stream

Params:
- **filter (str): String representing the type of filter to create. Accepts:**
  - FaceOverlayFilter - Overlays an image on a face detected
  - ZBarFilter - Triggers an event when a QR code is detected
  - GStreamerFilter - (https://gstreamer.freedesktop.org/documentation/installing/index.html)
  - ImageOverlayFilter - Overlays an image on the stream
- **kwargs: Optional named arguments for different endpoints:**
  - command (str): The gstreamer command (https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html)
  - **filter_type (str): The type of the gstreamer filter (AUDIO, VIDEO, or AUTODETECT)**

    The textoverlay command DOES NOT work

    If a command in unrecognised, you might be missing a filter. Try:

    sudo apt install libgstreamer-plugins-bad1.0 libgstreamer-plugins-base1.0 libgstreamer-plugins-good1.0 libgstreamer-plugins-ugly1.0

Returns
- Object of the requested filter

3.3 Media Element

class pyforkurento.media_element.**MediaElement**(*session_id*, *elem_id*, *pipeline_class*)

Base class for ALL media elements i.e. Endpoints, Filters, and Hubs

_add_event_listener(*event*, *callback*)

[DO NOT OVERRIDE!!] Adds event listeners for events that all Media Elements can implement

Params:
- **event: The event to listen for. Accepted:**
  - MediaFlowIn - Invoked when media is ready for recording
  - MediaFlowOut - Invoked when media is no longer ready for recording
  - EndOfStream - Invoked when the stream that the element sends out is finished.
  - ElementConnected - Indicates that an element has been connected to other.
  - ElementDisconnected - Indicates that an element has been disconnected.
  - Error: An error related to the MediaObject has occurred.
- callback: Function to be called when event is registered

3.4 Endpoints

class pyforkurento.endpoints.**Endpoint**(*sess_id*, *point_id*, *pipeline_class*)

Bases: *pyforkurento.media_element.MediaElement*

All endpoints base class

class pyforkurento.endpoints.**PlayerEndpoint**(*session_id*, *elem_id*, *pipeline_class*)

Bases: *pyforkurento.endpoints.Endpoint*

An input endpoint that retrieves content from file system, HTTP URL or RTSP URL and injects it into the media pipeline.

add_event_listener(*event*, *callback*)

Adds an event listener function for a specific PlayerEndpoint event or a general MediaElement event

connect(*sink_elem=None*)

Connect PlayerEndpoint to another element

Params: sink_elem (obj): Media Element to connect to. If left blank, the element connects to itself

pause()

Pause playing the media item

play()

Start playing the media item

stop()

Stop playing the media item

class pyforkurento.endpoints.**WebRTCEndpoint**(*session_id*, *elem_id*, *pipeline_class*)

Bases: *pyforkurento.endpoints.Endpoint*

An output and input endpoint that provides media streaming for Real Time Communications (RTC) through the web. It implements WebRTC technology to communicate with browsers.

add_event_listener(*event*, *callback*)

Adds an event listener function for a specific WebRTCEndpoint event or a general MediaElement event

Params:

- **event (str): The event to listen for. Accepted:**
  - OnIceCandidate - Invoked when KMS starts generating ICE candidates
  - OnIceGatheringDone - Invoked when KMS is done gathering ICE candidates
- callback (func): Function to be called when event is registered

add_ice_candidate(*candidate*)

Adds Ice Candidate recevied from the WebRTC client to KMS

Params:

- candidate (ICE): The ICE Candidate from the client

connect(*sink_elem=None*)

Connect WebRTCEndpoint to another element

Params: sink_elem (obj): Media Element to connect to. If left blank, the element connects to itself

gather_ice_candidates()

Triggers ICE candidate generation by KMS. Call this method AFTER adding an event listener for 'OnIceCandidate'

process_offer(*session_desc_offer*)

Process the Session Description Protocol offer generated by the client

Params:

- session_desc_offer (str): SDP payload from a WebRTC client

Returns SDP answer from KMS. If a problem occured, sdp_answer is set to be 'error'

Return type

- sdp_answer (str)

class pyforkurento.endpoints.**RecorderEndpoint**(*session_id*, *elem_id*, *pipeline_class*)

Bases: *pyforkurento.endpoints.Endpoint*

An output endpoint that provides function to store contents in reliable mode (doesn't discard data).

add_event_listener(*event*, *callback*)

Adds an event listener function for a specific RecorderEndpoint event or a general MediaElement event

Params:

- **event (str): The event to listen for. Accepted:**
  - Recording - Invoked when the media recording effectively starts
- callback (func): Function to be called when event is registered

connect(*sink_elem=None*)

Connect RecorderEndpoint to another element

Params: sink_elem (obj): Media Element to connect to. If left blank, the element connects to itself

record()

Start the recording the media item

stop()

Stops the recording

class pyforkurento.endpoints.**RTPEndpoint**(*session_id*, *elem_id*, *pipeline_class*)

Bases: *pyforkurento.endpoints.Endpoint*

An output and input endpoint. That is, provides bidirectional content delivery capabilities with remote networked peers through RTP protocol.

class pyforkurento.endpoints.**HTTPPostEndpoint**(*session_id*, *elem_id*, *pipeline_class*)
Bases: *pyforkurento.endpoints.Endpoint*

An input endpoint that accepts media using http POST requests like HTTP file upload function.

3.5 Filters

class pyforkurento.filters.**Filter**(*sess_id*, *filter_id*, *pipeline_class*)
Bases: *pyforkurento.media_element.MediaElement*

All filters base class

class pyforkurento.filters.**FaceOverlayFilter**(*session_id*, *elem_id*, *pipeline_class*)
Bases: *pyforkurento.filters.Filter*

Detects faces in a video stream and overlays them with a configurable image.

add_event_listener(*event*, *callback*)
Adds an event listener function for a specific FaceOverlayFilter event or a general MediaElement event

connect(*sink_elem=None*)
Connect FaceOverlayFilter to another element
Params: sink_elem (obj): Media Element to connect to. If left blank, the element connects to itself

set_face_overlay_image(*image_uri*, *offset_x=0.0*, *offset_y=0.0*, *width=1.0*, *height=1.0*)
Sets the image to overlay on a detected face
Params:
- **image_uri (str): Location of image to use. Accepted URI schemas are:**
  - file:///path/to/file (File on local file system)
  - http(s)://<server-ip>/path/to/file (File on HTTP server)
- offset_x (float): How much left or right (a ratio of face width) to move the image in reference to detected face's upper right corner coordinates
- offset_y (float): How much up or down (a ratio of face height) to move the image in reference to detected face's upper right corner coordinates
- width (float) >=0.0 : How much of the face's width the image should cover e.g. 1.0 means cover the entire width
- height (float) >=0.0: How much of the face's height the image should cover e.g. 1.0 means cover the entire height

unset_face_overlay_image()
Removes the image overlayed on faces

class pyforkurento.filters.**ImageOverlayFilter**(*session_id*, *elem_id*, *pipeline_class*)
Bases: *pyforkurento.filters.Filter*

Overlays a configurable image on the video stream

add_event_listener(*event*, *callback*)
Adds an event listener function for a specific ImageOverlayFilter event or a general MediaElement event

connect(*sink_elem=None*)
Connect ImageOverlayFilter to another element
Params: sink_elem (obj): Media Element to connect to. If left blank, the element connects to itself

overlay_image(*image_uri*, *image_id*, *offset_x=0.0*, *offset_y=0.0*, *relative_width=1.0*, *relative_height=1.0*, *keep_aspect_ratio=True*, *to_centre=True*)

Draws an image on the video feed at the specified location

Params:

- image_id (str): A unique identifier for the image. Recommendation: str(uuid.uuid4())
- **image_uri (str): Location of image to use. Accepted URI schemas are:**
  - file:///path/to/file (File on local file system)
  - http(s)://<server-ip>/path/to/file (File on HTTP server)
- offset_x (float) [0 - 1]: Percentage of image width to set overlay image left upper conner X coords
- offset_y (float) [0 - 1]: Percentage of image height to set overlay image left upper conner Y coords
- relative_width (float) [0 - 1]: Width of the overlay image in relation to the video stream e.g. 1.0 means full width
- relative_height (float) [0 - 1]: Height of the overlay image in relation to the video stream e.g. 1.0 means full height
- keep_aspect_ratio (bool): Whether to keep the image's aspect ratio
- to_centre (bool): Whether to centre the image in the region defined

remove_image()

Remove the overlayed image from the stream

class pyforkurento.filters.**ZBarFilter**(*session_id*, *elem_id*, *pipeline_class*)

Bases: *pyforkurento.filters.Filter*

Detects QR and bar codes in a video stream. When a code is found, the filter raises a CodeFoundEvent

add_event_listener(*event*, *callback*)

Adds an event listener function for a specific ZBarFilter event or a general MediaElement event

Params:

- **event (str): The event to listen for. Accepted:**
  - CodeFoundEvent - Triggered when a BarCode or QR code is found in the video stream
- callback (func): Function to be called when event is registered

connect(*sink_elem=None*)

Connect ZBarFilter to another element

Params: sink_elem (obj): Media Element to connect to. If left blank, the element connects to itself

class pyforkurento.filters.**GStreamerFilter**(*session_id*, *elem_id*, *pipeline_class*)

Bases: *pyforkurento.filters.Filter*

A generic filter interface that allows usage of GStreamer filters in Kurento Media Pipelines.

add_event_listener(*event*, *callback*)

Adds an event listener function for a specific GStreamerFilter event or a general MediaElement event

connect(*sink_elem=None*)

Connect GStreamerFilter to another element

Params: sink_elem (obj): Media Element to connect to. If left blank, the element connects to itself

3.6 Hubs

class pyforkurento.hubs.**Hub**(*sess_id*, *hub_id*, *pipeline_class*)

Bases: *pyforkurento.media_element.MediaElement*

All hubs base class

class pyforkurento.hubs.**Composite**

Bases: *pyforkurento.hubs.Hub*

A hub that mixes the audio stream of its connected inputs and constructs a grid with the video streams of them.

class pyforkurento.hubs.**Dispatcher**

Bases: *pyforkurento.hubs.Hub*

A hub that allows routing between arbitrary input-output HubPort pairs.

class pyforkurento.hubs.**DispatcherOneToMany**

Bases: *pyforkurento.hubs.Hub*

A hub that sends a given input to all the connected output HubPorts.

PYTHON MODULE INDEX

p

INDEX

Symbols

A

C

D

E

F

G

H

I

K

M

O

P

R

S

U

W

Z