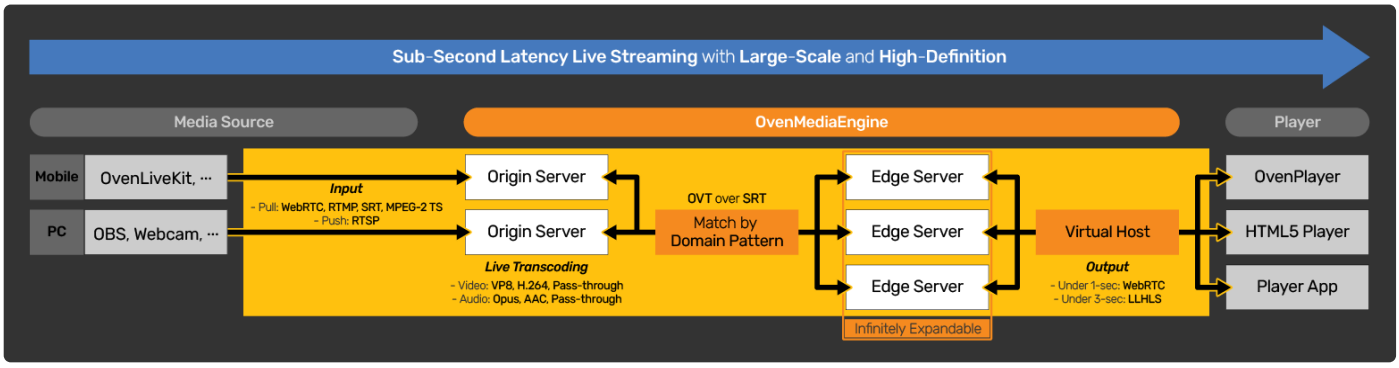
0.14.[9-10]

Introduction

What is OvenMediaEngine?

OvenMediaEngine (OME) is a **Sub-Second Latency Live Streaming Server** with **Large-Scale** and **High-Definition**. With OME, you can create platforms/services/systems that transmit high-definition video to hundreds-thousand viewers with sub-second latency and be scalable, depending on the number of concurrent viewers.



OvenMediaEngine can receive a video/audio, video, or audio source from encoders and cameras such as OvenLiveKit, OBS, XSplit, and more, to WebRTC, SRT, RTMP, MPEG-2 TS, and RTSP as Input. Then, OME transmits this source using **LLHLS** (Low Latency HLS) and **WebRTC** as output. Also, we provide OvenPlayer, an Open-Source and JavaScript-based WebRTC/LLHLS Player for OvenMediaEngine.

Our goal is to make it easier for you to build a stable broadcasting/streaming service with sub-second latency.

Features

- **Ingest**
  - Push: WebRTC, SRT, RTMP, MPEG-2 TS
  - Pull: RTSP
- **Adaptive Bitrate Streaming (ABR) for LLHLS and WebRTC**
- **Low-Latency Streaming using LLHLS**
- **Sub-Second Latency Streaming using WebRTC**
  - WebRTC over TCP (with embedded TURN server)
  - Embedded WebRTC Signaling Server (WebSocket based)
  - Retransmission with NACK
  -

    - ULPFEC (Uneven Level Protection Forward Error Correction)
        - *VP8, H.264*
    - In-band FEC (Forward Error Correction)
        - *Opus*
- **Embedded Live Transcoder**
    - Video: VP8, H.264, Pass-through
    - Audio: Opus, AAC, Pass-through
- **Clustering** (Origin-Edge Structure)
- **Monitoring**
- **Access Control**
    - AdmissionWebhooks
    - SignedPolicy
- **File Recording**
- **Push Publishing using RTMP and MPEG-TS** (Re-streaming)
- **Thumbnail**
- **REST API**
- **Experiment**
    - P2P Traffic Distribution (Only WebRTC)

Supported Platforms

We have tested OvenMediaEngine on platforms, listed below. However, we think it can work with other Linux packages as well:

- Docker (https://hub.docker.com/r/airensoft/ovenmediaengine)
- Ubuntu 18+
- CentOS 7+
- Fedora 28+

Getting Started

Please read Getting Started chapter in the tutorials.

How to Contribute

Thank you so much for being so interested in OvenMediaEngine.

We need your help to keep and develop our open-source project, and we want to tell you that you can contribute in many ways. Please see our Guidelines, Rules, and Contribute.

- Finding Bugs
- Reviewing Code
- Sharing Ideas
- Testing
- Improving Documentation
- Spreading & Use Cases
- Recurring Donations

We always hope that OvenMediaEngine will give you good inspiration.

For more information

- OvenMediaEngine GitHub
- OvenMediaEngine Website
- OvenMediaEngine Tutorial Source
- Test Player
  - *Without TLS: http://demo.ovenplayer.com*
  - *With TLS: https://demo.ovenplayer.com*
- OvenPlayer Github
- AirenSoft Website

License

Getting Started

Running with Docker

OvenMediaEngine supports the Docker image from AirenSoft's Docker Hub (**airensoft/ovenmediaengine)** repository. After installing Docker, you can simply run the following command:

```
docker run -d \
-p 1935:1935 -p 4000:4000/udp -p 3333:3333 -p 3334:3334 -p 3478:3478 -p 9000:9000 -p 9999:9999
airensoft/ovenmediaengine:0.14.9
```

> To use TLS, you must set up a certificate. See TLS Encryption for more information.

You can set the following environment variables.

Ports

Env	Default Value
OME_ORIGIN_PORT	9000
OME_RTMP_PROV_PORT	1935
OME_SRT_PROV_PORT	9999/udp
OME_MPEGTS_PROV_PORT	4000/udp
OME_LLHLS_STREAM_PORT	3333
OME_LLHLS_STREAM_TLS_PORT	3334
OME_WEBRTC_SIGNALLING_PORT	3333
OME_WEBRTC_SIGNALLING_TLS_PORT	3334
OME_TCP_RELAY_ADDRESS	*:3478

Manual Installation and Execution

Install dependencies

OvenMediaEngine can work with a variety of open-sources and libraries. First, install them on your clean Linux machine as described below. We think that OME can support most Linux packages, but the tested platforms we use are Ubuntu 18+, Fedora 28+, and CentOS 7+.

```
(curl -LOJ https://github.com/AirenSoft/OvenMediaEngine/archive/v0.14.9.tar.gz && tar xvfz Ov
OvenMediaEngine-0.14.9/misc/prerequisites.sh
```

> If the prerequisites.sh script fails, proceed with the manual installation.

****

Build & Run

You can build the OvenMediaEngine source using the following command:

Ubuntu 18

```
sudo apt-get update
cd OvenMediaEngine-0.14.9/src
make release
sudo make install
systemctl start ovenmediaengine
# If you want automatically start on boot
systemctl enable ovenmediaengine.service
```

Fedora 28

```
sudo dnf update
cd OvenMediaEngine-0.14.9/src
make release
sudo make install
systemctl start ovenmediaengine
# If you want automatically start on boot
systemctl enable ovenmediaengine.service
```

CentOS 7

```
sudo yum update
source scl_source enable devtoolset-7
cd OvenMediaEngine-0.14.9/src
make release
sudo make install
systemctl start ovenmediaengine
# If you want automatically start on boot
systemctl enable ovenmediaengine.service
```

In addition, we recommend that you permanently set environment variables as follows.

```
$ echo 'source scl_source enable devtoolset-7' >> ~/.bashrc
```

> ⓘ if `systemctl start ovenmediaengine` fails in Fedora, SELinux may be the cause. See Check SELinux section of Troubleshooting.

Ports used by default

The default configuration uses the following ports, so you need to open it in your firewall settings.

Port	Purpose
1935/TCP	RTMP Input
9999/UDP	SRT Input
4000/UDP	MPEG-2 TS Input
9000/TCP	Origin Server (OVT)
3333/TCP 3334/TLS	LLHLS Streaming *** Streaming over Non-TLS is not allowed with modern browsers.**
3333/TCP 3334/TLS	WebRTC Signaling (both ingest and streaming)
3478/TCP	WebRTC TCP relay (TURN Server, both ingest and streaming)
10000 - 1005/UDP	WebRTC Ice candidate (both ingest and streamin

> ⚠ To use TLS, you must set up a certificate. See TLS Encryption for more information.

You can open firewall ports as in the following example:

```
$ sudo firewall-cmd --add-port=3333/tcp
$ sudo firewall-cmd --add-port=3334/tcp
$ sudo firewall-cmd --add-port=1935/tcp
$ sudo firewall-cmd --add-port=9999/udp
$ sudo firewall-cmd --add-port=4000/udp
$ sudo firewall-cmd --add-port=3478/tcp
$ sudo firewall-cmd --add-port=9000/tcp
$ sudo firewall-cmd --add-port=10000-10005/udp
```

Hello Sub-Second Latency Streaming

Start Streaming

You can live streaming using live encoders such as OBS, XSplit, and OvenStreamEncoder. Please set the RTMP URL as below:

`rtmp://<Server IP>[:<RTMP Port>]/<Application name>/<Stream name>`

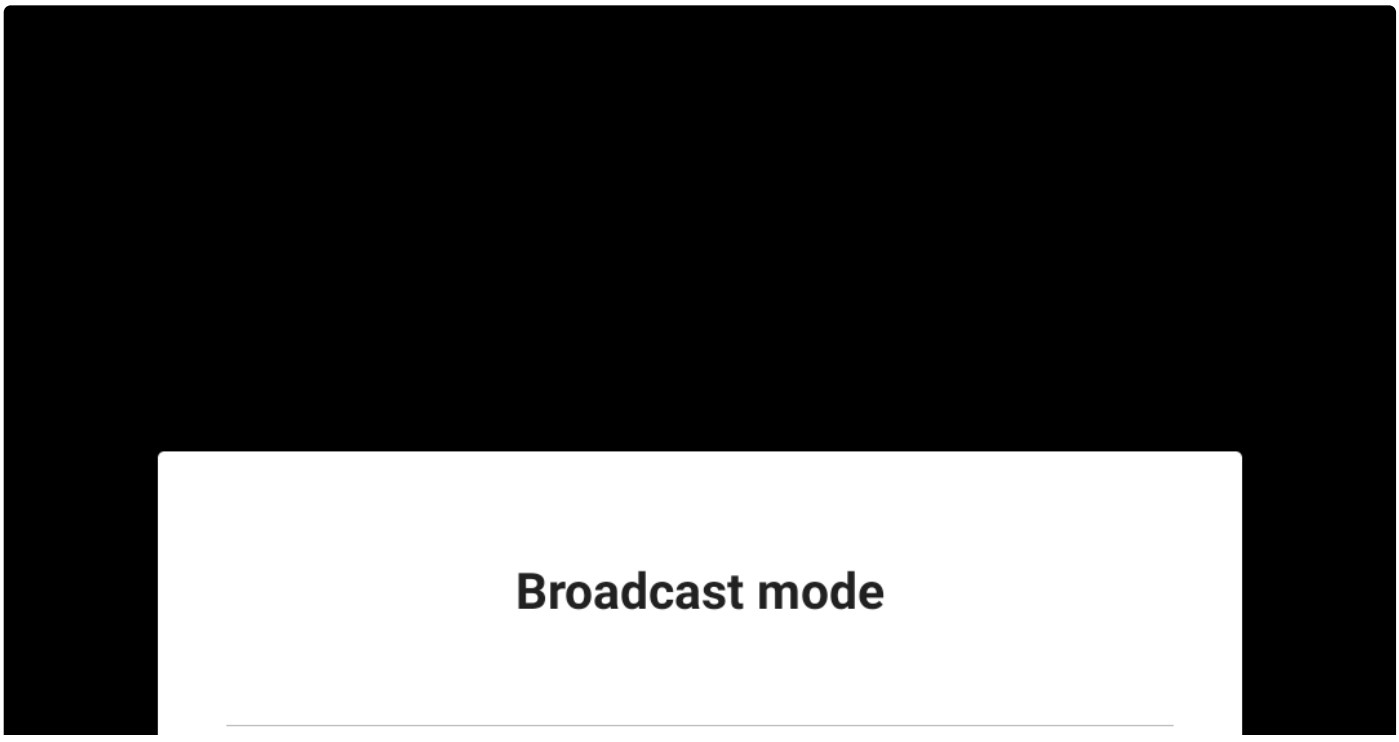
The meanings of each item are as follows:

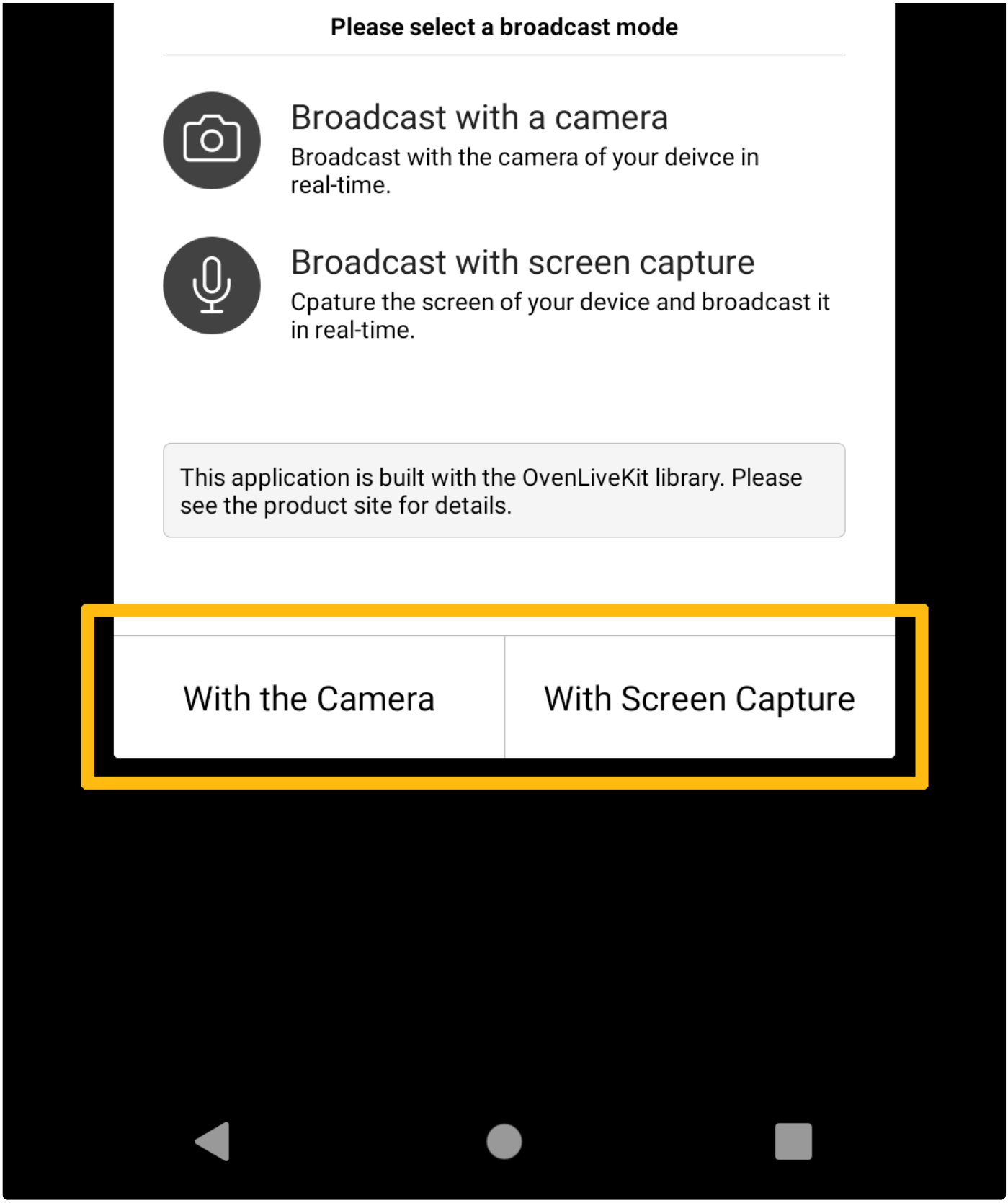
- `<Server IP>` : IP address or domain of the OvenMediaEngine server.
- `<RTMP Port>` : You can use `<Port>` of `<Provider>` in the above `Server.xml` file. With the default configuration, the RTMP default port (1935) is used. Also, by setting the default port, you can omit the port.
- `<Application name>` : This value corresponds to `<Name>` of `<Application>` in `conf/Server.xml` . If you use the default configuration, you can use the `app` .
- `<Stream name>` : Name of the stream you defined.

After you enter the above RTMP URL into the encoder and start publishing, you will have an environment in which the player can view the live stream.

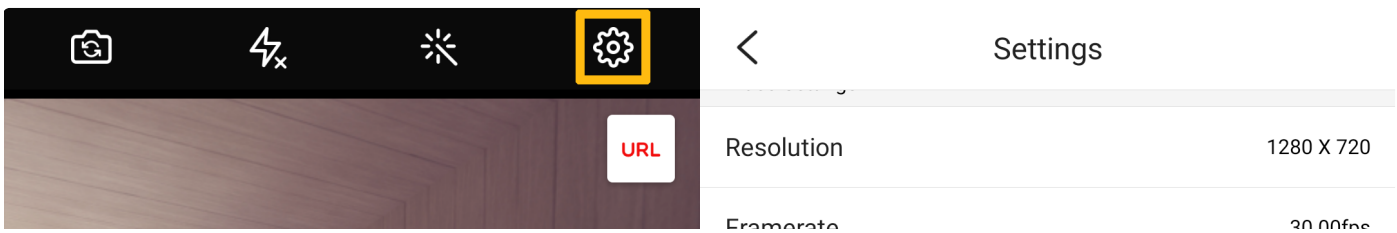
Example of using OvenLiveKit (OvenStreamEncoder)

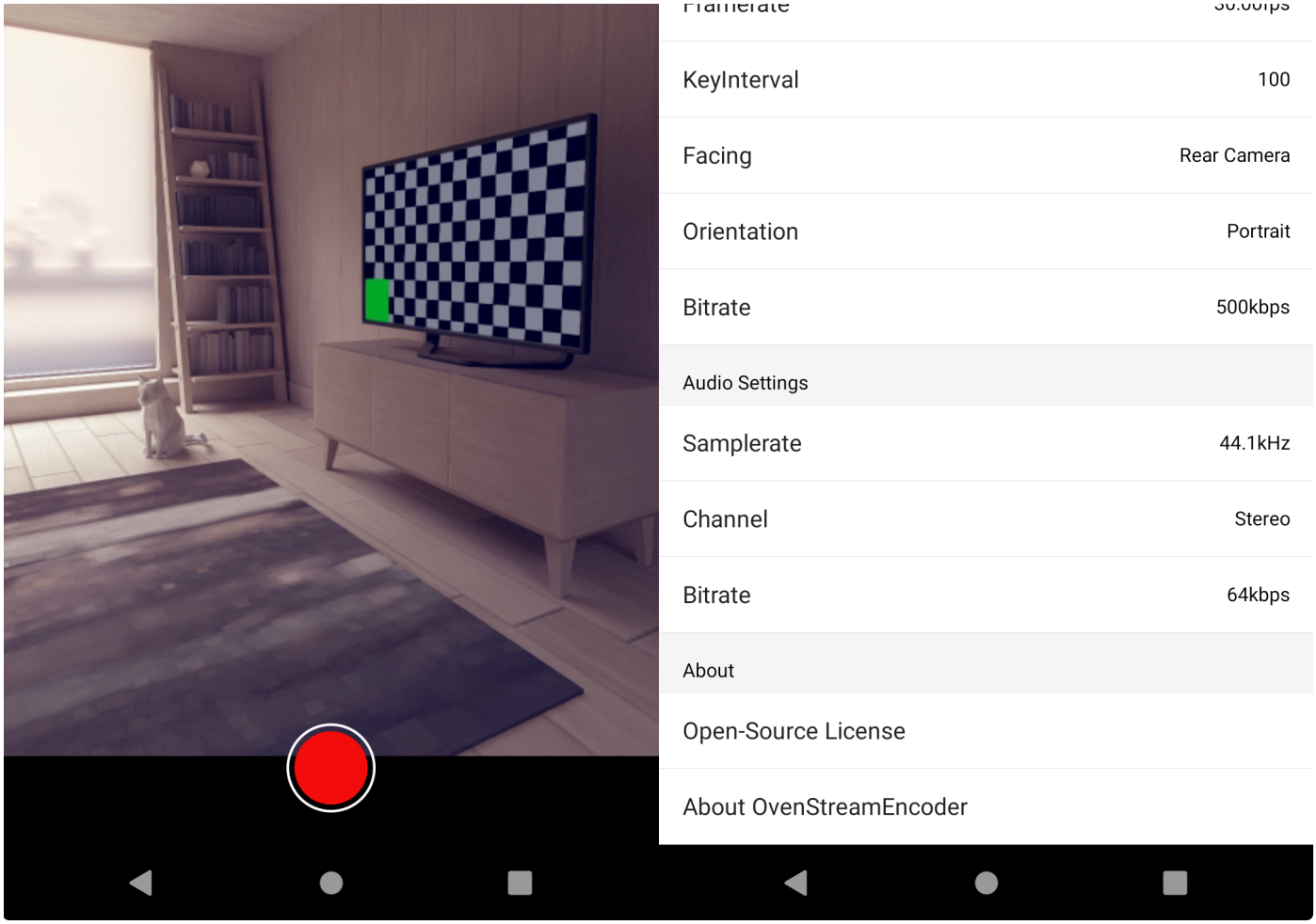
OvenLiveKit is a transmission SDK. So, you can easily add broadcast transmission functions to your apps using this SDK. And OvenStreamEncoder is a sample app that shows you can make and use it with OvenLiveKit. You can use it by searching OvenStreamEncoder in Google Play.



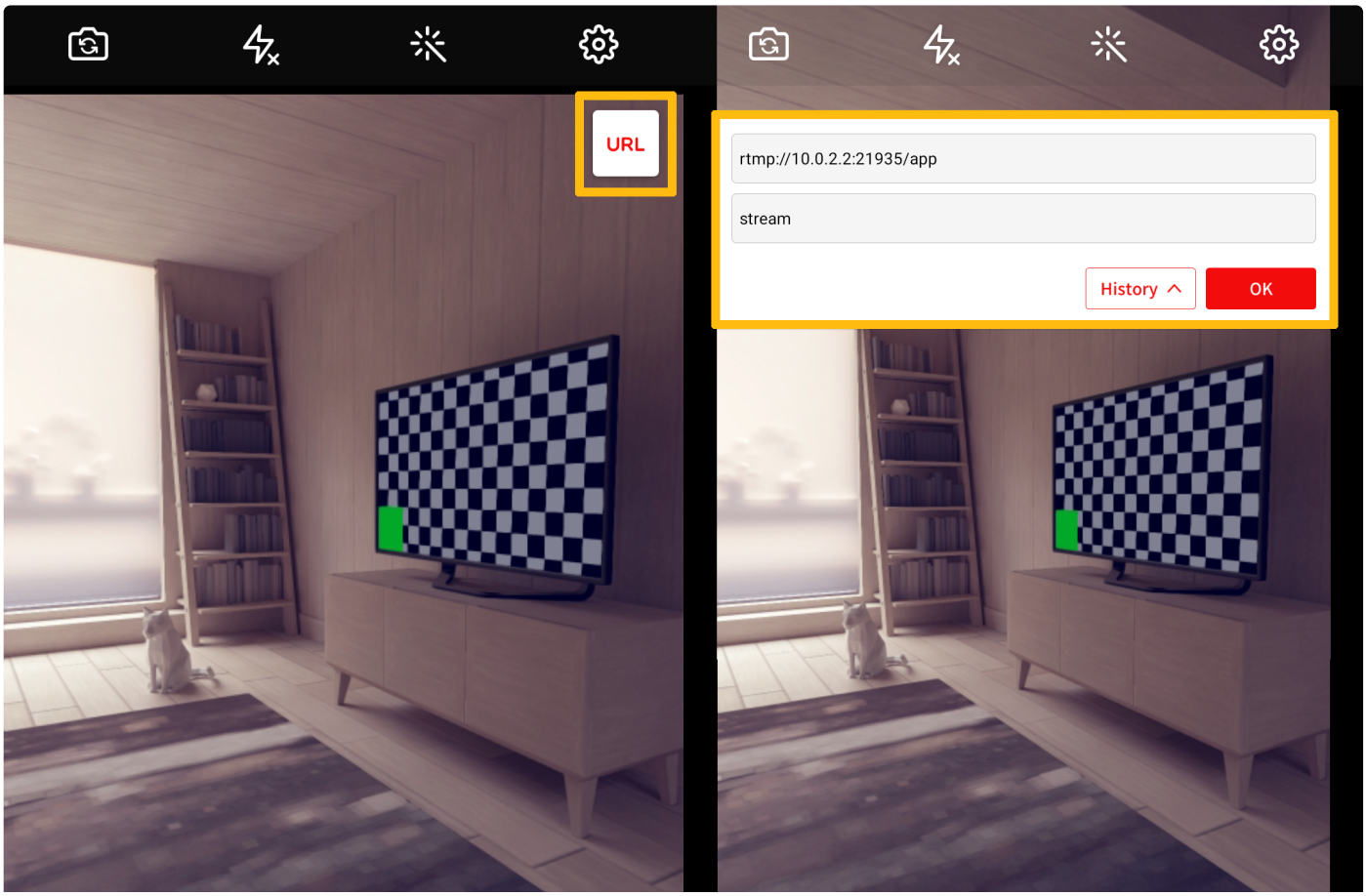


OvenStreamEncoder supports the mode that implements the streaming of the screen recorded by a camera and another that performs the streaming of the current screen by capturing it.

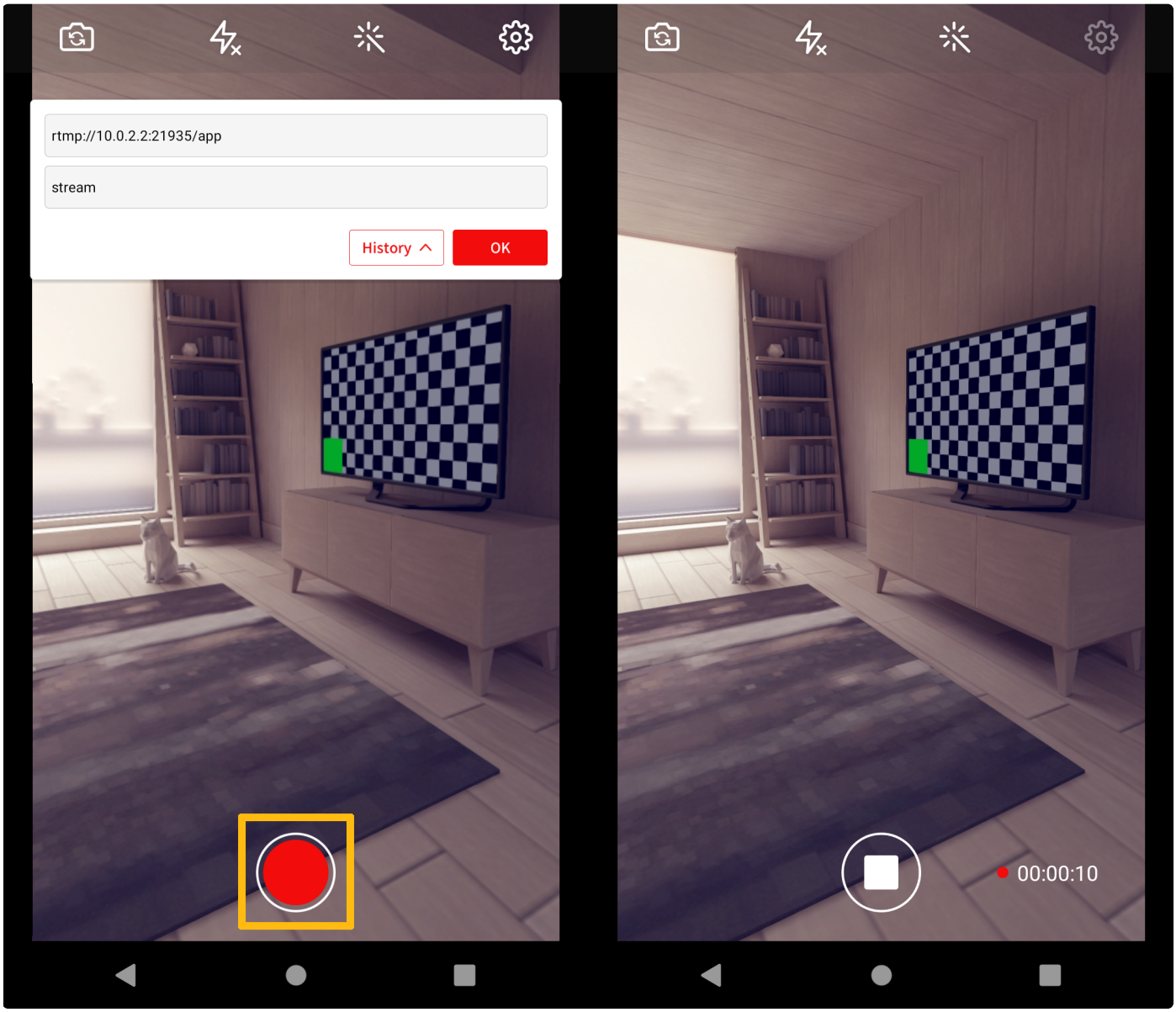




So, select the mode along with the broadcasting concept you want and proceed with the optimum setting by pressing the Setting icon at this right upper position.



After broadcast setting, return to the original screen, press the URL button at right upper, and input **RTMP URL** and **Stream Key** to transmitting.



If all preparation is ready, begin the broadcast by pressing the Recording button at the center bottom.

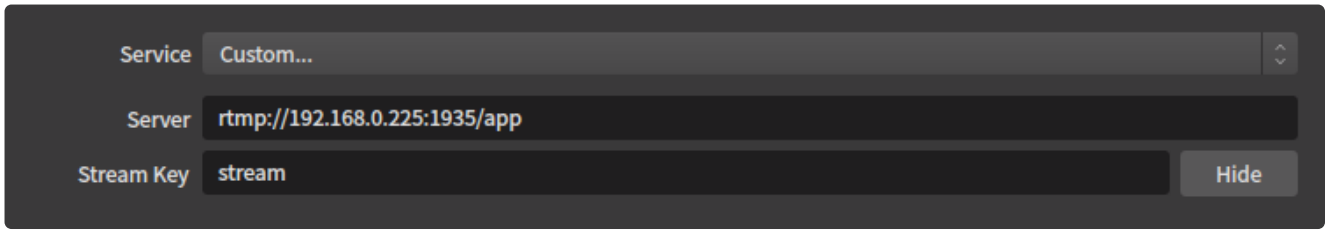
Also, we ran tests to see how well OvenStreamEnocder is optimized for OvenMediaEngine. If you are interested, click HERE to check.

Example of using OBS Encoder

The server address in OBS needs to use `<Application name>` generated in `Server.xml`. If you use

the default configuration, the app is already created and ready to use.

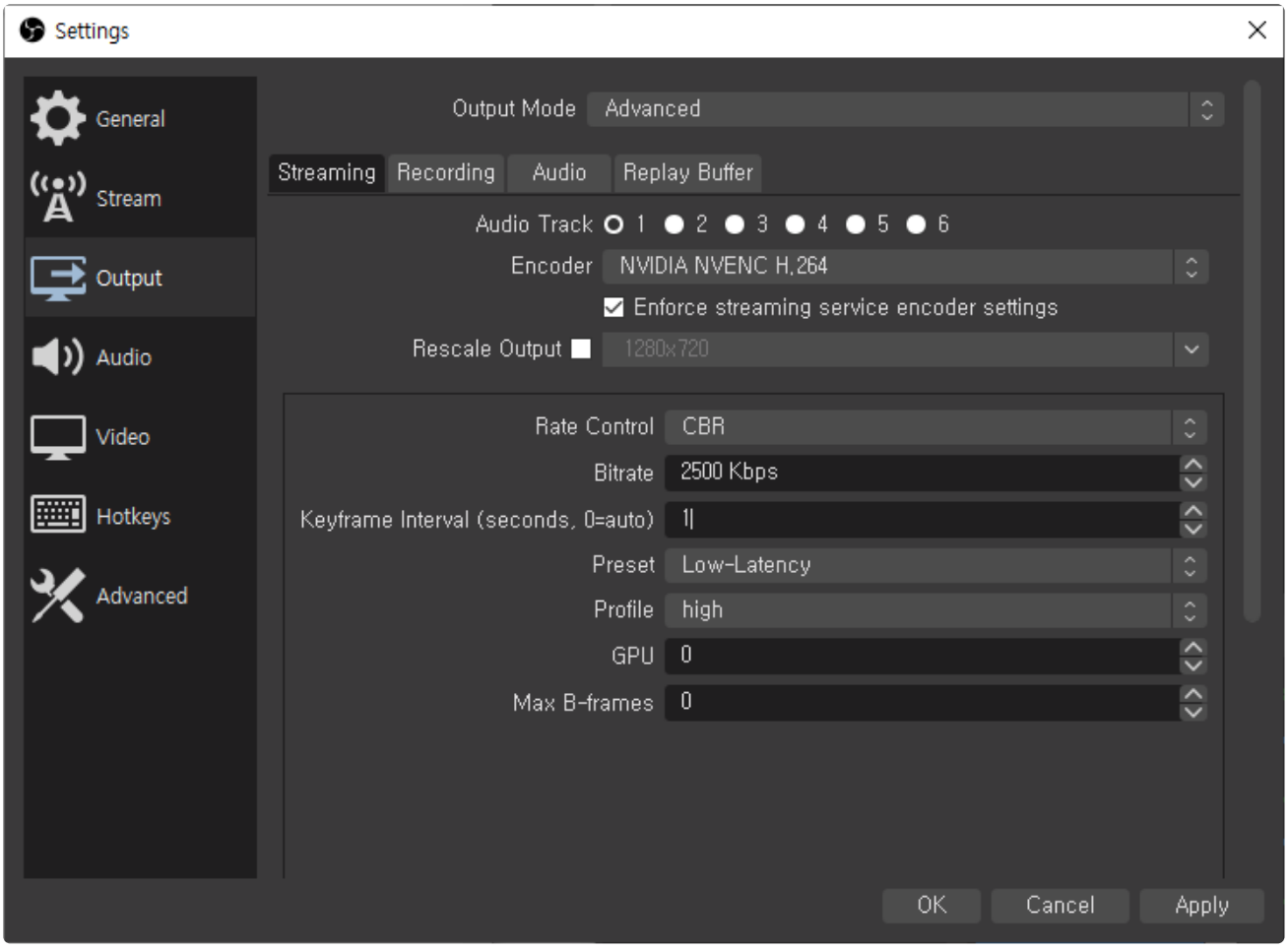
- Install OBS on your PC and run it.
- Click "File" in the top menu, then click "Settings" (or press "Settings" on the lower right).
- Select the "Stream" tab and enter your stream information.



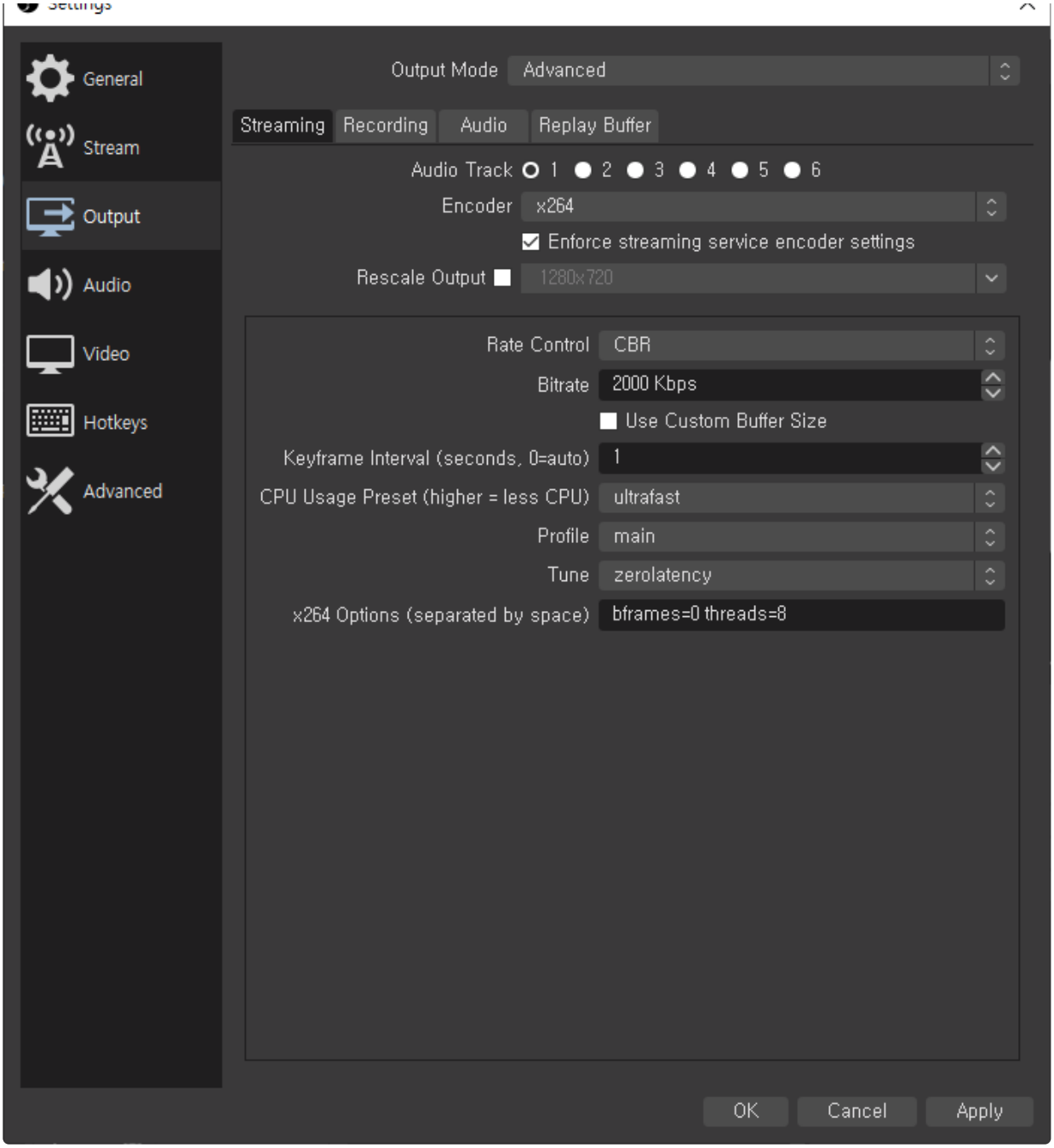
Press "Service" and select "Custom", your OBS is the same as this image.

- Go to the "Output" tab.
- Set the following entries.

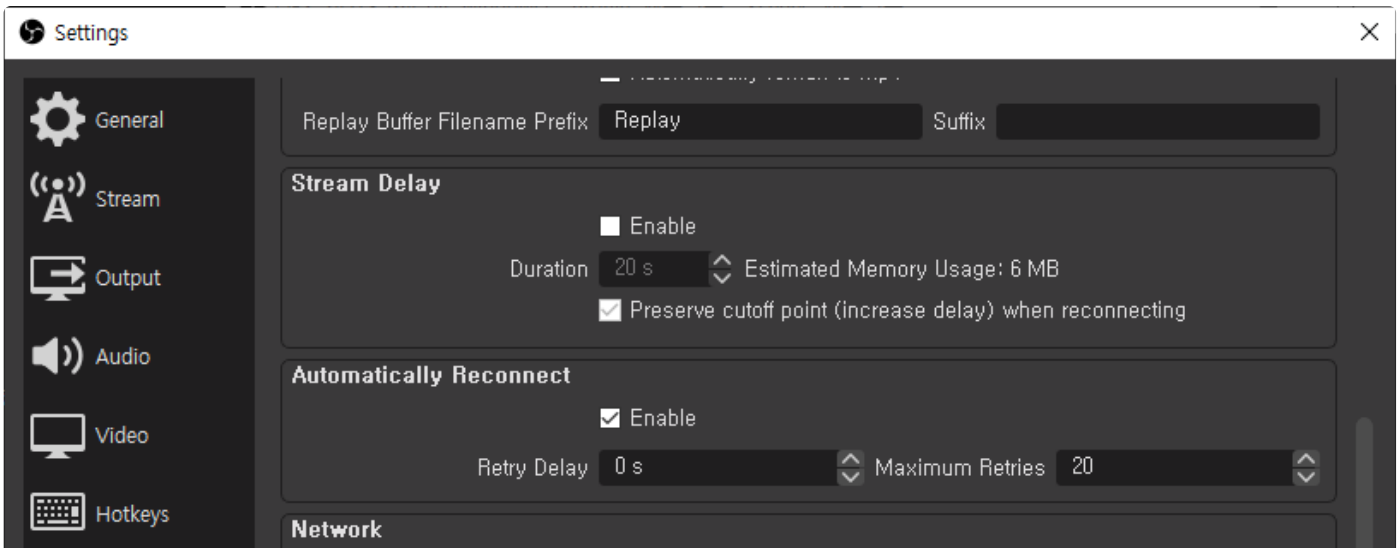
For lower latency, we recommend using the Hardware Encoder as follows: "NVENC" provides a Low-Latency preset. It's also important to set "MAX B-frame = 0" to reduce latency.

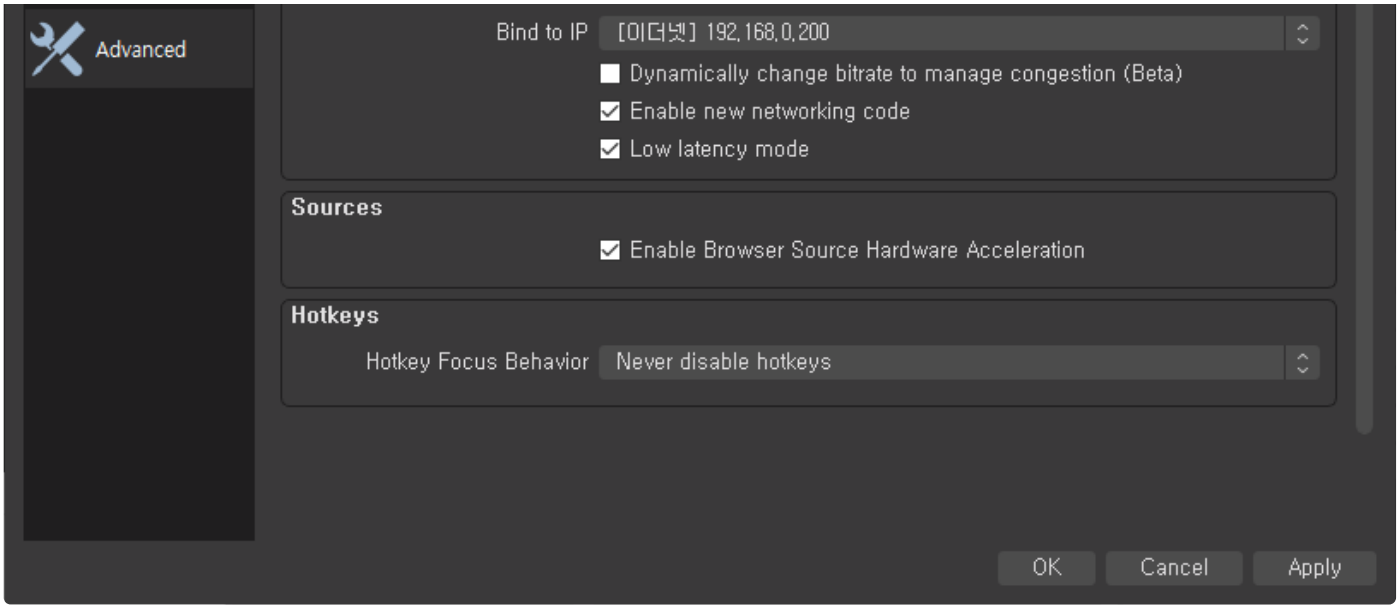


If Hardware Encoder isn't installed on your PC, it's recommended to set x264 as follows: We highly recommend setting "bframes = 0" to reduce latency. Then set the "threads" option to 8 or less. Chrome doesn't handle more than 10 Nal Units. The best way to avoid this is to set "thread = 8".



We recommend checking "Enable new networking code" and "Low latency mode" on Network in Advanced Settings as follows:





Playback

We have prepared a test player so that you can easily check if OvenMediaEngine is working properly. Please see the chapter on Test Player for more information.

Please note that WebRTC Signalling URL is similar to the RTMP URL and consists of the following:

- `ws://<Server IP>:[<Signalling Port>/<Application name>/<Output Stream name>[?transport=tcp]`
  - `<Server IP>`: IP address or domain of the OvenMediaEngine server.
  - `<Signalling Port>`: You can use the value of `<Signalling><ListenPort>` in `Server.xml` above. If you use the default configuration, the WebRTC Signalling default port (3333) is used.
  - `<Application name>`: This value corresponds to `<Name>` of `<Application>` in `conf/Server.xml`. If you use the default configuration, you can use the `app`.
  - `<Output Stream name>`: You have to use an output stream name for streaming. If you use the default configuration, an output stream named `<Stream Name>` is automatically generated when the stream is input.
  - `?transport=tcp` : You can use this query string to play through webrtc over tcp. Useful in environments with severe packet loss.

> As of version 0.10.4, the default output stream name has been changed from **<Input Stream Name>_o** to **<Input Stream Name>**, and has been updated to use the input stream name as the output stream name for convenience.

> If you use the default configuration and the RTMP publishing URL is `rtmp://192.168.0.1:1935/app/stream`
>
> then the WebRTC URL will be `ws://192.168.0.1:3333/app/stream`

> In addition,
>
> LLHLS streaming URL will be `https://domain:3334/app/stream/llhls.m3u8`

If you want to build OvenPlayer in your environment, see OvenPlayer QuickStart.

Configuration

OvenMediaEngine has an XML configuration file. If you start OvenMediaEngine with `systemctl start ovenmediaengine`, the config file is loaded from the following path.

```
/usr/share/ovenmediaengine/conf/Server.xml
```

If you run it directly from the command line, it loads the configuration file from:

```
/<OvenMediaEngine Binary Path>/conf/Server.xml
```

If you run it in Docker container, the path to the configuration file is:

```
# For Origin mode
/opt/ovenmediaengine/bin/origin_conf/Server.xml
# For Edge mode
/opt/ovenmediaengine/bin/edge_conf/Server.xml
```

Server

The `Server` is the root element of the configuration file. The `version` attribute indicates the version of the configuration file. OvenMediaEngine uses this version information to check if the config file is a compatible version.

```
<?xml version="1.0" encoding="UTF-8"?>
<Server version="8">
    <Name>OvenMediaEngine</Name>
    <IP>*</IP>
    <PrivacyProtection>false</PrivacyProtection>
    <StunServer>stun.l.google.com:19302</StunServer>
    <Bind>...</Bind>
    <VirtualHosts>...</VirtualHosts>
</Server>
```

IP

```
    <IP>*</IP>
```

The `IP address` is OvenMediaEngine will bind to. If you set *, all IP addresses of the system are used. If you enter a specific IP, the Host uses that IP only.

PrivacyProtection

PrivacyProtection is an option to comply with GDPR, PIPEDA, CCPA, LGPD, etc. by deleting the client's personal information (IP, Port) from all records. When this option is turned on, the client's IP and Port are converted to `xxx.xxx.xxx.xxx:xxx` in all logs and REST APIs.

StunServer

OvenMediaEngine needs to know its public IP in order to connect to the player through WebRTC. The server must inform the player of the IceCandidates and TURN server addresses when signaling, and this information must be the IP the player can connect to. However, in environments such as Docker or AWS, public IP cannot be obtained through a local interface, so a method of obtaining public IP using stun server is provided (available from version 0.11.1).

If OvenMediaEngine obtains the public IP through communication with the set stun server, you can set the public IP by using * or ${PublicIP} in IceCandidate and TcpRelay.

```
<StunServer>stun.l.google.com:19302</StunServer>
```

Bind

The `Bind` is the configuration for the server port that will be used. Bind consists of `Providers` and `Publishers`. The Providers are the server for stream input, and the Publishers are the server for streaming.

```
<!-- Settings for the ports to bind -->
<Bind>
        <!-- Enable this configuration if you want to use API Server -->
        <!--
        <Managers>
                <API>
                        <Port>8081</Port>
                        <WorkerCount>1</WorkerCount>
                </API>
        </Managers>
        -->
```

```
        <Providers>
                <!-- Pull providers -->
                <RTSPC>
                        <WorkerCount>1</WorkerCount>
                </RTSPC>
                <OVT>
                        <WorkerCount>1</WorkerCount>
                </OVT>
                <!-- Push providers -->
                <RTMP>
                        <Port>1935</Port>
                        <WorkerCount>1</WorkerCount>
                </RTMP>
                <SRT>
                        <Port>9999</Port>
                        <WorkerCount>1</WorkerCount>
                </SRT>
                <MPEGTS>
                        <!--
                                Listen on port 4000~4005 (<Port>4000-4004,4005/udp</Port>)
                                This is just a demonstration to show that you can configure th
                        -->
                        <Port>4000/udp</Port>
                </MPEGTS>
                <WebRTC>
                        <Signalling>
                                <Port>3333</Port>
                                <TLSPort>3334</TLSPort>
                                <WorkerCount>1</WorkerCount>
                        </Signalling>

                        <IceCandidates>
                                <IceCandidate>*:10000/udp</IceCandidate>
                                <!--
                                        If you want to stream WebRTC over TCP, specify IP:Port
                                        This uses the TURN protocol, which delivers the strean
                                        For detailed information, refer https://airensoft.gitl
                                -->
                                <TcpRelay>*:3478</TcpRelay>
                                <!-- TcpForce is an option to force the use of TCP rather than
                                <TcpForce>true</TcpForce>
                                <TcpRelayWorkerCount>1</TcpRelayWorkerCount>
                        </IceCandidates>
                </WebRTC>
        </Providers>

        <Publishers>
                <OVT>
                        <Port>9000</Port>
                        <WorkerCount>1</WorkerCount>
                </OVT>
                <LLHLS>
                        <!--
```

```
                        OME only supports h2 so LLHLS works over HTTP/1.1 on non-TLS ports.
                        LLHLS works with higher performance over HTTP/2,
                        so it is recommended to use a TLS port.
                        -->
                        <Port>3333</Port>
                        <!-- If you want to use TLS, specify the TLS port -->
                        <TLSPort>3334</TLSPort>
                        <WorkerCount>1</WorkerCount>
                </LLHLS>
                <WebRTC>
                        <Signalling>
                                <Port>3333</Port>
                                <TLSPort>3334</TLSPort>
                                <WorkerCount>1</WorkerCount>
                        </Signalling>
                        <IceCandidates>
                                <IceCandidate>*:10000-10005/udp</IceCandidate>
                                <!--
                                        If you want to stream WebRTC over TCP, specify IP:Por
                                        This uses the TURN protocol, which delivers the strea
                                        For detailed information, refer https://airensoft.gitl
                                -->
                                <TcpRelay>*:3478</TcpRelay>
                                <!-- TcpForce is an option to force the use of TCP rather tha
                                <TcpForce>true</TcpForce>
                                <TcpRelayWorkerCount>1</TcpRelayWorkerCount>
                        </IceCandidates>
                </WebRTC>
        </Publishers>
</Bind>
```

The meaning of each element is shown in the following table:

Element	Description
<Managers><API>	REST API Server port
RTMP	RTMP port for incoming RTMP stream.
SRT	SRT port for incoming SRT stream
MPEG-TS	MPEGTS ports for incoming MPEGTS/UDP stream.
WebRTC	Port for WebRTC. If you want more information on the WebRTC port, see the WebRTC Ingest and WebRTC Streaming chapters.
OVT	OVT port for an origin server. OVT is a protocol defined by OvenMediaEngine f Origin-Edge communication. For more informatio about Origin-Edge, see the Origin-Edge Clusterin chapter.

LLHLS	HTTP(s) port for LLHLS streaming.

Virtual Host

`VirtualHosts` are a way to run more than one streaming server on a single machine. OvenMediaEngine supports IP-based virtual host and Domain-based virtual host. "IP-based" means that you can separate streaming servers into multiples by setting different IP addresses, and "Domain-based" means that even if the streaming servers use the same IP address, you can split the streaming servers into multiples by setting different domain names.

`VirtualHosts` consist of `Name`, `Host`, `Origins`, `SignedPolicy`, and `Applications`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Server version="8">
    <Name>OvenMediaEngine</Name>
    <VirtualHosts>
        <VirtualHost>
            <Name>default</Name>
            <Host>
            ...
            </Host>

            <Origins>
            ...
            </Origins>

            <SignedPolicy>
            ...
            </SignedPolicy>

            <Applications>
            ...
            </Applications>
        </Host>
    </Hosts>
</Server>
```

Host

The Domain has `Names` and TLS. Names can be either a domain address or an IP address. Setting * means it allows all domains and IP addresses.

```
<Host>
        <Names>
            <!-- Domain names
            <Name>stream1.airensoft.com</Name>
```

```
            <Name>stream2.airensoft.com</Name>
            <Name>*.sub.airensoft.com</Name>
            -->
            <Name>*</Name>
        </Names>
        <TLS>
            <CertPath>path/to/file.crt</CertPath>
            <KeyPath>path/to/file.key</KeyPath>
            <ChainCertPath>path/to/file.crt</ChainCertPath>
        </TLS>
</Host>
```

SignedPolicy

SignedPolicy is a module that limits the user's privileges and time. For example, operators can distribute RTMP URLs that can be accessed for 60 seconds to authorized users, and limit RTMP transmission to 1 hour. The provided URL will be destroyed after 60 seconds, and transmission will automatically stop after 1 hour. Users who are provided with a SingedPolicy URL cannot access resources other than the provided URL. This is because the SignedPolicy URL is authenticated. See the SignedPolicy chapter for more information.

Origins

Origins (also we called OriginMap) are a feature to pull streams from external servers. It now supports OVT and RTSP for the pulling protocols. OVT is a protocol defined by OvenMediaEngine for Origin-Edge communication. It allows OvenMediaEngine to relay a stream from other OvenMediaEngines that have OVP Publisher turned on. Using RTSP, OvenMediaEngine pulls a stream from an RTSP server and creates a stream. RTSP stream from external servers can stream by WebRTC, HLS, and MPEG-DASH.

The Origin has `Location` and `Pass` elements. Location is a URI pattern for incoming requests. If the incoming URL request matches Location, OvenMediaEngine pulls the stream according to a Pass element. In the Pass element, you can set the origin stream's protocol and URLs.

To run the Edge server, Origin creates application and stream if there isn't those when user request. For more learn about Origin-Edge, see the Live Source chapter.

```
<Origins>
    <Origin>
        <Location>/app/stream</Location>
        <Pass>
            <Scheme>ovt</Scheme>

            <Urls><Url>origin.com:9000/app/stream_720p</Url></Urls>
        </Pass>
    </Origin>
    <Origin>
        <Location>/app/</Location>
        <Pass>
            <Scheme>ovt</Scheme>
            <Urls><Url>origin.com:9000/app/</Url></Urls>
        </Pass>
```

```
    </Origin>
    <Origin>
        <Location>/rtsp/stream</Location>
        <Pass>
            <Scheme>rtsp</Scheme>
            <Urls><Url>rtsp-server.com:554/</Url></Urls>
        </Pass>
    </Origin>
    <Origin>
        <Location>/</Location>
        <Pass>
            <Scheme>ovt</Scheme>
            <Urls><Url>origin2.com:9000/</Url></Urls>
        </Pass>
    </Origin>
</Origins>
```

Application

`<Application>` consists of various elements that can define the operation of the stream, including Stream input, Encoding, and Stream output. In other words, you can create as many `<Application>` as you like and build various streaming environments.

```
<VirtualHost>
    ...
    <Applications>
        <Application>
            ...
        </Application>
        <Application>
            ...
        </Application>
    </Applications>
</VirtualHost>
```

`<Application>` needs to set `<Name>` and `<Type>` as follows:

```
<Application>
    <Name>app</Name>
    <Type>live</Type>
    <OutputProfiles> ... </OutputProfiles>
    <Providers> ... </Providers>
    <Publishers> ... </Publishers>
</Application>
```

- `<Name>` is used to configure the Streaming URL.
- `<Type>` defines the operation of `<Application>`. Currently, there is only a `live` type.

OutputProfiles

`<OutputProfile>` is a configuration that creates an output stream. Output stream name can be set with `<OutputStreamName>`, and transcoding properties can be set through `<Encodes>`. If you want to stream one input to multiple output streams, you can set multiple `<OutputProfile>`.

```
<Application>
    <OutputProfiles>
            <OutputProfile>
                <Name>bypass_stream</Name>
                <OutputStreamName>${OriginStreamName}</OutputStreamName>
                <Encodes>
                    <Audio>
                        <Bypass>true</Bypass>
                    </Audio>
                    <Video>
                        <Bypass>true</Bypass>
                    </Video>
                    <Audio>
                        <Codec>opus</Codec>
                        <Bitrate>128000</Bitrate>
                        <Samplerate>48000</Samplerate>
                        <Channel>2</Channel>
                    </Audio>
                    <!--
                    <Video>
                        <Codec>vp8</Codec>
                        <Bitrate>1024000</Bitrate>
                        <Framerate>30</Framerate>
                        <Width>1280</Width>
                        <Height>720</Height>
                    </Video>
                    -->
                </Encodes>
            </OutputProfile>
        </OutputProfiles>
</Application>
```

For more information about the OutputProfiles, please see the Transcoding chapter.

Providers

`Providers` ingest streams that come from a media source.

```
<Application>
   <Providers>
      <RTMP/>
      <WebRTC/>
      <SRT/>
      <RTSPPull/>
```

```
      <OVT/>
      <MPEGTS>
         <StreamMap>
            ...
         </StreamMap>
      </MPEGTS>
   </Providers>
</Application>
```

If you want to get more information about the `<Providers>`, please refer to the Live Source chapter.

Publishers

You can configure the Output Stream operation in `<Publishers>`. `<ThreadCount>` is the number of threads used by each component responsible for the `<Publishers>` protocol.

> ⓘ You need many threads to transmit streams to a large number of users at the same time. So it's better to use a higher core CPU and set `<ThreadCount>` equal to the number of CPU cores.

```
<Application>
   <Publishers>
      <OVT />
      <HLS />
      <DASH />
      <LLDASH />
      <WebRTC />
   </Publishers>
</Application>
```

OvenMediaEngine currently supports WebRTC, Low-Latency DASH, MEPG-DASH, and HLS. If you don't want to use any protocol then you can delete that protocol setting, the component for that protocol isn't initialized. As a result, you can save system resources by deleting the settings of unused protocol components.

If you want to learn more about WebRTC, visit the WebRTC Streaming chapter. And if you want to get more information on Low-Latency DASH, MPEG-DASH, and HLS, refer to the chapter on HLS & MPEG-DASH Streaming.

Configuration Example

Finally, `Server.xml` is configured as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<Server version="8">
        <Name>OvenMediaEngine</Name>
```

```
        <!-- Host type (origin/edge) -->
        <Type>origin</Type>
        <!-- Specify IP address to bind (* means all IPs) -->
        <IP>*</IP>
        <PrivacyProtection>false</PrivacyProtection>

        <!--
        To get the public IP address(mapped address of stun) of the local server.
        This is useful when OME cannot obtain a public IP from an interface, such as AWS or do
        If this is successful, you can use ${PublicIP} in your settings.
        -->
        <StunServer>stun.l.google.com:19302</StunServer>

        <Modules>
                <!--
                Currently OME only supports h2 like all browsers do. Therefore, HTTP/2 only wo
                -->
                <HTTP2>
                        <Enable>true</Enable>
                </HTTP2>

                <LLHLS>
                        <Enable>true</Enable>
                </LLHLS>

                <!-- P2P works only in WebRTC and is experiment feature -->
                <P2P>
                        <!-- disabled by default -->
                        <Enable>false</Enable>
                        <MaxClientPeersPerHostPeer>2</MaxClientPeersPerHostPeer>
                </P2P>
        </Modules>

<!-- Settings for the ports to bind -->
<Bind>
        <!-- Enable this configuration if you want to use API Server -->
        <!--
        <Managers>
                <API>
                        <Port>8081</Port>
                        <TLSPort>8082</TLSPort>
                        <WorkerCount>1</WorkerCount>
                </API>
        </Managers>
        -->

        <Providers>
                <!-- Pull providers -->
                <RTSPC>
                        <WorkerCount>1</WorkerCount>
                </RTSPC>
                <OVT>
                        <WorkerCount>1</WorkerCount>
```

```
                </OVT>
                <!-- Push providers -->
                <RTMP>
                        <Port>1935</Port>
                        <WorkerCount>1</WorkerCount>
                </RTMP>
                <SRT>
                        <Port>9999</Port>
                        <WorkerCount>1</WorkerCount>
                </SRT>
                <MPEGTS>
                        <!--
                                Listen on port 4000~4005 (<Port>4000-4004,4005/udp</Port>)
                                This is just a demonstration to show that you can configure th
                        -->
                        <Port>4000/udp</Port>
                </MPEGTS>
                <WebRTC>
                        <Signalling>
                                <Port>3333</Port>
                                <TLSPort>3334</TLSPort>
                                <WorkerCount>1</WorkerCount>
                        </Signalling>

                        <IceCandidates>
                                <IceCandidate>*:10000/udp</IceCandidate>
                                <!--
                                        If you want to stream WebRTC over TCP, specify IP:Port
                                        This uses the TURN protocol, which delivers the strean
                                        For detailed information, refer https://airensoft.gith
                                -->
                                <TcpRelay>*:3478</TcpRelay>
                                <!-- TcpForce is an option to force the use of TCP rather than
                                <TcpForce>true</TcpForce>
                                <TcpRelayWorkerCount>1</TcpRelayWorkerCount>
                        </IceCandidates>
                </WebRTC>
        </Providers>

        <Publishers>
                <OVT>
                        <Port>9000</Port>
                        <WorkerCount>1</WorkerCount>
                </OVT>
                <LLHLS>
                        <!--
                        OME only supports h2, so LLHLS works over HTTP/1.1 on non-TLS ports.
                        LLHLS works with higher performance over HTTP/2,
                        so it is recommended to use a TLS port.
                        -->
                        <Port>3333</Port>
                        <!-- If you want to use TLS, specify the TLS port -->
                        <TLSPort>3334</TLSPort>
```

```
                        <WorkerCount>1</WorkerCount>
                </LLHLS>
                <WebRTC>
                        <Signalling>
                                <Port>3333</Port>
                                <TLSPort>3334</TLSPort>
                                <WorkerCount>1</WorkerCount>
                        </Signalling>
                        <IceCandidates>
                                <IceCandidate>*:10000-10005/udp</IceCandidate>
                                <!--
                                        If you want to stream WebRTC over TCP, specify IP:Por
                                        This uses the TURN protocol, which delivers the strea
                                        For detailed information, refer https://airensoft.gitl
                                -->
                                <TcpRelay>*:3478</TcpRelay>
                                <!-- TcpForce is an option to force the use of TCP rather tha
                                <TcpForce>true</TcpForce>
                                <TcpRelayWorkerCount>1</TcpRelayWorkerCount>
                        </IceCandidates>
                </WebRTC>
        </Publishers>
</Bind>

        <!--
                Enable this configuration if you want to use API Server

                <AccessToken> is a token for authentication, and when you invoke the API, you
                For example, if you set <AccessToken> to "ome-access-token", you must set "Bas
        -->
        <!--
        <Managers>
                <Host>
                        <Names>
                                <Name>*</Name>
                        </Names>
                        <TLS>
                                <CertPath>path/to/file.crt</CertPath>
                                <KeyPath>path/to/file.key</KeyPath>
                                <ChainCertPath>path/to/file.crt</ChainCertPath>
                        </TLS>
                </Host>
                <API>
                        <AccessToken>ome-access-token</AccessToken>

                        <CrossDomains>
                                <Url>*.airensoft.com</Url>
                                <Url>http://*.sub-domain.airensoft.com</Url>
                                <Url>http?://airensoft.*</Url>
                        </CrossDomains>
                </API>
        </Managers>
        -->
```

```
<VirtualHosts>
        <!-- You can use wildcard like this to include multiple XMLs -->
        <VirtualHost include="VHost*.xml" />
        <VirtualHost>
                <Name>default</Name>
                <!--Distribution is a value that can be used when grouping the same vh
                <Distribution>ovenmediaengine.com</Distribution>

                <!-- Settings for multi ip/domain and TLS -->
                <Host>
                        <Names>
                                <!-- Host names
                                        <Name>stream1.airensoft.com</Name>
                                        <Name>stream2.airensoft.com</Name>
                                        <Name>*.sub.airensoft.com</Name>
                                        <Name>192.168.0.1</Name>
                                -->
                                <Name>*</Name>
                        </Names>
                        <!--
                        <TLS>
                                <CertPath>path/to/file.crt</CertPath>
                                <KeyPath>path/to/file.key</KeyPath>
                                <ChainCertPath>path/to/file.crt</ChainCertPath>
                        </TLS>
                        -->
                </Host>

                <!--
                Refer https://airensoft.gitbook.io/ovenmediaengine/signedpolicy
                <SignedPolicy>
                        <PolicyQueryKeyName>policy</PolicyQueryKeyName>
                        <SignatureQueryKeyName>signature</SignatureQueryKeyName>
                        <SecretKey>aKq#1kj</SecretKey>

                        <Enables>
                                <Providers>rtmp,webrtc,srt</Providers>
                                <Publishers>webrtc,hls,llhls,dash,lldash</Publishers>
                        </Enables>
                </SignedPolicy>
                -->

                <!--
                <AdmissionWebhooks>
                        <ControlServerUrl></ControlServerUrl>
                        <SecretKey></SecretKey>
                        <Timeout>3000</Timeout>
                        <Enables>
                                <Providers>rtmp,webrtc,srt</Providers>
                                <Publishers>webrtc,hls,llhls,dash,lldash</Publishers>
                        </Enables>
                </AdmissionWebhooks>
```

```
                        -->
                        <!-- <Origins>
                                <Properties>
                                        <NoInputFailoverTimeout>3000</NoInputFailoverTimeout>
                                        <UnusedStreamDeletionTimeout>60000</UnusedStreamDelet
                                </Properties>
                                <Origin>
                                        <Location>/app/stream</Location>
                                        <Pass>
                                                <Scheme>ovt</Scheme>
                                                <Urls><Url>origin.com:9000/app/stream_720p</U
                                        </Pass>
                                        <ForwardQueryParams>false</ForwardQueryParams>
                                </Origin>
                                <Origin>
                                        <Location>/app/</Location>
                                        <Pass>
                                                <Scheme>ovt</Scheme>
                                                <Urls><Url>origin.com:9000/app/</Url></Urls>
                                        </Pass>
                                </Origin>
                                <Origin>
                                        <Location>/edge/</Location>
                                        <Pass>
                                                <Scheme>ovt</Scheme>
                                                <Urls><Url>origin.com:9000/app/</Url></Urls>
                                        </Pass>
                                </Origin>
                        </Origins> -->

                        <!-- Settings for applications -->
                        <Applications>
                                <Application>
                                        <Name>app</Name>
                                        <!-- Application type (live/vod) -->
                                        <Type>live</Type>
                                        <OutputProfiles>
                                                <!-- Enable this configuration if you want to
                                                <HardwareAcceleration>false</HardwareAccelera
                                                <OutputProfile>
                                                        <Name>bypass_stream</Name>
                                                        <OutputStreamName>${OriginStreamName}
                                                        <Encodes>
                                                                <Audio>
                                                                        <Bypass>true</Bypass>
                                                                </Audio>
                                                                <Video>
                                                                        <Bypass>true</Bypass>
                                                                </Video>
                                                                <Audio>
                                                                        <Codec>opus</Codec>
                                                                        <Bitrate>128000</Bitra
```

```
                                        <Samplerate>48000</Sa
                                        <Channel>2</Channel>
                                </Audio>
                                <!--
                                <Video>
                                        <Codec>vp8</Codec>
                                        <Bitrate>1024000</Bit
                                        <Framerate>30</Framer
                                        <Width>1280</Width>
                                        <Height>720</Height>
                                        <Preset>faster</Prese
                                </Video>
                                -->
                        </Encodes>
                </OutputProfile>
        </OutputProfiles>
        <Providers>
                <OVT />
                <WebRTC />
                <RTMP />
                <SRT />
                <MPEGTS>
                        <StreamMap>
                                <!--
                                        Set the stream name o
                                        For example, if a cli
                                        <Stream>
                                                <Name>stream_
                                                <Port>4000,40
                                        </Stream>
                                        <Stream>
                                                <Name>stream_
                                                <Port>4005</P
                                        </Stream>
                                -->
                                <Stream>
                                        <Name>stream_${Port}<
                                        <Port>4000</Port>
                                </Stream>
                        </StreamMap>
                </MPEGTS>
                <RTSPPull />
                <WebRTC>
                        <Timeout>30000</Timeout>
                </WebRTC>
        </Providers>
        <Publishers>
                <AppWorkerCount>1</AppWorkerCount>
                <StreamWorkerCount>8</StreamWorkerCount>
                <OVT />
                <WebRTC>
                        <Timeout>30000</Timeout>
                        <Rtx>false</Rtx>
```

```
                                                <UlpfecMaxFec>false</UlpfecMaxFec>
                                                <JitterBuffer>false</JitterBuffer>
                                        </WebRTC>
                                        <LLHLS>
                                                <ChunkDuration>0.2</ChunkDuration>
                                                <SegmentDuration>6</SegmentDuration>
                                                <SegmentCount>10</SegmentCount>
                                                <CrossDomains>
                                                        <Url>*</Url>
                                                </CrossDomains>
                                        </LLHLS>
                                </Publishers>
                        </Application>
                </Applications>
        </VirtualHost>
</VirtualHosts>
</Server>
```

TLS Encryption

Most browsers can't load resources via HTTP and WS (WebSocket) from HTTPS web pages secured with TLS. Therefore, if the player is on an HTTPS page, the player must request streaming through "https" and "wss" URLs secured with TLS. In this case, you must apply the TLS certificate to the OvenMediaEngine.

You can set the port for TLS in `TLSPort`. Currently, LLHLS and WebRTC Signaling support TLS.

```
<Bind>
        ...

        <Publishers>
        ...
                <LLHLS>
                        <Port>80</Port>
                        <TLSPort>443</TLSPort>
                </LLHLS>
                <WebRTC>
                        <Signalling>
                                <Port>3333</Port>
                                <TLSPort>3334</TLSPort>
                        </Signalling>
                        ...
                </WebRTC>
        </Publishers>
</Bind>
```

Add your certificate files to as follows:

```
<Host>
```

```
        <Names>
                <Name>*.airensoft.com</Name>
        </Names>
        <TLS>
                <CertPath>/etc/pki/airensoft.com/_airensoft_com.crt</CertPath>
                <KeyPath>/etc/pki/airensoft.com/_airensoft_com.key</KeyPath>
                <ChainCertPath>/etc/pki/airensoft.com/_airensoft_com.ca-bundle</ChainCertPath>
        </TLS>
</Host>
```

To configure HTTPs for HLS, and WebRTC Signaling server, the TLS element must be enabled. The `CertPath` has to indicate a server certificate and the `KeyPath` has to indicate a private key file. They can be set to absolute paths or relative paths from the executable. If the server certificate is issued using an intermediate certificate, some browsers may complain about a certificate. In this case, you should set a bundle of chained certificates provided by a Certificate Authority in `ChainCertPath`.

If you set up TLS, you can't set IP or * into <Name>. You can only set Domains that the certificate contains. If you have a certificate for `*.host.com`, it means you can set domains such as `aaa.host.com`, `bbb.host.com`, and `*.host.com`.

If the certificate settings are completed correctly, WebRTC streaming can be played `wss://url` with HLS and DASH streaming `https://url`.

Live Source

OvenMediaEngine supports multiple protocols for input from various live sources, without compromising basic usability. This allows you to publish a variety of live sources with sub-second latency. See the sub-page for more information.

RTMP

Configuration

`Providers` ingests streams that come from a media source. OvenMediaEngine supports RTMP protocol. You can set it in the configuration as follows:

```
<Server>
        ...
        <Bind>
                <Providers>
```

```
                        <RTMP>
                          <Port>1935</Port>
                        </RTMP>
                </Providers>
        </Bind>
        ...
        <VirtualHosts>
                <VirtualHost>
                        <Application>
                                <Providers>
                                        <RTMP>
                                ...
                            </RTMP>
                            ...
                                </Providers>
                        <Application>
                </VirtualHost>
        </VirtualHosts>
</Server>
```

When a live source inputs to the `<Application>`, a stream is automatically created in the `<Application>`. The created stream is passed to Encoder and Publisher.

RTMP live stream

If you set up a live stream using an RTMP-based encoder, you need to set the following in `Server.xml`:

```
<Application>
   <Providers>
      <RTMP>
         <BlockDuplicateStreamName>true</BlockDuplicateStreamName>
      </RTMP>
   </Providers>
<Application>
```

- `<BlockDuplicateStreamName>` is a policy for streams that are inputted as overlaps.

`<BlockDuplicateStreamName>` works with the following rules:

Value	Description
true	`Default` Rejects the new stream inputted as overlap and maintains the existing stream.
false	Accepts a new stream inputted as overlap and disconnects the existing stream.

> To allow the duplicated stream name feature can cause several problems. When a new stream is an input the player may be disconnected. Most encoders have the ability to automatically reconnect when it is disconnected from the server. As a result, two encoders compete and disconnect each other, which can cause serious problems in playback.

Publish

If you want to publish the source stream, you need to set the following in the Encoder:

- **`URL`** `RTMP://<OvenMediaEngine IP>[:<RTMP Listen Port>]/<App Name]>`
- **`Stream Key`** `Stream Name`

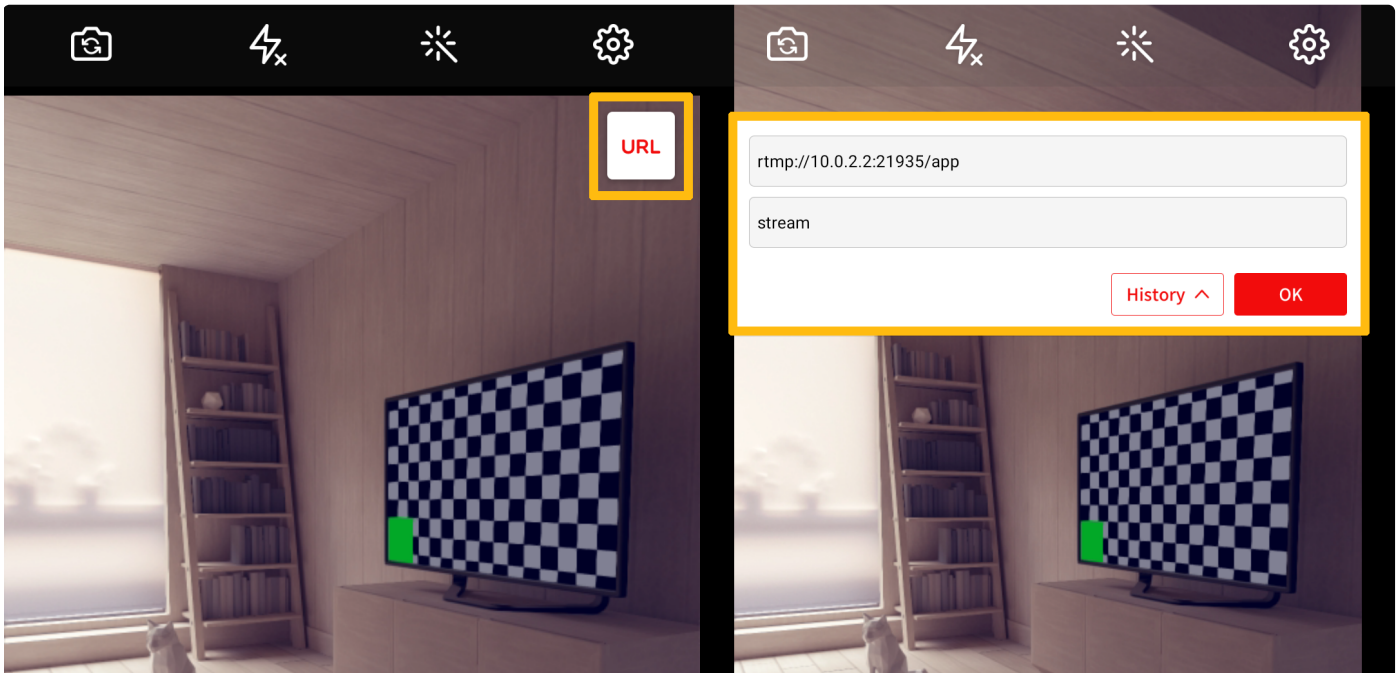
If you use the default configuration, the `<RTMP><ListenPort>` is 1935, which is the default port for RTMP. So it can be omitted. Also, since the Application named `app` is created by default in the default configuration, you can enter `app` in the `[App Name]`. You can define a Stream Key and use it in the Encoder, and the Streaming URL will change according to the Stream Key.

Moreover, some encoders can include a stream key in the URL, and if you use these encoders, you need to set it as follows:

- **`URL`** `RTMP://<OvenMediaEngine IP>[:<RTMP Listen Port>/<App Name>/<Stream Name>`

Example with OvenLiveKit (OvenStreamEncoder)

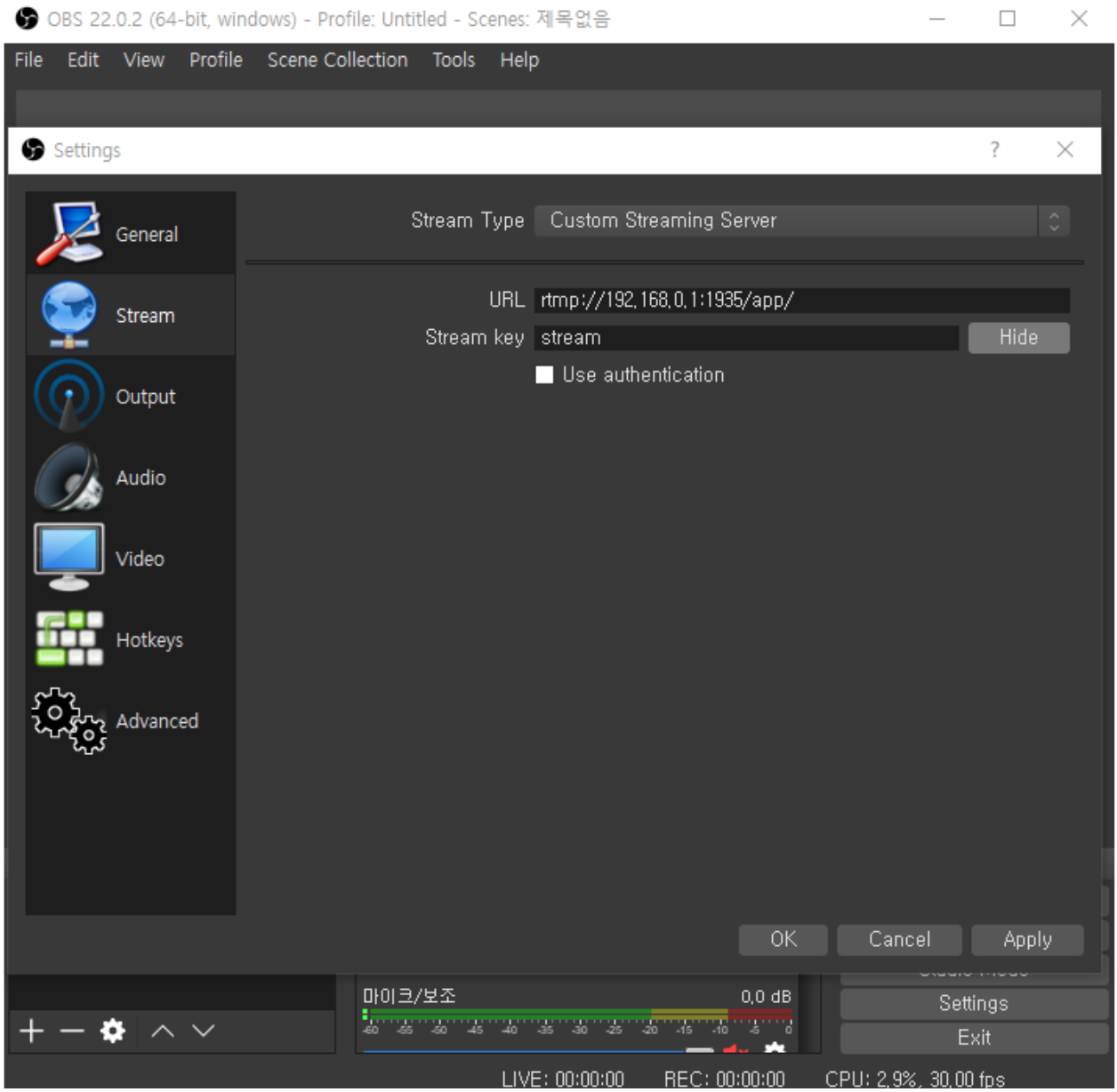
If you are using the default configuration, press the URL button in the top right corner of OvenStreamEnoder, and enter the URL as shown below:



Also, `<App name>` and `<Stream name>` can be changed and used as desired in the configuration.

Example with OBS

If you use the default configuration, set the OBS as follows:



You can set the Stream Key to any name you like at any time.

WebRTC (Beta)

User can send video/audio from web browser to OvenMediaEngine via WebRTC without plug-in. Of course, you can use any encoder that supports WebRTC transmission as well as a browser.

Configuration

Bind

In order for OvenMediaEngine to receive streams through WebRTC, web socket-based signaling port and ICE candidate must be set. The ICE candidate can configure a TCP relay. WebRTC provider and WebRTC publisher can use the same port. Ports of WebRTC provider can be set in as follows.

```
<Bind>
        <Providers>
                ...
                <WebRTC>
                        <Signalling>
                                <Port>3333</Port>
                        </Signalling>
                        <IceCandidates>
                                <TcpRelay>*:3478</TcpRelay>
                                <IceCandidate>*:10006-10010/udp</IceCandidate>
                        </IceCandidates>
                </WebRTC>
        </Providers>
```

Application

WebRTC input can be turned on/off for each application. As follows Setting enables the WebRTC input function of the application.

```
<Applications>
        <Application>
                <Name>app</Name>
                <Providers>
                        <WebRTC />
```

URL Pattern

The signaling url for WebRTC input uses the query string**(?direction=send)** as follows to distinguish it from the url for WebRTC playback.

> ws[s]://<host>[:port]/<app name>/<stream name>**?direction=send**

When WebRTC transmission starts, a stream is created with under application.

WebRTC over TCP

WebRTC transmission is sensitive to packet loss because it affects all players who access the stream. Therefore, it is recommended to provide WebRTC transmission over TCP. OvenMediaEngine has a built-in TURN server for WebRTC/TCP, and receives or transmits streams using the TCP session that the player's TURN client connects to the TURN server as it is. To use WebRTC/TCP, use transport=tcp query string as in WebRTC playback. See WebRTC/tcp playback for more information.

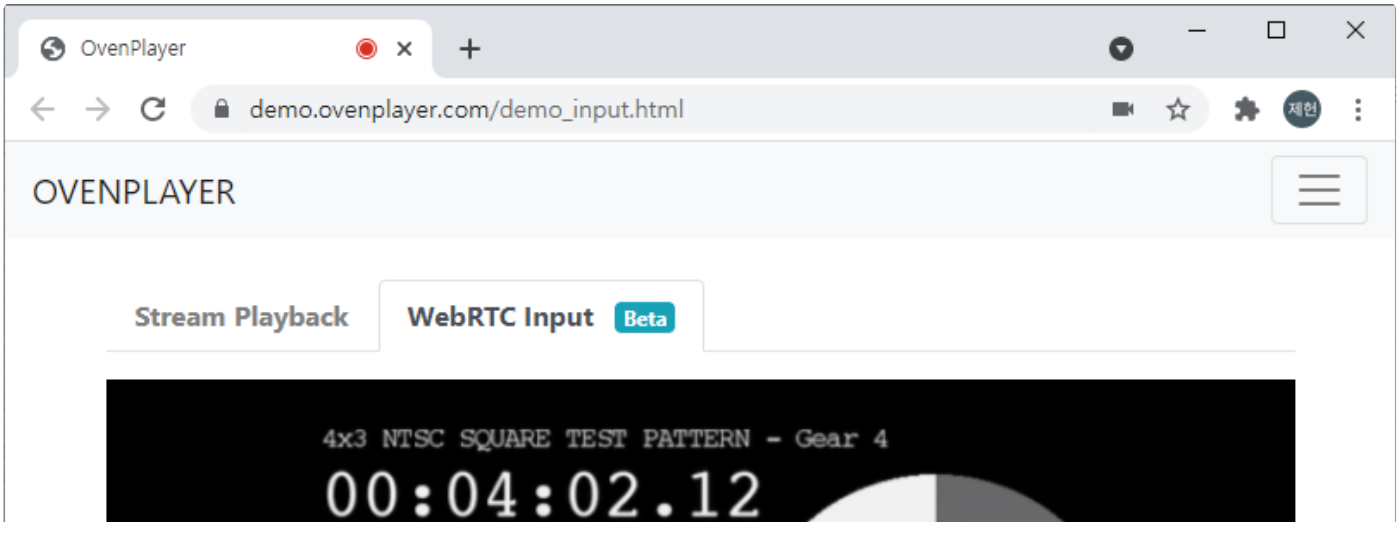
> ws[s]://<host>[:port]/<app name>/<stream name>**?direction=send&transport=tcp**

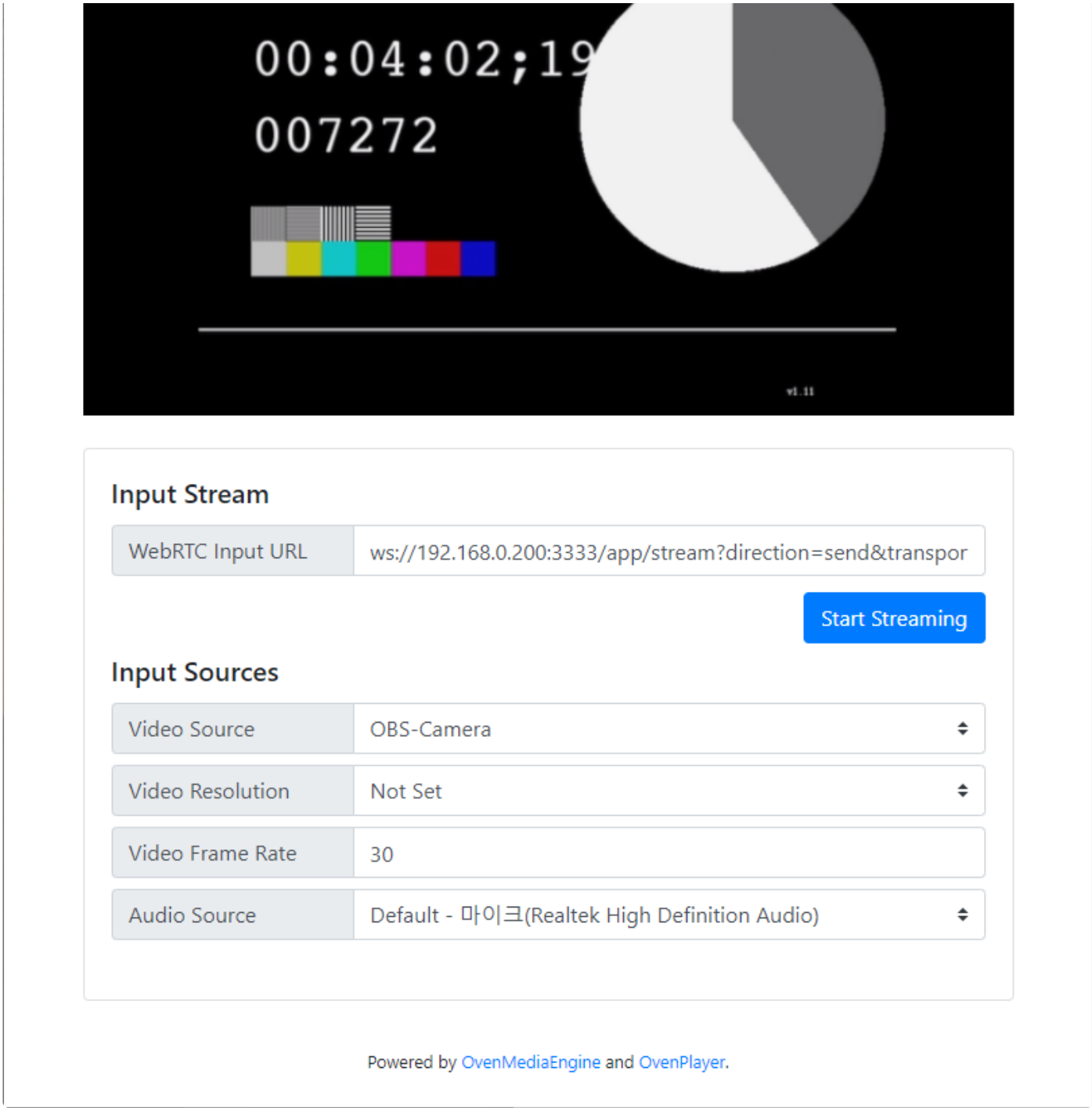
> ⓘ To use WebRTC/tcp, **<**TcpRelay> must be turned on in <Bind> setting.

WebRTC Producer

We provide a demo page so you can easily test your WebRTC input. You can access the demo page at the URL below.

OvenPlayer

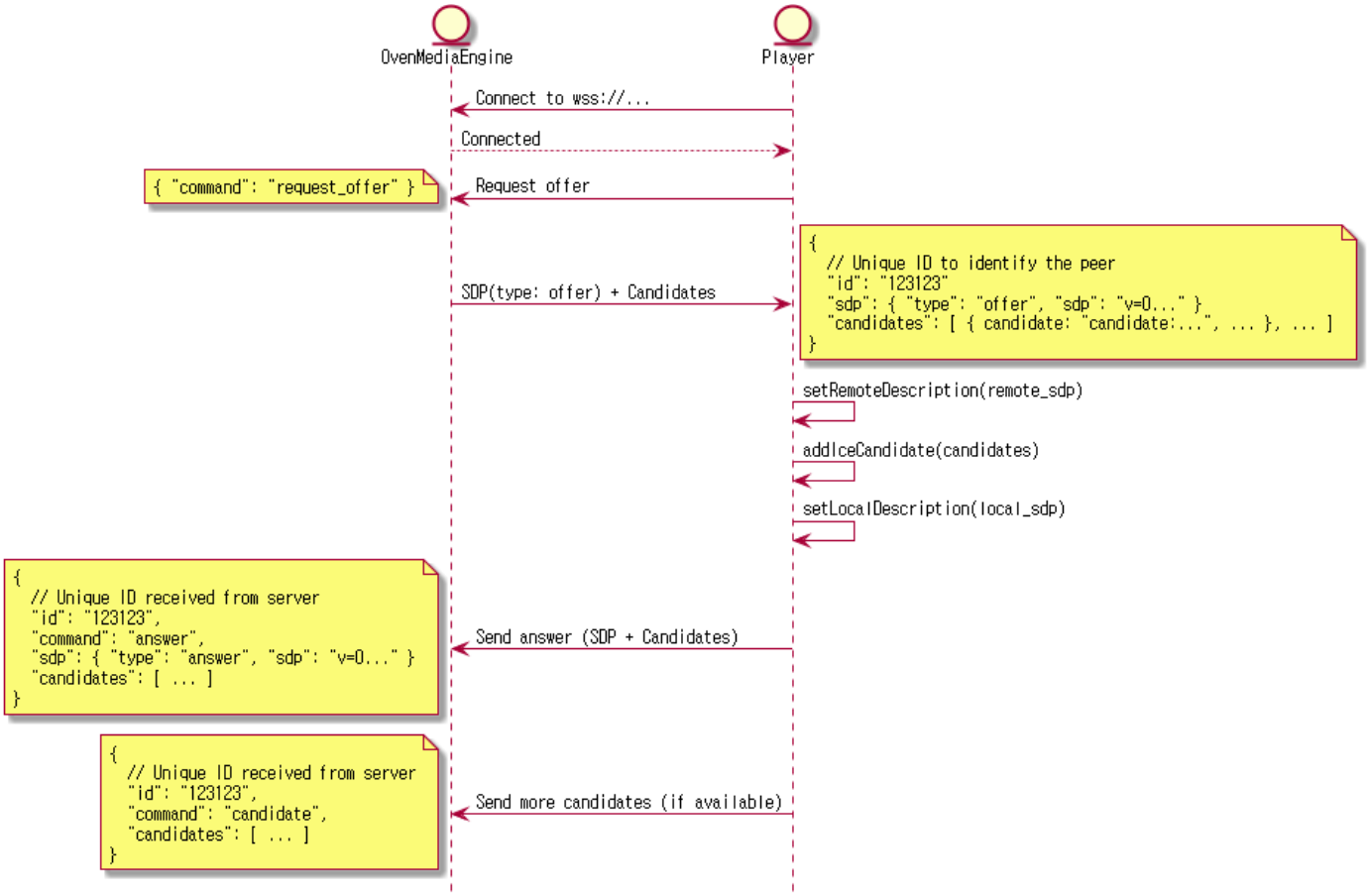




The getUserMedia API to access the local device only works in a secure context. So, the WebRTC Input demo page can only work on the https site **** **https**://demo.ovenplayer.com/demo_input.html. This means that due to mixed content you have to install the certificate in OvenMediaEngine and use the signaling URL as wss to test this. If you can't install the certificate in OvenMediaEngine, you can temporarily test it by allowing the insecure content of the demo.ovenplayer.com URL in your browser.

Custom WebRTC Producer

To create a custom WebRTC Producer, you need to implement OvenMediaEngine's Signaling Protocol. The protocol is structured in a simple format and uses the same method as WebRTC Streaming.



When the player connects to ws[s]://host:port/app/stream?**direction=send** through a web socket and sends a request offer command, the server responds to the offer sdp. If transport=tcp exists in the query string of the URL, iceServers information is included in offer sdp, which contains the information of OvenMediaEngine's built-in TURN server, so you need to set this in RTCPeerConnection to use WebRTC/TCP. The player then setsRemoteDescription and addIceCandidate offer sdp, generates an answer sdp, and responds to the server.

SRT (Beta)

Secure Reliable Transport (or SRT in short) is an open source video transport protocol and technology stack that optimizes streaming performance across unpredictable networks with secure streams and easy firewall traversal, bringing the best quality live video over the worst networks. We consider SRT to be one of the great alternatives to RTMP, and OvenMediaEngine can receive video streaming over SRT. For more information on SRT, please visit the SRT Alliance website.

SRT uses the MPEG-TS format when transmitting live streams. This means that unlike RTMP, it can support many codecs. Currently, OvenMediaEngine supports H.264, H.265, and AAC codecs received by SRT.

Configuration

Bind

Set the SRT listen port as follows:

```
<Bind>
        <Providers>
                ...
                <SRT>
                        <Port>9999</Port>
                        <!-- <WorkerCount>1</WorkerCount> -->
                </SRT>
        </Providers>
```

Application

SRT input can be turned on/off for each application. As follows Setting enables the SRT input function of the application.

```
<Applications>
        <Application>
                <Name>app</Name>
                <Providers>
                        <SRT/>
```

Encoders and streamid

There are various encoders that support SRT such as FFMPEG, OBS Studio, and srt-live-transmit. Please check the specifications of each encoder on how to transmit streams through SRT from the encoder. We describe an example using OBS Studio.

OvenMediaEngine classifies each stream using SRT's streamid. This means that unlike MEPG-TS/udp, OvenMediaEngine can receive multiple SRT streams through one port. For more information on streamid, see Haivision's official documentation.

Therefore, in order for the SRT encoder to transmit a stream to OvenMediaEngine, the following information must be included in the streamid as percent encoded.

> streamid = percent_encoding("srt://{host}[:port]/{app name}/{stream name}[?query=value]")

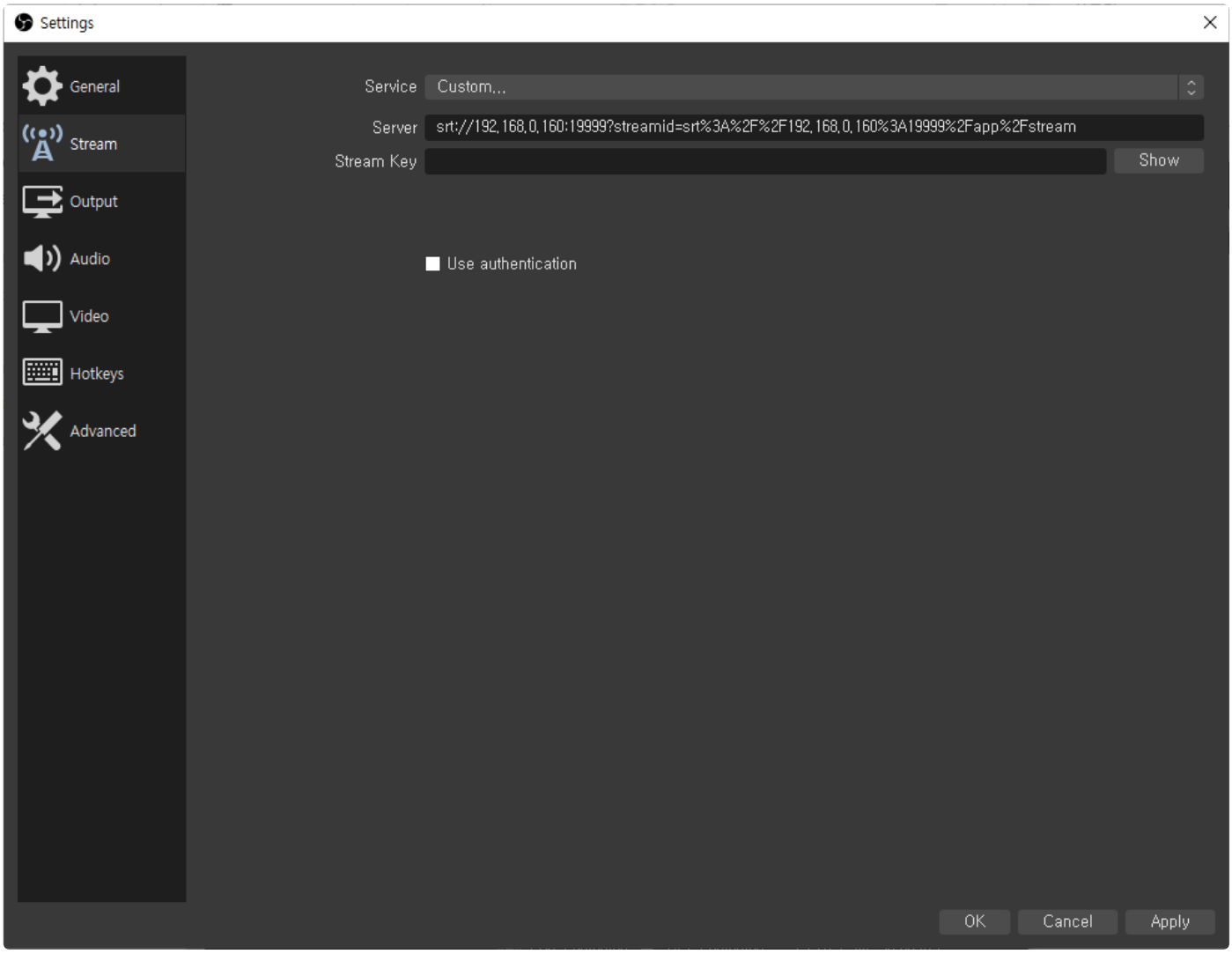
> ⓘ The **streamid** contains the URL format, so it must be **percent encoded******

OBS Studio

OBS Studio 25.0 or later supports SRT. Please refer to the OBS official documentation for more information. Enter the address of OvenMediaEngine in OBS Studio's Server as follows: When using SRT in OBS, you

can leave the Stream Key blank.

`srt://ip:port?streamid=srt%3A%2F%2F{domain or IP address}[%3APort]%2F{App name}%2F{Stream name}`



MPEG-2 TS

From version 0.10.4, MPEG-2 TS input is supported as a beta version. The supported codecs are H.264, AAC(ADTS). Supported codecs will continue to be added. And the current version only supports basic MPEG-2 TS with 188 bytes packet size. Since the information about the input stream is obtained using PAT and PMT, the client must send this table information as required.

> ⓘ This version supports MPEG-2 TS over UDP. MPEG-2 TS over TCP or MPEG-2 TS over SRT will be supported soon.

Configuration

To enable MPEG-2 TS, you must bind the ports fist and map the bound ports and streams.

Bind

To use multiple streams, it is necessary to bind multiple ports, so we provide a way to bind multiple ports as in the example below. You can use the dash to specify the port as a range, such as `Start port-End port`, and multiple ports using commas.

Stream mapping

First, name the stream and map the port bound above. The macro ${Port} is provided to map multiple streams at once. Check out the example below.

```
<Server>
        ...
        <Bind>
                <Providers>
                        <MPEGTS>
                                <!--
                                        Listen on port 4000,4001,4004,4005
                                        This is just a demonstration to show that you can con
                                -->
                                <Port>4000-4001,4004,4005/udp</Port>
                        </MPEGTS>
                </Providers>
        </Bind>
        ...
        <VirtualHosts>
                <VirtualHost>
                        <Application>
                                <Providers>
                                        <MPEGTS>
                                                        <StreamMap>
                                                                <!--
                                                                        Set the stream name o
                                                                        For example, if a cli
                                                                -->
                                                                <Stream>
                                                                        <Name>stream_${Port}<
                                                                        <Port>4000-4001,4004<
                                                                </Stream>
                                                                <Stream>
                                                                        <Name>stream_name_for
                                                                        <Port>4005</Port>
                                                                </Stream>
                                                        </StreamMap>
                                                </MPEGTS>
                                </Providers>
                        <Application>
                </VirtualHost>
```

```
</Server></VirtualHosts>
```

Publish

This is an example of publishing using FFMPEG.

```
# Video / Audio
ffmpeg.exe -re -stream_loop -1 -i <file.ext> -c:v libx264 -bf 0 -x264-params keyint=30:scenecu
# Video only
ffmpeg.exe -re -stream_loop -1 -i <file.ext> -c:v libx264 -bf 0 -x264-params keyint=30:scenecu
# Audio only
ffmpeg.exe -re -stream_loop -1 -i <file.ext> -vn  -acodec aac -pes_payload_size 0 -f mpegts uc
```

> ⓘ Giving the -pes_payload_size 0 option to the AAC codec is very important for AV synchronization and low latency. If this option is not given, FFMPEG bundles several ADTSs and is transmitted at once, which may cause high latency and AV synchronization errors.

RTSP Pull

From version 0.10.4, RTSP Pull input is supported as a beta version. The supported codecs are H.264, AAC(ADTS). Supported codecs will continue to be added.

This function pulls a stream from an external RTSP server and operates as an RTSP client.

Configuration

RTSP Pull is provided through OriginMap configuration. OriginMap is the rule that the Edge server pulls the stream of the Origin server. Edge server can pull a stream of origin with RTSP and OVT (protocol defined by OvenMediaEngine for Origin-Edge) protocol. See the Clustering section for more information about OVT.

```
<VirtualHosts>
                <VirtualHost include="VHost*.xml" />
                <VirtualHost>
                        <Name>default</Name>

                        <Host>
                                <Names>
                                        <!-- Host names
                                                <Name>stream1.airensoft.com</Name>
                                                <Name>stream2.airensoft.com</Name>
                                                <Name>*.sub.airensoft.com</Name>
                                                <Name>192.168.0.1</Name>
```

```
                        <Name>*</Name>
                    </Names>
                    <!--
                    <TLS>
                        <CertPath>path/to/file.crt</CertPath>
                        <KeyPath>path/to/file.key</KeyPath>
                        <ChainCertPath>path/to/file.crt</ChainCertPath>
                    </TLS>
                    -->
                </Host>

                <Origins>
                    <Origin>
                        <Location>/app_name/rtsp_stream_name</Location>
                        <Pass>
                            <Scheme>rtsp</Scheme>
                            <Urls><Url>192.168.0.200:554/</Url></Urls>
                        </Pass>
                    </Origin>
                </Origins>
            </VirtualHost>
        </VirtualHosts>
```

For example, in the above setup, when a player requests "ws://ome.com/**app_name/rtsp_stream_name"** to stream WebRTC, it pulls the stream from "rtsp://**192.168.0.200:554"** and publishes it to WebRTC.

> If the app name set in Location isn't created, OvenMediaEngine creates the app with default settings. The default generated app doesn't have an OPUS encoding profile, so to use WebRTC streaming, you need to add the app to your configuration.

Publish

The pull-type provider is activated by the publisher's streaming request. And if there is no client playing for 30 seconds, the provider is automatically disabled.

According to the above setting, the RTSP pull provider operates for the following streaming URLs.

Protocol	URL
WebRTC	ws::///ome.com:3333/app_name/rtsp_stream_nam
HLS	http://ome.com:8080/app_name/rtsp_stream_nam playlist.m3u8
DASH	http://ome.com:8080/app_name/rtsp_stream_nam manifest.mpd
LL DASH	http://ome.com:8080/app_name/rtsp_stream_nam manifest_ll.mpd

ABR and Transcoding

OvenMediaEngine has a built-in live transcoder. The live transcoder can decode the incoming live source and re-encode it with the set codec or adjust the quality to encode at multiple bitrates.

Supported Video, Audio and Image Codecs

OvenMediaEngine currently supports the following codecs:

Type	Decoder	Encoder
Video	VP8, H.264, H.265	VP8, H.264, H.265(GPU only)
Audio	AAC, OPUS	AAC, OPUS
Image		JPEG, PNG

OutputProfiles

OutputProfile

The `<OutputProfile>` setting allows incoming streams to be re-encoded via the `<Encodes>` setting to create a new output stream. The name of the new output stream is determined by the rules set in `<OutputStreamName>`, and the newly created stream can be used according to the streaming URL format.

```
<OutputProfiles>
    <OutputProfile>
        <Name>bypass_stream</Name>
        <OutputStreamName>${OriginStreamName}_bypass</OutputStreamName>
        <Encodes>
            <Audio>
                <Bypass>true</Bypass>
            </Audio>
            <Video>
                <Bypass>true</Bypass>
            </Video>
```

```
            <Audio>
                <Codec>opus</Codec>
                <Bitrate>128000</Bitrate>
                <Samplerate>48000</Samplerate>
                <Channel>2</Channel>
            </Audio>
        </Encodes>
    </OutputProfile>
</OutputProfiles>
```

According to the above setting, if the incoming stream name is `stream`, the output stream becomes `stream_bypass` and the stream URL can be used as follows.

- **`WebRTC`** ws://192.168.0.1:3333/app/stream_bypass
- **`LLHLS`** http://192.168.0.1:8080/app/stream_bypass/llhls.m3u8

Encodes

Video

You can set the video profile as below:

```
<Encodes>
    <Video>
        <Name>720_vp8</Name>
        <Codec>vp8</Codec>
        <Width>1280</Width>
        <Height>720</Height>
        <Bitrate>2000000</Bitrate>
        <Framerate>30.0</Framerate>
        <Preset>fast</Preset>
        <ThreadCount>4</ThreadCount>
    </Video>
</Encodes>
```

The meaning of each property is as follows:

Property	Description
Codec*	Specifies the `vp8` or `h264` codec to use
Bitrate*	Bit per second
Name	Encode name for Renditions
Width	Width of resolution
Height	Height of resolution
Framerate	Frames per second

Preset	Presets of encoding quality and performance
ThreadCount	Number of threads in encoding

Table of presets

A table in which presets provided for each codec library are mapped to OvenMediaEngine's presets. Slow presets are of good quality and use a lot of resources, whereas Fast presets have lower quality and better performance. It can be set according to your own system environment and service purpose.

Presets	openh264	libvpx	h264/265 NVC	h264/265 QSV
slower	Quantizer(10-41)	best	hq	-
slow	Quantizer(10-41)	best	llhq	-
medium	Quantizer(10-51)	good	bd	-
fast	Quantizer(25-51)	realtime	hp	-
faster	Quantizer(25-51)	*realtime	*llhp	-

References

- https://trac.ffmpeg.org/wiki/Encode/VP8
- https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-preset-migration-guide/

Audio

You can set the audio profile as below:

```
<Encodes>
    <Audio>
        <Name>opus_128</Name>
        <Codec>opus</Codec>
        <Bitrate>128000</Bitrate>
        <Samplerate>48000</Samplerate>
        <Channel>2</Channel>
    </Audio>
</Encodes>
```

The meaning of each property is as follows:

Property	Description
Codec*	Specifies the `opus` or `aac` codec to use

Bitrate* Name	Bits per second Encode name for Renditions
Samplerate	Samples per second
Channel	The number of audio channels

It is possible to have an audio only output profile by specifying the Audio profile and omitting a Video one.

Image

You can set the Image profile as below:

```
<Encodes>
    <Image>
        <Codec>jpeg</Codec>
        <Width>1280</Width>
        <Height>720</Height>
        <Framerate>1</Framerate>
    </Image>
</Encodes>
```

The meaning of each property is as follows:

Property	Description
Codec	Specifies the `jpeg` or `png` codec to use
Width	Width of resolution
Height	Height of resolution
Framerate	Frames per second

> The image encoding profile is only used by thumbnail publishers. and, bypass option is not supported.

Bypass without transcoding

You can configure Video and Audio to bypass transcoding as follows:

```
<Video>
    <Bypass>true</Bypass>
</Video>
<Audio>
    <Bypass>true</Bypass>
</Audio>
```

> ⓘ You need to consider codec compatibility with some browsers. For example, chrome only supports OPUS codec for audio to play WebRTC stream. If you set to bypass incoming audio, it can't play on chrome.

WebRTC doesn't support AAC, so if video bypasses transcoding, audio must be encoded in OPUS.

```
<Encodes>
    <Video>
        <Bypass>true</Bypass>
    </Video>
    <Audio>
        <Codec>opus</Codec>
        <Bitrate>128000</Bitrate>
        <Samplerate>48000</Samplerate>
        <Channel>2</Channel>
    </Audio>
</Encodes>
```

Keep the original with transcoding

If you want to transcode with the same quality as the original. See the sample below for possible parameters that OME supports to keep original. If you remove the **Width**, **Height**, **Framerate**, **Samplerate**, and **Channel** parameters. then, It is transcoded with the same options as the original.

```
<Encodes>
    <Video>
        <Codec>vp8</Codec>
        <Bitrate>2000000</Bitrate>
    </Video>
     <Audio>
        <Codec>opus</Codec>
        <Bitrate>128000</Bitrate>
    </Audio>
    <Image>
        <Codec>jpeg</Codec>
    </Image>
</Encodes>
```

Rescaling while keep the aspect ratio

To change the video resolution when transcoding, use the values of width and height in the Video encode option. If you don't know the resolution of the original, it will be difficult to keep the aspect ratio after transcoding. Please use the following methods to solve these problems. For example, if you input only the **Width** value in the Video encoding option, the **Height** value is automatically generated according to the ratio of the original video.

```
<Encodes>
    <Video>
        <Codec>h264</Codec>
        <Bitrate>2000000</Bitrate>
        <Width>1280</Width>
        <!-- Height is automatically calculated as the original video ratio -->
        <Framerate>30.0</Framerate>
    </Video>
    <Video>
        <Codec>h264</Codec>
        <Bitrate>2000000</Bitrate>
        <!-- Width is automatically calculated as the original video ratio -->
        <Height>720</Height>
        <Framerate>30.0</Framerate>
    </Video>
</Encodes>
```

Adaptive Bitrates Streaming (ABR)

From version 0.14.0, OvenMediaEngine can encode same source with multiple bitrates renditions and deliver it to the player.

As shown in the example configuration below, you can provide ABR by adding `<Playlists>` to `<OutputProfile>`. There can be multiple playlists, and each playlist can be accessed with `<FileName>`.

The method to access the playlist set through LLHLS is as follows.

`http[s]://<domain>[:port]/<app>/<stream>/` **`<FileName>`** `.m3u8`

The method to access the Playlist set through WebRTC is as follows.

`ws[s]://<domain>[:port]/<app>/<stream>/` **`<FileName>`**

Note that `<FileName>` must never contain the **`playlist`** and **`chunklist`** keywords. This is a reserved word used inside the system.

To set up `<Rendition>`, you need to add `<Name>` to the elements of `<Encodes>`. Connect the set `<Name>` into `<Rendition><Video>` or `<Rendition><Audio>`.

In the example below, three quality renditions are provided and the URL to play the `abr` playlist as LLHLS is `https://domain:port/app/stream/abr.m3u8` and The WebRTC playback URL is `wss://domain:port/app/stream/abr`

```
<OutputProfile>
        <Name>bypass_stream</Name>
        <OutputStreamName>${OriginStreamName}</OutputStreamName>
```

```
<!--LLHLS URL : https://domain/app/stream/abr.m3u8 -->
<Playlist>
        <Name>For LLHLS</Name>
        <FileName>abr</FileName>
        <Options> <!-- Optinal -->
                <!--
                Automatically switch rendition in WebRTC ABR
                [Default] : true
                -->
                <WebRtcAutoAbr>true</WebRtcAutoAbr>

        </Options>
        <Rendition>
                <Name>Bypass</Name>
                <Video>bypass_video</Video>
                <Audio>bypass_audio</Audio>
        </Rendition>
        <Rendition>
                <Name>FHD</Name>
                <Video>video_1280</Video>
                <Audio>bypass_audio</Audio>
        </Rendition>
        <Rendition>
                <Name>HD</Name>
                <Video>video_720</Video>
                <Audio>bypass_audio</Audio>
        </Rendition>
</Playlist>
<!--LLHLS URL : https://domain/app/stream/llhls.m3u8 -->
<Playlist>
        <Name>Change Default</Name>
        <FileName>llhls</FileName>
        <Rendition>
                <Name>HD</Name>
                <Video>video_720</Video>
                <Audio>bypass_audio</Audio>
        </Rendition>
</Playlist>
<Encodes>
        <Audio>
                <Name>bypass_audio</Name>
                <Bypass>true</Bypass>
        </Audio>
        <Video>
                <Name>bypass_video</Name>
                <Bypass>true</Bypass>
        </Video>
        <Audio>
                <Codec>opus</Codec>
                <Bitrate>128000</Bitrate>
                <Samplerate>48000</Samplerate>
                <Channel>2</Channel>
        </Audio>
        <Video>
```

```
                        <Name>video_1280</Name>
                        <Codec>h264</Codec>
                        <Bitrate>5024000</Bitrate>
                        <Framerate>30</Framerate>
                        <Width>1920</Width>
                        <Height>1280</Height>
                        <Preset>faster</Preset>
                </Video>
                <Video>
                        <Name>video_720</Name>

                        <Codec>h264</Codec>
                        <Bitrate>2024000</Bitrate>
                        <Framerate>30</Framerate>
                        <Width>1280</Width>
                        <Height>720</Height>
                        <Preset>faster</Preset>
                </Video>
        </Encodes>
</OutputProfile>
```

Supported codecs by streaming protocol

Even if you set up multiple codecs, there is a codec that matches each streaming protocol supported by OME, so it can automatically select and stream codecs that match the protocol. However, if you don't set a codec that matches the streaming protocol you want to use, it won't be streamed.

The following is a list of codecs that match each streaming protocol:

Protocol	Supported Codec
WebRTC	VP8, H.264, Opus
LLHLS	H.264, AAC

Therefore, you set it up as shown in the table. If you want to stream using LLHLS, you need to set up H.264 and AAC, and if you want to stream using WebRTC, you need to set up Opus.

Also, if you are going to use WebRTC on all platforms, you need to configure both VP8 and H.264. This is because different codecs are supported for each browser, for example, VP8 only, H264 only, or both.

However, don't worry. If you set the codecs correctly, OME automatically sends the stream of codecs requested by the browser.

Enable GPU Acceleration

OvenMediaEngine supports GPU-based hardware decoding and encoding. Currently supported GPU acceleration devices are Intel's QuickSync and NVIDIA's NVDECODE/NVENCODE. This document describes how to install the video driver for OvenMediaEngine to use the GPU and how to set the Config file.
Please check what graphics card you have and refer to the NVIDIA or Intel driver installation guide.

Reference

- Quick Sync Video format support: https://en.wikipedia.org/wiki/Intel_Quick_Sync_Video

---

Install GPU Driver

Intel QuickSync Driver

If you are using an Intel CPU that supports QuickSync, please refer to the following guide to install the driver. The OSes that support installation using the provided scripts are **CentOS 7/8** and **Ubuntu 18/20** versions. If you want to install the driver on a different OS, please refer to the Manual Installation Guide document.

When the Intel QuickSync driver installation is complete, the OS must be rebooted for normal operation.

```
(curl -LOJ https://github.com/AirenSoft/OvenMediaEngine/archive/master.tar.gz && tar xvfz Ove
OvenMediaEngine-master/misc/install_intel_driver.sh
```

How to check driver installation

After the driver installation is complete, check whether the driver operates normally with the Matrix Monitor program.

```
# Use the samples provided in the Intel Media SDK
# Check the list of codecs supported by iGPU
/MediaSDK-intel-mediasdk-21.1.2/build/__bin/release/simple_7_codec
```

NVIDIA GPU Driver

If you are using an NVIDIA graphics card, please refer to the following guide to install the driver. The OS that supports installation with the provided script are **CentOS 7/8** and **Ubuntu 18/20** versions. If you want to install the driver in another OS, please refer to the manual installation guide document.

CentOS environment requires the process of uninstalling the nouveau driver. After uninstalling the driver, the first reboot is required, and a new NVIDIA driver must be installed and rebooted. Therefore, two install scripts must be executed.

```
(curl -LOJ https://github.com/AirenSoft/OvenMediaEngine/archive/master.tar.gz && tar xvfz Ove
OvenMediaEngine-master/misc/install_nvidia_driver.sh
```

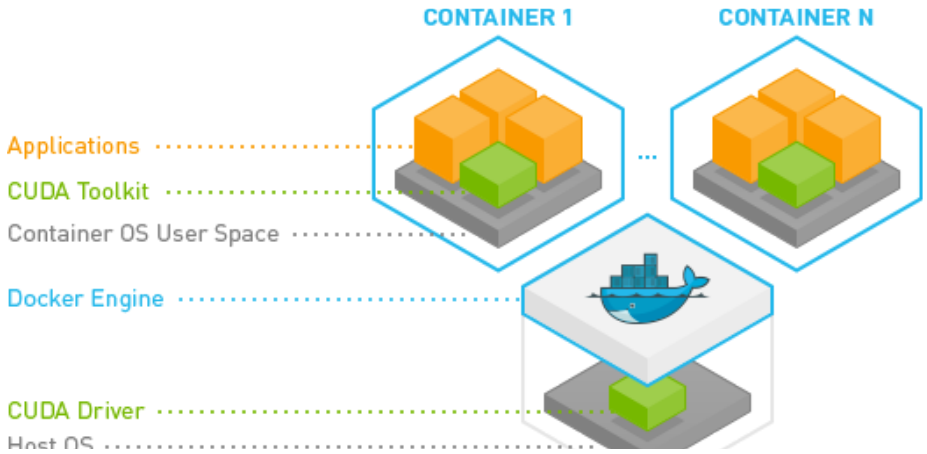
How to check driver installation

After the driver installation is complete, check whether the driver is operating normally with the nvidia-smi command.

```
$ nvidia-smi

Thu Jun 17 10:20:23 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.19.01    Driver Version: 465.19.01    CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:01:00.0 Off |                  N/A |
| 20%   35C    P8    N/A /  75W |    156MiB /  1997MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1589      G   /usr/libexec/Xorg                  38MiB |
|    0   N/A  N/A      1730      G   /usr/bin/gnome-shell              115MiB |
+-----------------------------------------------------------------------------+
```

Container Toolkit for Docker

Describes how to enable GPU acceleration for users running OvenMediaEngine in the Docker runtime environment. To use GPU acceleration in Docker, the NVIDIA Driver must be installed on the host OS and the NVIDIA Container Toolkit must be installed. This toolkit includes container runtime libraries and utilities to use NVIDIA GPUs in Docker containers.



Reference : https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/overview.html

```
OvenMediaEngine-master/misc/install_nvidia_docker_container.sh
```

> ⓘ The NVIDIA Driver must have been previously installed

How to run docker

To use GPU when running Docker, you need to add the **--gpus all** option.

```
docker run -d \
-p 1935:1935 -p 4000-4005:4000-4005/udp -p 3333:3333 -p 3478:3478 -p 8080:8080 -p 9000:9000 -p
--gpus all
airensoft/ovenmediaengine:dev
```

Manual Installation

If the provided installation script fails, please refer to the manual installation guide.

Manual Installation

Prerequisites Additional Options

If you have finished installing the driver to use the GPU, you need to reinstall the open source library using Prerequisites.sh . The purpose is to allow external libraries to use the installed graphics driver.

When using NVIDIA Driver

```
OvenMediaEngine-master/misc/prerequisites.sh --enable-nvc
```

When using Intel QuickSync Driver

```
OvenMediaEngine-master/misc/prerequisites.sh --enable-qsv
```

When using Docker container

```
docker build --file Dockerfile -t airensoft/ovenmediaengine:dev --build-arg GPU=TRU
```

Configuration

To use hardware acceleration, set the **HardwareAcceleration** option to **true** under OutputProfiles. If this option is enabled, a hardware codec is automatically used when creating a stream, and if it is unavailable due to insufficient hardware resources, it is replaced with a software codec.

```
<VirtualHosts>
   <VirtualHost>
      <Name>default</Name>

      <!-- Settings for multi domain and TLS -->
      <Host>
         ...
      </Host>

      <!-- Settings for applications -->
      <Applications>
         <Application>
            <Name>app</Name>
            <Type>live</Type>
            <OutputProfiles>
               <!-- Settings to use hardware codecs -->
               <HardwareAcceleration>true</HardwareAcceleration>
               <OutputProfile>
                  <Name>bypass_stream</Name>
                  <OutputStreamName>${OriginStreamName}_o</OutputStreamName>
                  <Encodes>
                     <Video>
                        <Codec>h264</Codec>
                        <Width>1920</Width>
                        <Height>1080</Height>
                        <Bitrate>5000000</Bitrate>
                        <Framerate>30</Framerate>
                     </Video>

                     <Video>
                        <Codec>h265</Codec>
                        <Width>1280</Width>
                        <Height>720</Height>
                        <Bitrate>5000000</Bitrate>
```

```
                    <Framerate>30</Framerate>
                </Video>
            </Encodes>
        </OutputProfile>


    </OutputProfiles>
    <Providers>
        ...
    </Providers>
    <Publishers>
        ...
    </Publishers>
</Application>
</Applications>
</VirtualHost>
</VirtualHosts>
```

Configuration

Build & Run

You can build the OvenMediaEngine source using the following command. Same as the contents of **Getting Started**.

Ubuntu 18+

```
sudo apt-get update
cd OvenMediaEngine-master/src
make release
sudo make install
systemctl start ovenmediaengine
# If you want automatically start on boot
systemctl enable ovenmediaengine.service
```

Fedora 28+

```
sudo dnf update
cd OvenMediaEngine-master/src
make release
sudo make install
systemctl start ovenmediaengine
```

```
# If you want automatically start on boot
systemctl enable ovenmediaengine.service
```

CentOS 7+

```
sudo yum update
source scl_source enable devtoolset-7
cd OvenMediaEngine-master/src
make release
sudo make install
systemctl start ovenmediaengine
# If you want automatically start on boot
systemctl enable ovenmediaengine.service
```

 Getting Started

---

Support Format

The codecs available using hardware accelerators in OvenMediaEngine are as shown in the table below. Different GPUs support different codecs. If the hardware codec is not available, you should check if your GPU device supports the codec.

Device	H264	H265	VP8	VP9
QuickSync	D / E	D / E	-	-
NVIDIA	D / E	D / E	-	-
Docker on NVIDIA Container Toolkit	D / E	D / E	-	-

D : Decoding, E : Encoding

Reference

- Quick Sync Video Format : https://en.wikipedia.org/wiki/Intel_Quick_Sync_Video
- NVIDIA NVDEC Video Format : https://en.wikipedia.org/wiki/Nvidia_NVDEC

- NVIDIA NVENV Video Format : https://en.wikipedia.org/wiki/Nvidia_NVENC
- CUDA Toolkit Installation Guide : https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#introduction
- NVIDIA Container Toolkit : https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/arch-overview.html#arch-overview

---

Manual Installation

Intel QuickSync Driver Installation

Platform Specific Installation

Ubuntu 18/20

```
sudo apt install -y libdrm-dev xorg xorg-dev openbox libx11-dev
sudo apt install -y libgl1-mesa-glx libgl1-mesa-dev
```

CentOS 7

```
# Centos 7 uses the 2.8.x version of cmake by default.
# It must be changed to version 3.x or higher.
sudo yum remove -y cmake
sudo yum install -y cmake3
sudo ln -s /usr/bin/cmake3 /usr/bin/cmake

sudo yum install -y libdrm-devel libX11-devel libXi-devel
```

CentOS 8

```
sudo yum install -y libdrm-devel libX11-devel libXi-devel
```

Common Installation

LibVA

```
PREFIX=/opt/ovenmediaengine && \
TEMP_PATH=/tmp && \
LIBVA_VERSION=2.11.0 && \
DIR=${TEMP_PATH}/libva && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/intel/libva/archive/refs/tags/${LIBVA_VERSION}.tar.gz | tar -xz -
./autogen.sh --prefix="${PREFIX}" && \
make -j$(nproc) && \
sudo make install && \
rm -rf ${DIR})
```

GMMLIB

```
PREFIX=/opt/ovenmediaengine && \
TEMP_PATH=/tmp && \
GMMLIB_VERSION=20.4.1 && \
DIR=${TEMP_PATH}/gmmlib && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/intel/gmmlib/archive/refs/tags/intel-gmmlib-${GMMLIB_VERSION}.tar
mkdir -p ${DIR}/build && \
cd ${DIR}/build && \
cmake -DCMAKE_INSTALL_PREFIX="${PREFIX}" .. && \
make -j$(nproc) && \
sudo make install && \
rm -rf ${DIR}
```

Intel Media Driver

```
PREFIX=/opt/ovenmediaengine && \
TEMP_PATH=/tmp && \
INTEL_MEDIA_DRIVER_VERSION=20.4.5 && \
DIR_IMD=${TEMP_PATH}/media-driver && \
mkdir -p ${DIR_IMD} && \
cd ${DIR_IMD} && \
curl -sLf https://github.com/intel/media-driver/archive/refs/tags/intel-media-${INTEL_MEDIA_DI
DIR_GMMLIB=${TEMP_PATH}/gmmlib && \
mkdir -p ${DIR_GMMLIB} && \
cd ${DIR_GMMLIB} && \
curl -sLf https://github.com/intel/gmmlib/archive/refs/tags/intel-gmmlib-${GMMLIB_VERSION}.tar
DIR=${TEMP_PATH}/build && \
mkdir -p ${DIR} && \
cd ${DIR} && \
```

```
PKG_CONFIG_PATH=${PREFIX}/lib/pkgconfig:${PKG_CONFIG_PATH} cmake \
    $DIR_IMD \
    -DBUILD_TYPE=release \
    -DBS_DIR_GMMLIB="$DIR_GMMLIB/Source/GmmLib" \
    -DBS_DIR_COMMON=$DIR_GMMLIB/Source/Common \
    -DBS_DIR_INC=$DIR_GMMLIB/Source/inc \
    -DBS_DIR_MEDIA=$DIR_IMD \
    -DCMAKE_INSTALL_PREFIX=${PREFIX} \
    -DCMAKE_INSTALL_LIBDIR=${PREFIX}/lib \
    -DINSTALL_DRIVER_SYSCONF=OFF \
    -DLIBVA_DRIVERS_PATH=${PREFIX}/lib/dri && \
sudo make -j$(nproc) install && \
rm -rf ${DIR} && \
rm -rf ${DIR_IMD} && \
rm -rf ${DIR_GMMLIB}

export LIBVA_DRIVERS_PATH=${PREFIX}/lib
export LIBVA_DRIVER_NAME=iHD
```

Intel Media SDK

```
PREFIX=/opt/ovenmediaengine && \
TEMP_PATH=/tmp && \
INTEL_MEDIA_SDK_VERSION=20.5.1 && \
DIR=${TEMP_PATH}/medka-sdk && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/Intel-Media-SDK/MediaSDK/archive/refs/tags/intel-mediasdk-${INTE
mkdir -p ${DIR}/build && \
cd ${DIR}/build && \
PKG_CONFIG_PATH=${PREFIX}/lib/pkgconfig:${PKG_CONFIG_PATH} cmake -DCMAKE_INSTALL_PREFIX="${PR
make -j$(nproc) && \
sudo make install && \
rm -rf ${DIR}
```

> ⓘ All libraries are installed, the system must be rebooted.

NVIDIA Driver Installation

Platform Specific Installation

Ubuntu 18/20

```
# Uninstalling a previously installed NVIDIA Driver
sudo apt-get remove --purge nvidia-*
sudo apt-get -y autoremove
sudo apt-get -y update

# Remove the nouveau driver. If the nouveau driver is in use, the nvidia driver can
USE_NOUVEAU=`sudo lshw -class video | grep nouveau`
if [ ! -z "$USE_NOUVEAU" ]; then

        # Disable nouveau Driver
        echo "blacklist nouveau" >> /etc/modprobe.d/blacklist.conf
        echo "blacklist lbm-nouveau" >> /etc/modprobe.d/blacklist.conf
        echo "options nouveau modeset=0" >> /etc/modprobe.d/blacklist.conf
        echo "alias nouveau off" >> /etc/modprobe.d/blacklist.conf
        echo "alias lbm-nouveau off" >> /etc/modprobe.d/blacklist.conf
        sudo update-initramfs -u
        echo "Using a driver display nouveau.Remove the driver and reboot.Reboot an

        sleep 5s
        reboot
fi

# Install Nvidia Driver and Nvidia Toolkit
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt -y update
sudo apt-get install -y $(ubuntu-drivers devices | grep recommended | awk '{print $
sudo apt-get install -y nvidia-cuda-toolkit curl make
```

CentOS 7

```
# Update Kernel
yum -y update
yum -y groupinstall "Development Tools"
yum -y install kernel-devel
yum -y install epel-release
yum -y install dkms curl
echo "Reboot is required to run with a new version of the kernel."

# Remove the nouveau driver. If the nouveau driver is in use, the nvidia driver can
USE_NOUVEAU=`sudo lshw -class video | grep nouveau`
if [ ! -z "$USE_NOUVEAU" ]; then

        # Disable nouveau Driver
        sed "s/GRUB_CMDLINE_LINUX=\"\(.*\)\"/GRUB_CMDLINE_LINUX=\"\1 rd.driver.blac
        grub2-mkconfig -o /boot/grub2/grub.cfg
        echo "blacklist nouveau" >> /etc/modprobe.d/blacklist.conf
        mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
        dracut /boot/initramfs-$(uname -r).img $(uname -r)

        echo "Using a driver display nouveau. so, remove the driver and reboot. "
```

```
        echo "After reboot and installation script to rerun the nvidia display the

        sleep 5s
        reboot
fi

# Install Nvidia Driver
# https://www.nvidia.com/en-us/drivers/unix/
systemctl isolate multi-user.target
wget -N https://us.download.nvidia.com/XFree86/Linux-x86_64/460.84/NVIDIA-Linux-x86
sh ./NVIDIA-Linux-x86_64-460.84.run --ui=none --no-questions

# Install Nvidia Toolkit
# https://developer.nvidia.com/cuda-downloads
wget -N https://developer.download.nvidia.com/compute/cuda/11.3.1/local_installers/
sh cuda_11.3.1_465.19.01_linux.run --silent

# Configure Envionment Variables
echo "Please add the PATH below to the environment variable."
echo ""
echo "export PATH=${PATH}:/usr/local/cuda/bin/"
echo ""
export PATH=${PATH}:/usr/local/cuda/bin/
```

CentOS 8

```
dnf update -y
dnf groupinstall -y "Development Tools"
dnf install -y elfutils-libelf-devel "kernel-devel-uname-r == $(uname -r)"

# Remove the nouveau driver. If the nouveau driver is in use, the nvidia driver can
USE_NOUVEAU=`sudo lshw -class video | grep nouveau`
if [ ! -z "$USE_NOUVEAU" ]; then

        # Disable nouveau Driver
        echo "blacklist nouveau" >> /etc/modprobe.d/blacklist.conf
        mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-nouveau.img
        dracut /boot/initramfs-$(uname -r).img $(uname -r)

        systemctl set-default multi-user.target
        systemctl get-default

        echo "Using a driver display nouveau. so, remove the driver and reboot. "
        echo "After reboot and installation script to rerun the nvidia display the

        sleep 5s
        reboot
fi
```

```
wget https://us.download.nvidia.com/XFree86/Linux-x86_64/460.84/NVIDIA-Linux-x86
sh NVIDIA-Linux-x86_64-460.84.run --no-questions

# Install Nvidia Toolkit
# https://developer.nvidia.com/cuda-downloads
wget -N https://developer.download.nvidia.com/compute/cuda/11.3.1/local_installers/
sh cuda_11.3.1_465.19.01_linux.run --silent

systemctl set-default graphical.target
systemctl get-default

# Configure Envionment Variables
echo "Please add the PATH below to the environment variable."
echo ""
echo "export PATH=${PATH}:/usr/local/cuda/bin/"
echo ""
export PATH=${PATH}:/usr/local/cuda/bin/
```

Common Installation

NVCC Headers

```
PREFIX=/opt/ovenmediaengine && \
TEMP_PATH=/tmp && \
NVCC_HEADERS=11.0.10.1 && \
DIR=${TEMP_PATH}/nvcc-hdr && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/FFmpeg/nv-codec-headers/releases/download/n${NVCC_HEADERS}/nv-co
sudo make install && \
rm -rf ${DIR}
```

NVIDIA Container Toolkit Installation

Platform Specific Installation

Ubuntu 18/20

```
# Install Docker 2.x
curl https://get.docker.com && sudo systemctl --now enable docker

# Install NVIDIA Docker Toolkit
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) && \
```

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - && \
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list

sudo apt-get update
sudo apt-get install -y nvidia-docker2

# Restart Docker
sudo systemctl restart docker

# Test
sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

CentOS 7

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo=https://download.docker.com/linux/centos/docker-
sudo yum repolist -v
sudo yum install -y https://download.docker.com/linux/centos/7/x86_64/stable/Packag
sudo yum install docker-ce -y
sudo systemctl --now enable docker

# Install NVIDIA Docker Toolkit
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) && \
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.repo

if [[ "${OSVERSION}" == "7" ]]; then
    sudo yum clean expire-cache
    sudo yum install -y nvidia-docker2
elif [[ "${OSVERSION}" == "8" ]]; then
    sudo dnf clean expire-cache --refresh
    sudo dnf install -y nvidia-docker2
else
    fail_exit
fi

# Restart Docker
sudo systemctl restart docker

# Test
sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

CentOS 8

```
sudo dnf install -y dnf-utils
sudo dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-
sudo dnf repolist -v
```

```
sudo dnf install -y https://download.docker.com/linux/centos/7/x86_64/stable/Packag
sudo dnf install docker-ce -y
sudo systemctl --now enable docker

# Install NVIDIA Docker Toolkit
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) && \
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.repo

if [[ "${OSVERSION}" == "7" ]]; then
    sudo yum clean expire-cache
    sudo yum install -y nvidia-docker2
elif [[ "${OSVERSION}" == "8" ]]; then
    sudo dnf clean expire-cache --refresh
    sudo dnf install -y nvidia-docker2
else
    fail_exit
fi

# Restart Docker
sudo systemctl restart docker

# Test
sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

Streaming

WebRTC Streaming

OvenMediaEngine uses WebRTC to provide sub-second latency streaming. WebRTC uses RTP for media transmission and provides various extensions.

OvenMediaEngine provides the following features:

Title	Functions
Delivery	RTP / RTCP
Security	DTLS, SRTP
Connectivity	ICE
Error Correction	ULPFEC (VP8, H.264), In-band FEC (Opus)
Codec	VP8, H.264, Opus

Signalling	Self-Defined Signalling Protocol and Embedded Web Socket-Based Server

Configuration

If you want to use the WebRTC feature, you need to add `<WebRTC>` element to the `<Publishers>` and <Ports> in the `Server.xml` configuration file, as shown in the example below.

```
<Bind>
        <Publishers>
                <WebRTC>
                        <Signalling>
                                <Port>3333</Port>
                                <TLSPort>3334</TLSPort>
                                <WorkerCount>1</WorkerCount>
                        </Signalling>
                        <IceCandidates>
                                <IceCandidate>*:10000-10005/udp</IceCandidate>
                                <TcpRelay>*:3478</TcpRelay>
                                <TcpForce>true</TcpForce>
                                <TcpRelayWorkerCount>1</TcpRelayWorkerCount>
                        </IceCandidates>
                </WebRTC>
        </Publishers>
</Bind>
```

ICE

WebRTC uses ICE for connections and specifically NAT traversal. The web browser or player exchanges the Ice Candidate with each other in the Signalling phase. Therefore, OvenMediaEngine provides an ICE for WebRTC connectivity.

If you set IceCandidate to `*: 10000-10005/udp`, as in the example above, OvenMediaEngine automatically gets IP from the server and generates `IceCandidate` using UDP ports from 10000 to 10005. If you want to use a specific IP as IceCandidate, specify a specific IP. You can also use only one 10000 UDP Port, not a range, by setting it to *: 10000.

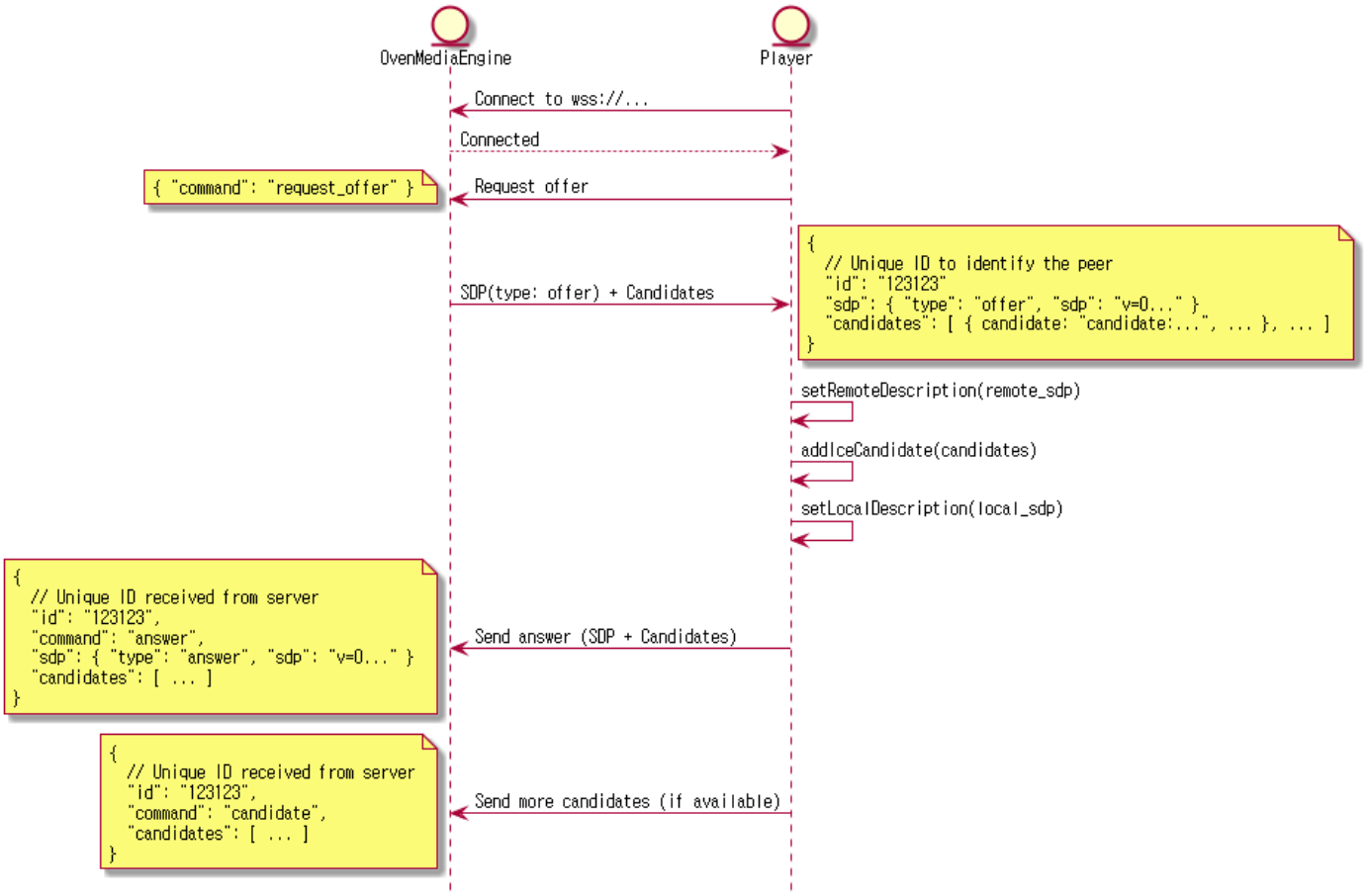
Signalling

OvenMediaEngine has embedded a WebSocket-based signalling server and provides our defined signalling protocol. Also, OvenPlayer supports our signalling protocol. WebRTC requires signalling to exchange Offer SDP and Answer SDP, but this part isn't standardized. If you want to use SDP, you need to create your exchange protocol yourself.

If you want to change the signaling port, change the value of `<Ports><WebRTC><Signalling>`.

Signalling Protocol

The Signalling protocol is defined in a simple way:



If you want to use a player other than OvenPlayer, you need to develop the signalling protocol as shown above and can integrate OvenMediaEngine.

Streaming

Publisher

Add `WebRTC` element to Publisher to provide streaming through WebRTC.

```
<Server version="7">
        <VirtualHosts>
                <VirtualHost>
                        <Applications>
                                <Application>
                                        <Publishers>
                                          <WebRTC>
                                                        <Timeout>30000</Timeout>
                                                        <Rtx>false</Rtx>
                                                        <Ulpfec>false</Ulpfec>
```

```
                                        </WebRTC><JitterBuffer>false</JitterBuffer>
                                    </Publishers>
                                </Application>
                            </Applications>
                        </VirtualHost>
                </VirtualHosts>
</Server>
```

Option	Description	Default
Timeout	ICE (STUN request/response) timeout as milliseconds, if there is no request or response during this time, the session is terminated.	30000
Rtx	WebRTC retransmission, a useful option in WebRTC/udp, but ineffective in WebRTC/tcp.	false
Ulpfec	WebRTC forward error correction, a useful option in WebRTC/udp, but ineffective in WebRTC/tcp.	false
JitterBuffer	Audio and video are interleaved and output evenly, see below for details	false

> WebRTC Publisher's `<JitterBuffer>` is a function that evenly outputs A/V (interleave) and is useful when A/V synchronization is no longer possible in the browser (player) as follows.
>
> - If the A/V sync is excessively out of sync, some browsers may not be able to handle this or it may take several seconds to synchronize.
> - Players that do not support RTCP also cannot A/V sync.

Encoding

WebRTC Streaming starts when a live source is inputted and a stream is created. Viewers can stream using OvenPlayer or players that have developed or applied the OvenMediaEngine Signalling protocol.

Also, the codecs supported by each browser are different, so you need to set the Transcoding profile according to the browser you want to support. For example, Safari for iOS supports H.264 but not VP8. If you want to support all browsers, please set up VP8, H.264, and Opus codecs in all transcoders.

WebRTC doesn't support AAC, so when trying to bypass transcoding RTMP input, audio must be encoded as opus. See the settings below.

```
<OutputProfiles>
    <OutputProfile>
        <Name>bypass_stream</Name>
        <OutputStreamName>${OriginStreamName}</OutputStreamName>
        <Encodes>
            <Audio>
                <Bypass>true</Bypass>
            </Audio>
            <Video>
                <Bypass>true</Bypass>
            </Video>
            <Video>
                <!-- vp8, h264 -->
                <Codec>vp8</Codec>
                <Width>1280</Width>
                <Height>720</Height>
                <Bitrate>2000000</Bitrate>
                <Framerate>30.0</Framerate>
            </Video>
            <Audio>
                <Codec>opus</Codec>
                <Bitrate>128000</Bitrate>
                <Samplerate>48000</Samplerate>
                <Channel>2</Channel>
            </Audio>
        </Encodes>
    </OutputProfile>
</OutputProfiles>
```

> Some browsers support both H.264 and VP8 to send Answer SDP to OvenMediaEngine, but sometimes H.264 can't be played. In this situation, if you write the VP8 above the H.264 code line in the Transcoding profile setting, you can increase the priority of the VP8.
>
> Using this manner so that some browsers, support H.264 but can't be played, can stream smoothly using VP8. This means that you can solve most problems with this method.

Playback

If you created a stream as shown in the table above, you can play WebRTC on OvenPlayer via the following URL:

Protocol	URL format
WebRTC Signalling	ws://<Server IP>[:<Signalling Port]/<Application name>/<Stream name>

Secure WebRTC Signalling	wss://<Server IP>[:<Signalling Port]/<Application name>/<Stream name>

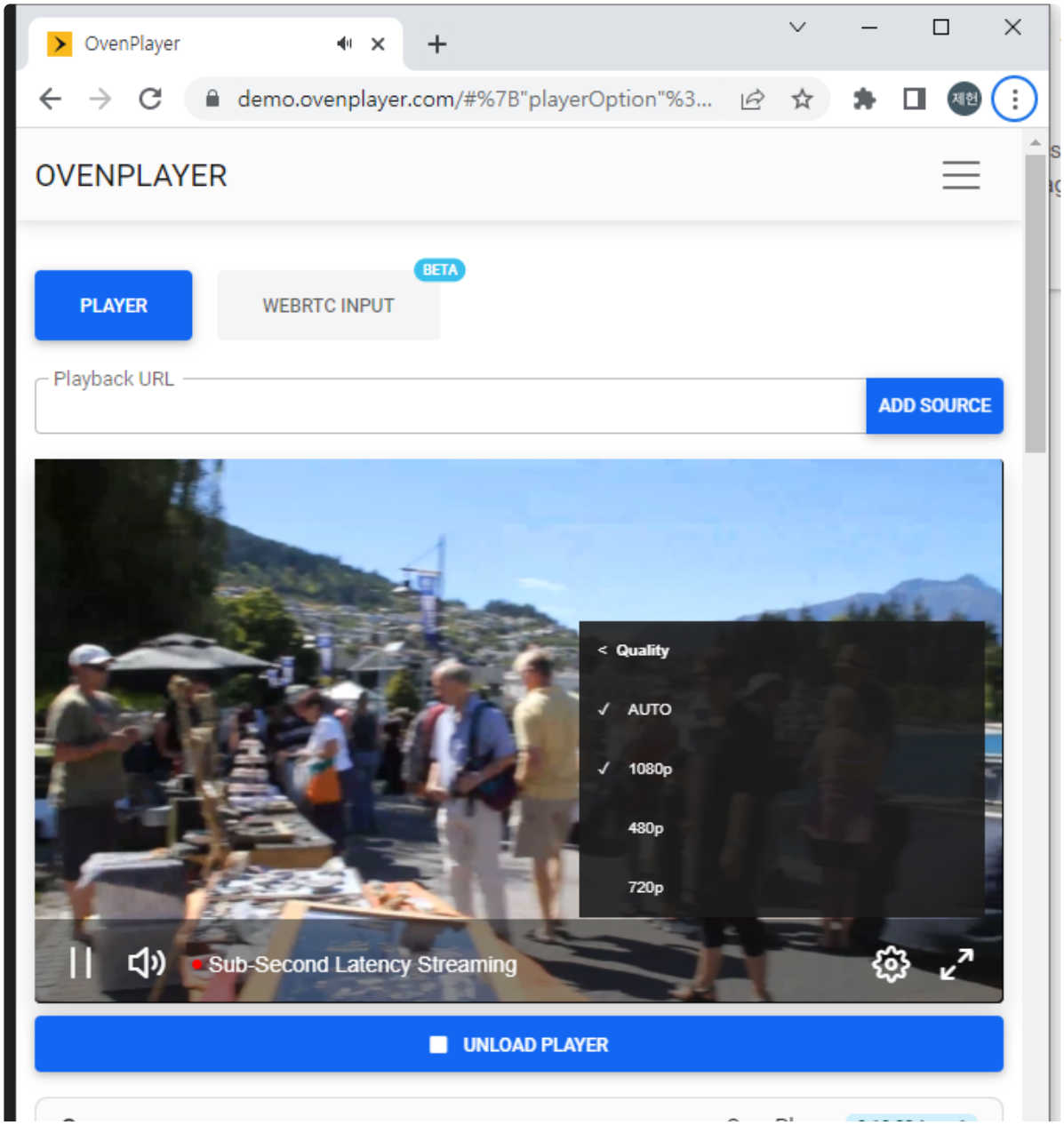
If you use the default configuration, you can stream to the following URL:

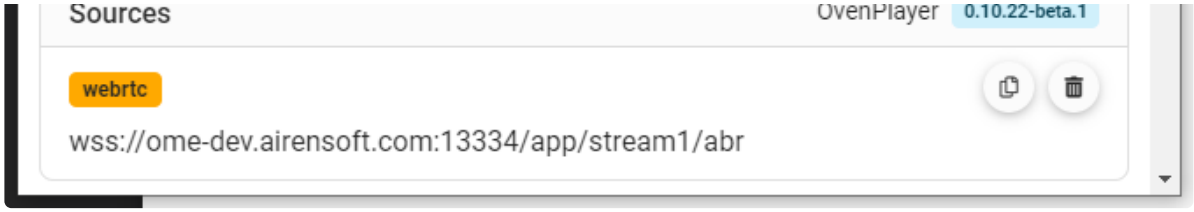
- `ws://[OvenMediaEngine IP]:3333/app/stream`
- `wss://[OvenMediaEngine IP]:3333/app/stream`

We have prepared a test player to make it easy to check if OvenMediaEngine is working. Please see the Test Player chapter for more information.

Adaptive Bitrates Streaming (ABR)

OvenMediaEnigne provides adaptive bitrates streaming over WebRTC. OvenPlayer can also play and display OvenMediaEngine's WebRTC ABR URL.





Create Playlist for WebRTC ABR

You can provide ABR by creating a `playlist` in `<OutputProfile>` as shown below. The URL to play the playlist is `ws[s]://domain[:port]/<app name>/<stream name>/<playlist file name>`

`<Playlist><Rendition><Video>` and `<Playlist><Rendition><Audio>` can connected using `<Encodes><Video><Name>` or `<Encodes><Audio><Name>`.

> It is not recommended to use a <Bypass>true</Bypass> encode item if you want a seamless transition between renditions because there is a time difference between the transcoded track and bypassed track.

If `<Options><WebRtcAutoAbr>` is set to true, OvenMediaEngine will measure the bandwidth of the player session and automatically switch to the appropriate rendition.

Here is an example play URL for ABR in the playlist settings below.
`wss://domain:13334/app/stream/abr`

> Streaming starts from the top rendition of Playlist, and when Auto ABR is true, the server finds the best rendition and switches to it. Alternatively, the user can switch manually by selecting a rendition in the player.

```
<OutputProfiles>
<OutputProfile>
        <Name>default</Name>
        <OutputStreamName>${OriginStreamName}</OutputStreamName>

        <Playlist>
                <Name>for Webrtc</Name>
                <FileName>abr</FileName>
                <Options>
                        <WebRtcAutoAbr>false</WebRtcAutoAbr>
                </Options>
                <Rendition>
                        <Name>1080p</Name>
                        <Video>1080p</Video>
                        <Audio>opus</Audio>
                </Rendition>
                <Rendition>
```

```
                <Name>480p</Name>
                <Video>480p</Video>
                <Audio>opus</Audio>
        </Rendition>
        <Rendition>
                <Name>720p</Name>
                <Video>720p</Video>
                <Audio>opus</Audio>
        </Rendition>
</Playlist>

<Playlist>
        <Name>for llhls</Name>
        <FileName>llhls_abr</FileName>
        <Rendition>
                <Name>480p</Name>
                <Video>480p</Video>
                <Audio>bypass_audio</Audio>
        </Rendition>
        <Rendition>
                <Name>720p</Name>
                <Video>720p</Video>
                <Audio>bypass_audio</Audio>
        </Rendition>
</Playlist>

<Encodes>
        <Video>
                <Name>bypass_video</Name>
                <Bypass>true</Bypass>
        </Video>
        <Video>
                <Name>480p</Name>
                <Codec>h264</Codec>
                <Width>640</Width>
                <Height>480</Height>
                <Bitrate>500000</Bitrate>
                <Framerate>30</Framerate>
        </Video>
        <Video>
                <Name>720p</Name>
                <Codec>h264</Codec>
                <Width>1280</Width>
                <Height>720</Height>
                <Bitrate>2000000</Bitrate>
                <Framerate>30</Framerate>
        </Video>
        <Video>
                <Name>1080p</Name>
                <Codec>h264</Codec>
                <Width>1920</Width>
                <Height>1080</Height>
                <Bitrate>5000000</Bitrate>
```

```
                </Video><Framerate>30</Framerate>
                <Audio>
                        <Name>bypass_audio</Name>
                        <Bypass>True</Bypass>
                </Audio>
                <Audio>
                        <Name>opus</Name>
                        <Codec>opus</Codec>
                        <Bitrate>128000</Bitrate>
                        <Samplerate>48000</Samplerate>
                        <Channel>2</Channel>
                </Audio>
        </Encodes>
</OutputProfile>
</OutputProfiles>
```

See the Adaptive Bitrates Streaming section for more details on how to configure renditions.

Multiple codec support in Playlist

WebRTC can negotiate codecs with SDP to support more devices. Playlist can set rendition with different kinds of codec. And OvenMediaEngine includes only renditions corresponding to the negotiated codec in the playlist and provides it to the player.

> ⓘ If an unsupported codec is included in the Rendition, the Rendition is not used. For example, if the Rendition's Audio contains aac, WebRTC ignores the Rendition.

In the example below, it consists of renditions with H.264 and Opus codecs set and renditions with VP8 and Opus codecs set. If the player selects VP8 in the answer SDP, OvenMediaEngine creates a playlist with only renditions containing VP8 and Opus and passes it to the player.

```
<Playlist>
        <Name>for Webrtc</Name>
        <FileName>abr</FileName>
        <Options>
                <WebRtcAutoAbr>false</WebRtcAutoAbr>
        </Options>
        <Rendition>
                <Name>1080p</Name>
                <Video>1080p</Video>
                <Audio>opus</Audio>
        </Rendition>
        <Rendition>
                <Name>480p</Name>
                <Video>480p</Video>
                <Audio>opus</Audio>
        </Rendition>
        <Rendition>
```

```
                <Name>720p</Name>
                <Video>720p</Video>
                <Audio>opus</Audio>
        </Rendition>

        <Rendition>
                <Name>1080pVp8</Name>
                <Video>1080pVp8</Video>
                <Audio>opus</Audio>
        </Rendition>
        <Rendition>
                <Name>480pVp8</Name>
                <Video>480pVp8</Video>
                <Audio>opus</Audio>
        </Rendition>
        <Rendition>
                <Name>720pVp8</Name>
                <Video>720pVp8</Video>
                <Audio>opus</Audio>
        </Rendition>

</Playlist>
```

WebRTC over TCP

There are environments where the network speed is fast but UDP packet loss is abnormally high. In such an environment, WebRTC may not play normally. WebRTC does not support streaming using TCP, but connections to the TURN (https://tools.ietf.org/html/rfc8656) server support TCP. Based on these characteristics of WebRTC, OvenMediaEngine supports TCP connections from the player to OvenMediaEngine by embedding a TURN server.

Turn on TURN server

You can turn on the TURN server by setting <TcpRelay> in the WebRTC Bind.

> Example : <TcpRelay>*:3478</TcpRelay>

OME may sometimes not be able to get the server's public IP to its local interface. (Environment like Docker or AWS) So, specify the public IP for `Relay IP`. If * is used, the public IP obtained from <StunServer> and all IPs obtained from the local interface are used. `Port` is the tcp port on which the TURN server is listening.

```
<Server version="8">
    ...
    <StunServer>stun.l.google.com:19302</StunServer>
                <Bind>
                        <Publishers>
                                <WebRTC>
```

```
                                ...
                                <IceCandidates>
                                        <!-- <TcpRelay>*:3478</TcpRelay> -->
                                        <TcpRelay>Relay IP:Port</TcpRelay>
                                        <TcpForce>false</TcpForce>
                                        <IceCandidate>*:10000-10005/udp</IceCandidate>
                                </IceCandidates>
                        </WebRTC>
                </Publishers>
        </Bind>
        ...
```

> ⓘ If * is used as the IP of TcpRelay and IceCandidate, all available candidates are generated and sent to the player, so the player tries to connect to all candidates until a connection is established. This can cause delay in initial playback. Therefore, specifying the ${PublicIP} macro or IP directly may be more beneficial to quality.

WebRTC over TCP with OvenPlayer

WebRTC players can configure the TURN server through the iceServers setting.

You can play the WebRTC stream over TCP by attaching the query `transport=tcp` to the existing WebRTC play URL as follows.

```
ws(s)://host:port/app/stream?transport=tcp
```

OvenPlayer automatically sets iceServers by obtaining TURN server information set in <TcpRelay> through signaling with OvenMediaEngine.

> ⓘ If `<TcpForce>` is set to true, it will force a TCP connection even if `?transport=tcp` is not present. To use this, `<TcpRelay>` must be set.

Custom player

If you are using custom player, set iceServers like this:

```
myPeerConnection = new RTCPeerConnection({
  iceServers: [
    {
      urls: "turn:Relay IP:Port?transport=tcp",
      username: "ome",
      credential: "airen"
    }
  ]
});
```

When sending `Request Offer` in the signaling phase with OvenMediaEngine, if you send the `transport=tcp` query string, `ice_servers` information is delivered as follows. You can use this information to set iceServers.

```
candidates: [{candidate: "candidate:0 1 UDP 50 192.168.0.200 10006 typ host", sdpMLineIndex: 
code: 200
command: "offer"
ice_servers: [{credential: "airen", urls: ["turn:192.168.0.200:3478?transport=tcp"], user_name
id: 506764844
peer_id: 0
sdp: {,…}
```

Low-Latency HLS

Apple supports Low-Latency HLS (LLHLS), which enables low-latency video streaming while maintaining scalability. LLHLS enables broadcasting with an end-to-end latency of about 2 to 5 seconds. OvenMediaEngine officially supports LLHLS as of v0.14.0.

LLHLS is an extension of HLS, so legacy HLS players can play LLHLS streams. However, the legacy HLS player plays the stream without using the low-latency function.

Title	Descriptions
Delivery	HTTP/1.1 HTTP/2
Security	TLS (HTTPS)
Container	fMP4
Codecs	H.264 AAC

Configuration

To use LLHLS, you need to add the `<LLHLS>` elements to the `<Publishers>` in the configuration as shown in the following example.

```
<Server version="8">
    <Bind>
        <Publishers>
            <LLHLS>
                <!--
                OME only supports h2, so LLHLS works over HTTP/1.1 on non-TLS ports.
```

```
                LLHLS works with higher performance over HTTP/2,
                so it is recommended to use a TLS port.
                -->
                <Port>80</Port>
                <TLSPort>443</TLSPort>
                <WorkerCount>1</WorkerCount>
            </LLHLS>
        </Publishers>
    </Bind>
    <VirtualHosts>
        <VirtualHost>
            <Applications>
                <Application>
                    <Publishers>
                        <LLHLS>
                                <ChunkDuration>0.2</ChunkDuration>
                                <SegmentDuration>6</SegmentDuration>
                                <SegmentCount>10</SegmentCount>
                                <CrossDomains>
                                        <Url>*</Url>
                                </CrossDomains>
                        </LLHLS>
                    </Publishers>
                </Application>
            </Applications>
        </VirtualHost>
    </VirtualHosts>
</Server>
```

Element	Decscription
Bind	Set the HTTP ports to provide LLHLS.
ChunkDuration	Set the partial segment length to fractional seconds. This value affects low-latency HLS player. We recommend **0.2** seconds for this value.
SegmentDuration	Set the length of the segment in seconds. Therefore, a shorter value allows the stream to start faster. However, a value that is too short will make legacy HLS players unstable. Apple recommends **6** seconds for this value.
SegmentCount	The number of segments listed in the playlist. This value has little effect on LLHLS players, so use **10** as recommended by Apple. 5 is recommended for legacy HLS players. Do not set below 3. It can only be used for experimentation.

CrossDomains	Control the domain in which the player works through `<CorssDomain>`. For more information, please refer to the CrossDomain section.

> HTTP/2 outperforms HTTP/1.1, especially with LLHLS. Since all current browsers only support h2, HTTP/2 is supported only on TLS port. Therefore, it is highly recommended to use LLHLS on the TLS port.

Adaptive Bitrates Streaming (ABR)

LLHLS can deliver adaptive bitrate streaming. OME encodes the same source with multiple renditions and delivers it to the players. And LLHLS Player, including OvenPlayer, selects the best quality rendition according to its network environment. Of course, these players also provide option for users to manually select rendition.

See the Adaptive Bitrates Streaming section for how to configure renditions.

CrossDomain

Most browsers and players prohibit accessing other domain resources in the currently running domain. You can control this situation through Cross-Origin Resource Sharing (CORS) or Cross-Domain (CrossDomain). You can set CORS and Cross-Domain as `<CrossDomains>` element.

Server.xml

```
<CrossDomains>
    <Url>*</Url>
    <Url>*.airensoft.com</Url>
    <Url>http://*.ovenplayer.com</Url>
    <Url>https://demo.ovenplayer.com</Url>
</CrossDomains>
```

You can set it using the `<Url>` element as shown above, and you can use the following values:

Url Value	Description
*	Allows requests from all Domains
domain	Allows both HTTP and HTTPS requests from the specified Domain

~~http://domain~~	~~Allows HTTP requests from the specified Domain~~
https://domain	Allows HTTPS requests from the specified Doma

Streaming

LLHLS is ready when a live source is inputted and a stream is created. Viewers can stream using OvenPlayer or other players.

If your input stream is already h.264/aac, you can use the input stream as is like below. If not, or if you want to change the encoding quality, you can do Transcoding.

```
<OutputProfiles>
        <OutputProfile>
                <Name>bypass_stream</Name>
                <OutputStreamName>${OriginStreamName}</OutputStreamName>
                <Encodes>
                        <Audio>
                                <Bypass>true</Bypass>
                        </Audio>
                        <Video>
                                <Bypass>true</Bypass>
                        </Video>
                </Encodes>
        </OutputProfile>
</OutputProfiles>
```

When you create a stream, as shown above, you can play LLHLS with the following URL:

> http[s]://domain[:port]/<app name>/<stream name>/llhls.m3u8

If you use the default configuration, you can start streaming with the following URL:

`https://domain:3334/app/<stream name>/llhls.m3u8`

We have prepared a test player that you can quickly see if OvenMediaEngine is working. Please refer to the Test Player for more information.

Low-Latency DASH and Legacy HLS streaming

> ⚠ From OvenMediaEngine v0.14.0, updates to legacy HLS, DASH, and LLDASH are now discontinued. These will be **deprecated**.

LLHLS, released from v0.14.0, is superior to Dash and LLDASH in all aspects of compatibility, performance

and function, and also support legacy HLS players. Therefore, we decided not to update legacy HLS, DASH and LLDASH anymore. With the energy that was used to maintain these features, we will focus on more wonderful features in the future.

For legacy HLS, DASH, and LLDASH, refer to the old version manual.

https://airensoft.gitbook.io/ovenmediaengine/v/0.13.0/streaming/hls-mpeg-dash

Access Control

SignedPolicy

Overview

SignedPolicy is a module that limits the user's privileges and time. For example, operators can distribute RTMP URLs that can be accessed for 60 seconds to authorized users, and limit RTMP transmission to 1 hour. The provided URL will be destroyed after 60 seconds, and transmission will automatically stop after 1 hour. Users who are provided with a SingedPolicy URL cannot access resources other than the provided URL. This is because the SignedPolicy URL is authenticated.

SingedPolicy URL consists of the query string of the streaming URL with Policy and Signature as shown below. If SignedPolicy is enabled in the configuration of OvenMediaEngine, access to URLs with no signature or invalid signature is not allowed. Signature uses HMAC-SHA1 to authenticate all URLs except signature.

```
scheme://domain.com:port/app/stream?policy=<>&signature=<>
```

Policy

Policy is in json format and provides the following properties.

```
{
    "url_activate":1399711581,
    "url_expire":1399721581,
    "stream_expire":1399821581,
    "allow_ip":"192.168.100.5/32"
}
```

Key	Value	Description

url_expire **(Required)**	<Number> Milliseconds since unix epoch	The time the URL expires Reject on request after the expiration
url_activate **(Optional)**	<Number> Milliseconds since unix epoch	The time the URL activates Reject on request before activation
stream_expire **(Optional)**	<Number> Milliseconds since unix epoch	The time the Stream expires Transmission and playback stc when the time expires
allow_ip **(Optional)**	<String> IPv4 CIDR	Allowed IP address range, 192.168.0.0/24

ⓘ **url_expire** means the time the URL is valid, so if you connect before the URL expires, you can continue to use it, and sessions that have already been connected will not be deleted even if the time expires. However, **stream_expire** forcibly terminates the session when the time expires even if it is already playing.

Signature

Signature is generated by HMAC-SHA1 encoding all URLs except signature query string. The generated Signature is encoded using Base64URL and included as a query string of the existing URL.

```
Base64URL.Encode(
    HMAC.Encrypt(
        SHA1,
        secret_key,
        "scheme://domain.com:port/app/stream[/file]?policy='encoded policy'>"
    )
)
```

⚠ The URL entered into HMAC to generate the Signature must include :port.

When creating a signature, you cannot omit the default port such as http port 80, https port 443, or rtmp port 1935. This is because when OvenMediaEngine creates a signature for checking the signature, it is created by putting the port value.

⚠ When using SignedPolicy with SRT providers, only use the **streamid** portion of the URL, e.g. srt://myserver:9999?streamid=**srt://myserver:9999/app/stream?policy=abc123**

Configuration

To enable SignedPolicy, you need to add the following <SingedPolicy> setting in Server.xml under <VirtualHost>.

```
<VirtualHost>
    <SignedPolicy>
        <PolicyQueryKeyName>policy</PolicyQueryKeyName>
        <SignatureQueryKeyName>signature</SignatureQueryKeyName>
        <SecretKey>aKq#1kj</SecretKey>

        <Enables>
            <Providers>rtmp</Providers>
            <Publishers>webrtc,llhls</Publishers>
        </Enables>
    </SignedPolicy>
</VirtualHost>
```

Key	Description
PolicyQueryKeyName	The query string key name in the URL pointing to the policy value
SignatureQueryKeyName	The query string key name in the URL pointing to the signature value
SecretKey	The secret key used when encoding with HMAC-SHA1
Enables	List of providers and publishers to enable SignedPolicy. Currently, SingedPolicy supports rtmp among providers, and among publishers, WebRTC, LLHLS are supported.

Make SignedPolicy URL with a script

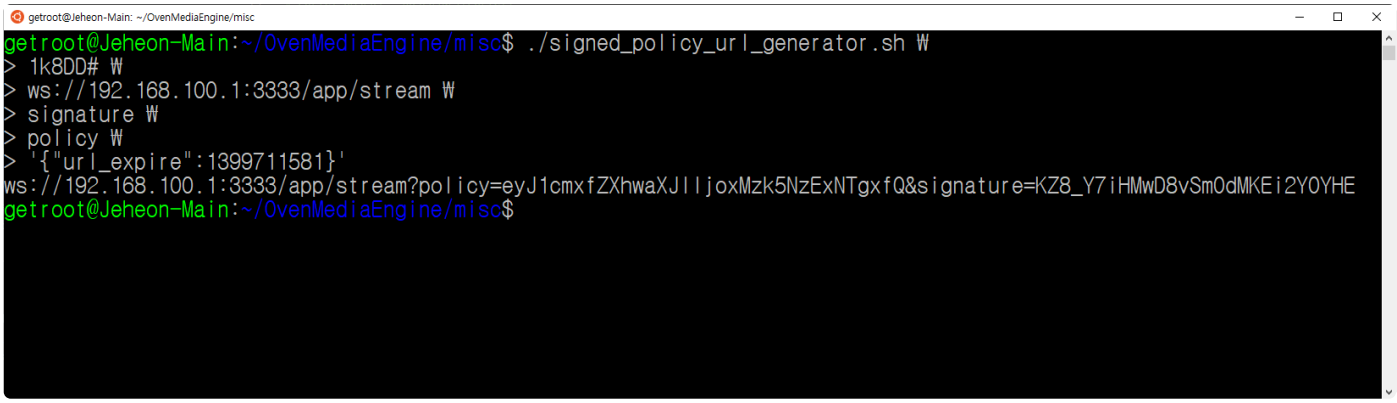
We provide a script that can easily generate SingedPolicy URL. The script can be found in the path below.

```
/misc/signed_policy_url_generator.sh
```

Here's how to use this script:

```
./signed_policy_generator.sh [HMAC_KEY] [BASE_URL] [SIGNATURE_QUERY_KEY_NAME] [POLICY_QUERY_KE
```

For example, you can use it like this:



Make SingedPolicy URL manually

> We hope to provide SignedPolicy URL Generator Library in various languages. If you have created the SignedPolicy URL Generator Library in another language, please send a Pull Request to our GITHUB. Thank you for your open source contributions.

Encoding policy

In order to include the policy in the URL, it must be encoded with Base64URL.

Plain {Policy}

```
{"url_expire":1399721581}
```

Base64URL Encoded {Policy}

```
eyJ1cmxfZXhwaXJlIjoxMzk5NzIxNTgxfQ
```

Policy encoded with Base64URL is added as a query string to the existing streaming URL. (The query string key is set in Server.xml.)

```
ws://192.168.0.100:3333/app/stream?policy=eyJ1cmxfZXhwaXJlIjoxMzk5NzIxNTgxfQ
```

Signature

Signature hashes the entire URL including the policy in HMAC (SHA-1) method, encodes it as Base64URL, and includes it in the query string.

URL input to signature generation

```
ws://192.168.0.100:3333/app/stream?policy=eyJ1cmxfZXhwaXJlIjoxMzk5NzIxNTgxfQ
```

Create a hash using the secret key (1kU^b6 in the example) and the URL above using HMAC-SHA1.

Base64URL encoded { HMAC-SHA1 <KEY : 1kU^b6> (URL) }

```
dvVdBpoxAeCPl94Kt5RoiqLI0YE
```

If you include it as a signature query string (query string key is set in Server.xml), the following SignedPolicy URL is finally generated.

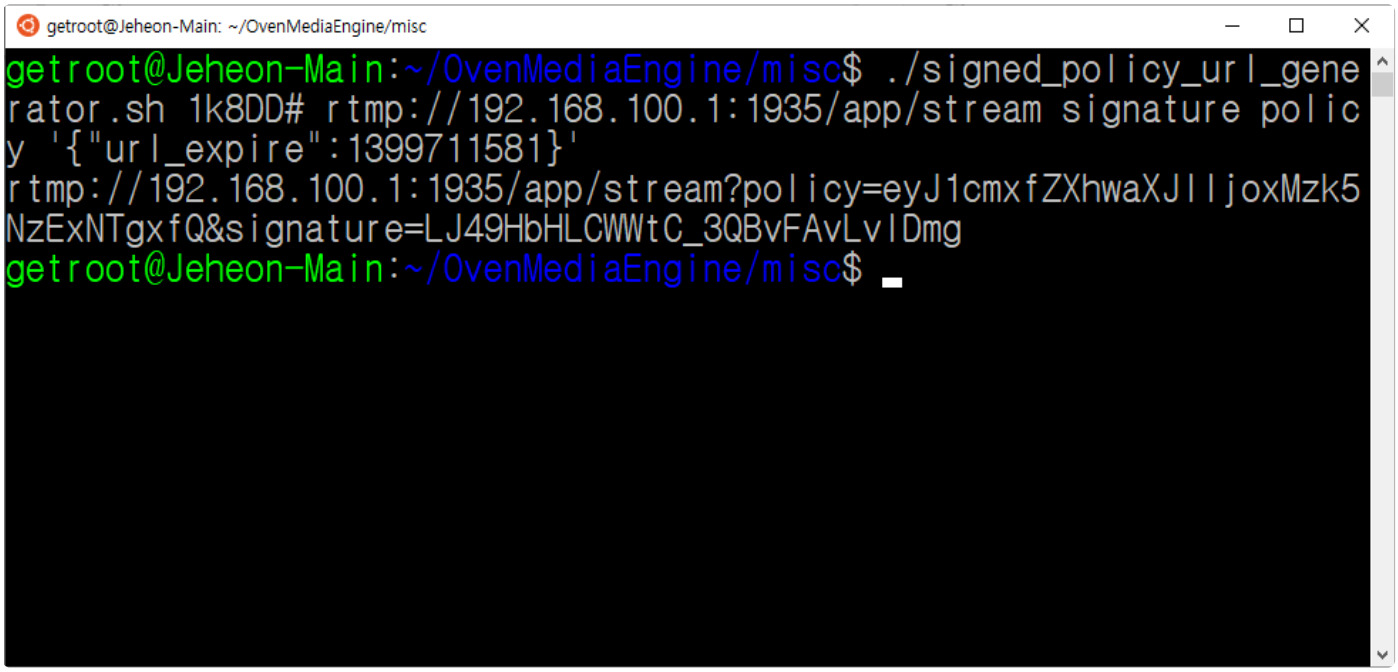
URL with signature

```
ws://192.168.0.100/app/stream?policy=eyJ1cmxfZXhwaXJlIjoxMzk5NzIxNTgxfQ&signature=dvVdBpoxAeCl
```

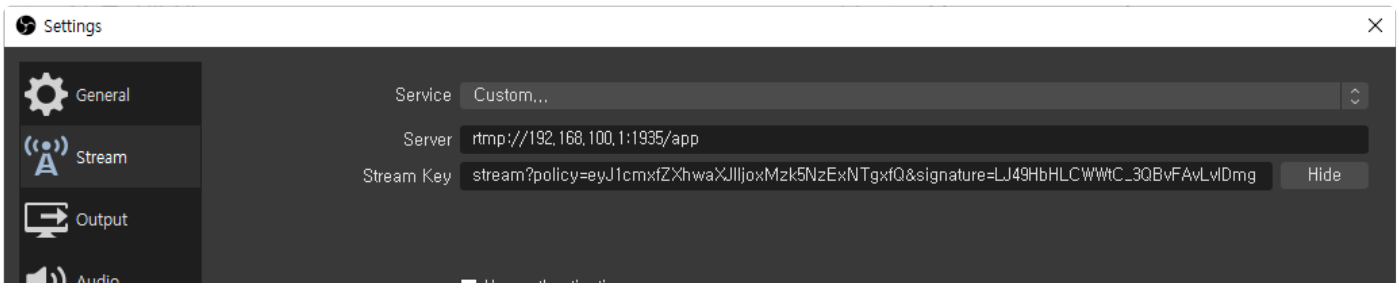
Usage examples

Applying SignedPolicy in OBS

Generate SingedPolicy URL with the script.



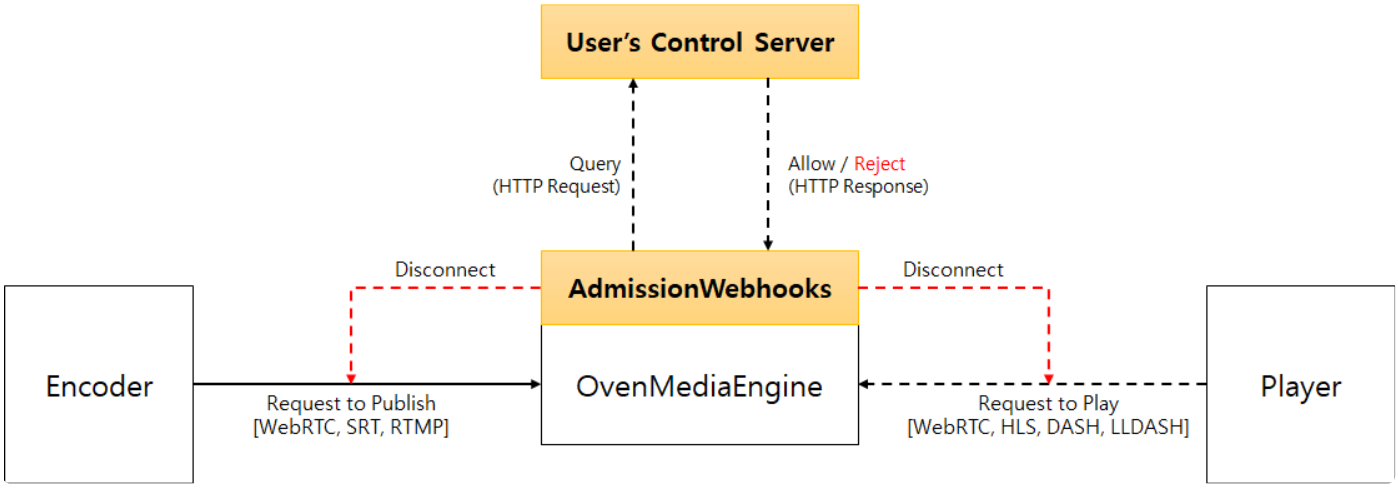
Separate the URL based on "app" as shown in the example below and enter all the parts under the stream in the Stream Key.



AdmissionWebhooks (beta)

Overview

AdmissionWebhooks are HTTP callbacks that query the control server to control publishing and playback admission requests.



Users can use the AdmissionWebhook for a variety of purposes, including customer authentication, tracking published streams, hide app/stream names, logging and more.

Configuration

AdmissionWebhooks can be set up on VirtualHost, as shown below.

```
<VirtualHost>
        <AdmissionWebhooks>
                <ControlServerUrl>https://192.168.0.161:9595/v1/admission</ControlServerUrl>
                <SecretKey>1234</SecretKey>
                <Timeout>3000</Timeout>
```

```
                <Enables>
                        <Providers>rtmp,webrtc,srt</Providers>
                        <Publishers>webrtc,llhls</Publishers>
                </Enables>
        </AdmissionWebhooks>
</VirtualHost>
```

Key	Description
ControlServerUrl	The HTTP Server to receive the query. HTTP and HTTPS are available.
SecretKey	The secret key used when encrypting with HMAC SHA1 For more information, see Security.
Timeout	Time to wait for a response after request (in milliseconds)
Enables	Enable Providers and Publishers to use AdmissionWebhooks

Request

Format

AdmissionWebhooks send HTTP/1.1 request message to the configured user's control server when an encoder requests publishing or a player requests playback. The request message format is as follows.

```
POST /configured/tartget/url/ HTTP/1.1
Content-Length: 325
Content-Type: application/json
Accept: application/json
X-OME-Signature: f871jd991jj1929jsjd91pqa0amm1
{
  "client":
  {
    "address": "211.233.58.86",
    "port": 29291
  },
  "request":
  {
    "direction": "incoming | outgoing",
    "protocol": "webrtc | rtmp | srt | llhls",
    "status": "opening | closing",
    "url": "scheme://host[:port]/app/stream/file?query=value&query2=value2",
    "time": ""2021-05-12T13:45:00.000Z"
```

```
} }
```

The message is sent in POST method and the payload is in application/json format. X-OME-Signature is a base64 url safe encoded value obtained by encrypting the payload with HMAC-SHA1 so that the ControlServer can validate this message. See the Security section for more information on X-OME-Signature.

Here is a detailed explanation of each element of Json payload:

Element	Sub-Element	Description
client		Information of the client who requested the connection.
	address	Client's IP address
	port	Client's Port number
request		Information about the client's request
	direction	incoming : A client requests to publish a stream outgoing : A client requests to play a stream
	protocol	webrtc, srt, rtmp, hls, dash, lldash
	status	opening : A client requests to open a stream outgoing : A client closed the stream
	url	url requested by the client
	time	time requested by the client (ISO8601 format)

Security

The control server may need to validate incoming http requests for security reasons. To do this, the AdmissionWebhooks module puts the `X-OME-Signature` value in the HTTP request header. `X-OME-Signature` is a base64 url safe encoded value obtained by encrypting the payload of an HTTP request with the HMAC-SHA1 algorithm using the secret key set in `<AdmissionWebhooks><SecretKey>` of the configuration.

Conditions that triggers the request

As shown below, the trigger condition of request is different for each protocol.

Protocol	Condition
WebRTC	When a client requests Offer SDP
RTMP	When a client sends a publish message
SRT	When a client send a streamid
LLHLS	When a client requests a playlist (llhls.m3u8)

Response for closing status

The engine in the closing state does not need any parameter in response. To the query just answer with empty json object.

```
HTTP/1.1 200 OK
Content-Length: 5
Content-Type: application/json
Connection: Closed
{
}
```

Response for opening status

Format

ControlServer must respond with the following Json format. In particular, the `"allowed"` element is required.

```
HTTP/1.1 200 OK
Content-Length: 102
Content-Type: application/json
Connection: Closed
{
  "allowed": true,
  "new_url": "scheme://host[:port]/app/stream/file?query=value&query2=value2",
  "lifetime": milliseconds,
  "reason": "authorized"
}
```

Element	Description

allowed **(required)**	true or false Allows or rejects the client's request.
new_url (optional)	Redirects the client to a new url. However, the `scheme`, `port`, and `file` cannot be different from the request. The host can only be changed to another virtual host on the same server.
lifetime (optional)	The amount of time (in milliseconds) that a client can maintain a connection (Publishing or Playback) • 0 means infinity HTTP based streaming (HLS, DASH, LLDASH) does not keep a connection, so this value does not apply.
reason (optional)	If allowed is false, it will be output to the log.

User authentication and control

`new_url` redirects the original request to another app/stream. This can be used to hide the actual app/stream name from the user or to authenticate the user by inserting additional information instead of the app/stream name.

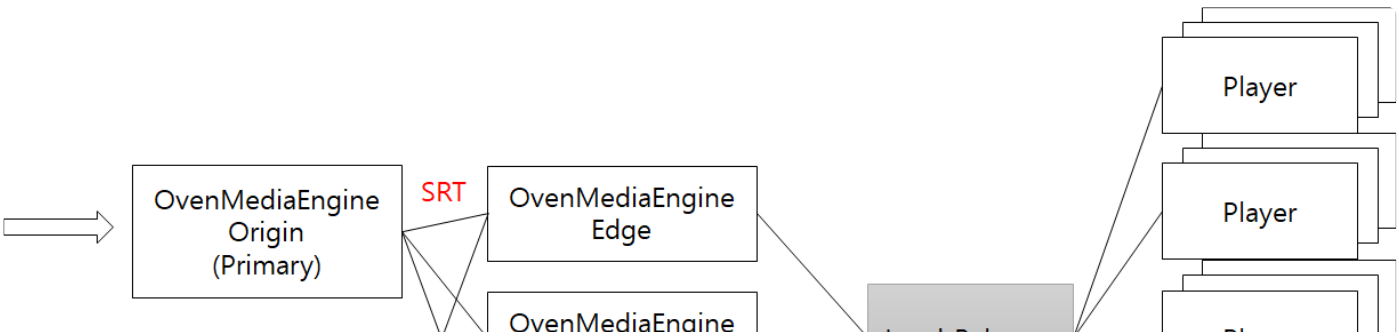
For example, you can issue a WebRTC streaming URL by inserting the user ID as follows: `ws://domain.com:3333/user_id` It will be more effective if you issue a URI with the encrypted value that contains the user ID, url expiration time, and other information.

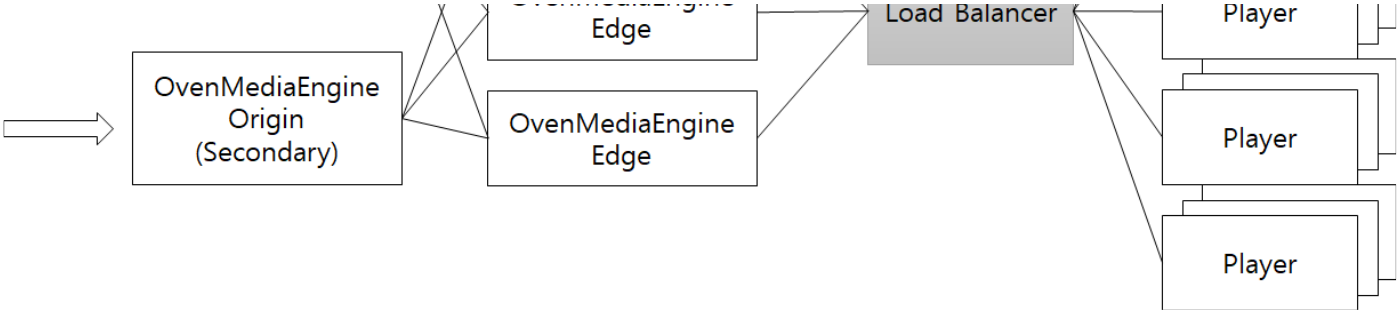
After the Control Server checks whether the user is authorized to play using `user_id`, and responds with `ws://domain.com:3333/app/sport-3` to `new_url`, the user can play app/sport-3.

If the user has only one hour of playback rights, the Control Server responds by putting 3600000 in the `lifetime`.

Clustering

OvenMediaEngine supports clustering and ensures High Availability (HA) and scalability.





OvenMediaEngine supports the Origin-Edge structure for cluster configuration and provides scalability. Also, you can set Origin as `Primary` and `Secondary` in OvenMediaEngine for HA.

Origin-Edge Configuration

The OvenMediaEngine running as edge pulls a stream from an external server when a user requests it. The external server could be another OvenMediaEngine with OVT enabled or another stream server that supports RTSP.

The OVT is a protocol defined by OvenMediaEngine to relay stream between Origin-Edge and OVT can be run over SRT and TCP. For more information on the SRT Protocol, please visit the SRT Alliance site.

Origin

OvenMediaEngine provides OVT protocol for passing streams from the origin to the edge. To run OvenMediaEngine as Origin, OVT port, and OVT Publisher must be enabled as follows :

```
<Server version="5">
        <Bind>
                <Publishers>
                        <OVT>
                                <Port>9000</Port>
                        </OVT>
                </Publishers>
        </Bind>
        <VirtualHosts>
                <VirtualHost>
                        <Applications>
                                <Application>
                                        ...
                                        <Publishers>
                                                <OVT />
                                        </Publishers>
                                </Application>
                        </Applications>
                </VirtualHost>
        </VirtualHosts>
</Server>
```

Edge

The role of the edge is to receive and distribute streams from an origin. You can configure hundreds of Edge to distribute traffic to your players. As a result of testing, a single edge can stream 4-5Gbps traffic by WebRTC based on AWS C5.2XLarge. If you need to stream to thousands of people, you can configure and use multiple edges.

The edge supports OVT and RTSP to pull stream from an origin. In the near future, we will support more protocols. The stream pulled through OVT or RTSP is bypassed without being encoded.

> In order to re-encode the stream created by OVT and RTSP, the function to put into an existing application will be supported in the future.

To run OvenMediaEngine as Edge, you need to add Origins elements to the configuration file as follows:

```
<VirtualHosts>
        <VirtualHost>
                <Origins>
                        <Properties>
                                <NoInputFailoverTimeout>3000</NoInputFailoverTimeout>
                                <UnusedStreamDeletionTimeout>60000</UnusedStreamDeletionTimeou
                        </Properties>
                        <Origin>
                                <Location>/app/stream</Location>
                                <Pass>
                                        <Scheme>ovt</Scheme>
                                        <Urls><Url>origin.com:9000/app/stream_720p</Url></Urls
                                </Pass>
                                <ForwardQueryParams>true</ForwardQueryParams>
                        </Origin>
                        <Origin>
                                <Location>/app/</Location>
                                <Pass>
                                        <Scheme>OVT</Scheme>
                                        <Urls><Url>origin.com:9000/app/</Url></Urls>
                                </Pass>
                        </Origin>
                        <Origin>
                                <Location>/</Location>
                                <Pass>
                                        <Scheme>RTSP</Scheme>
                                        <Urls><Url>origin2.com:9000/</Url></Urls>
                                </Pass>
                        </Origin>
                </Origins>
        </VirtualHost>
</VirtualHosts>
```

The `<Origin>` is a rule about where to pull a stream from for what request.

The `<Origin>` has the ability to automatically create an application with that name if the application you set in `<Location>` doesn't exist on the server. If an application exists in the system, a stream will be created in the application.

The automatically created application by `<Origin>` enables all providers but if you create an application yourself, you must enable the provider that matches the setting as follows.

```
<VirtualHosts>
        <VirtualHost>
                <Origins>
                        <Properties>
                                <NoInputFailoverTimeout>3000</NoInputFailoverTimeout>
                                <UnusedStreamDeletionTimeout>60000</UnusedStreamDeletionTimeou
                        </Properties>
                        <Origin>
                                <Location>/this_application/stream</Location>
                                <Pass>
                                        <Scheme>OVT</Scheme>
                                        <Urls><Url>origin.com:9000/app/stream_720p</Url></Urls
                                </Pass>
                                <ForwardQueryParams>true</ForwardQueryParams>
                        </Origin>
                        <Origin>
                                <Location>/this_application/rtsp_stream</Location>
                                <Pass>
                                        <Scheme>RTSP</Scheme>
                                        <Urls><Url>rtsp.origin.com/145</Url></Urls>
                                </Pass>
                        </Origin>
                </Origins>
                <Applications>
                        <Application>
                                <Name>this_application</Name>
                                <Type>live</Type>
                                <Providers>
                                        <!-- You have to enable the OVT provider
                                        because you used the ovt scheme for configuring Origi
                                        <OVT />
                                        <!-- If you set RTSP into Scheme,
                                        you have to enable RTSPPull provider -->
                                        <RTSPPull />
                                </Providers>
```

<Properties>

NoInputFailoverTimeout (default 3000)

NoInputFailoverTimeout is the time (in milliseconds) to switch to the next URL if there is no input for the set time.

UnusedStreamDeletionTimeout (default 60000)

UnusedStreamDeletionTimeout is a function that deletes a stream created with OriginMap if there is no viewer for a set amount of time (milliseconds). This helps to save network traffic and system resources for Origin and Edge.

<Origin>

For a detailed description of Origin's elements, see:

Location

Origin is already filtered by domain because it belongs to VirtualHost. Therefore, in Location, set App, Stream, and File to match except domain area. If a request matches multiple Origins, the top of them runs.

Pass

Pass consists of Scheme and Url.

`<Scheme>` is the protocol that will use to pull from the Origin Stream. It currently can be configured as `OVT` or `RTSP`.

If the origin server is OvenMediaEngine, you have to set `OVT` into the `<Scheme>`.

You can pull the stream from the RTSP server by setting `RTSP` into the `<Scheme>`. In this case, the `<RTSPPull>` provider must be enabled. The application automatically generated by Origin doesn't need to worry because all providers are enabled.

`Urls` is the address of origin stream and can consist of multiple URLs.

`ForwardQueryParams` is an option to determine whether to pass the query string part to the server at the URL you requested to play.(**Default : true**) Some RTSP servers classify streams according to query strings, so you may want this option to be set to false. For example, if a user requests `ws://host:port/app/stream?transport=tcp` to play WebRTC, the `?transport=tcp` may also be forwarded to the RTSP server, so the stream may not be found on the RTSP server. On the other hand, OVT does not affect anything, so you can use it as the default setting.

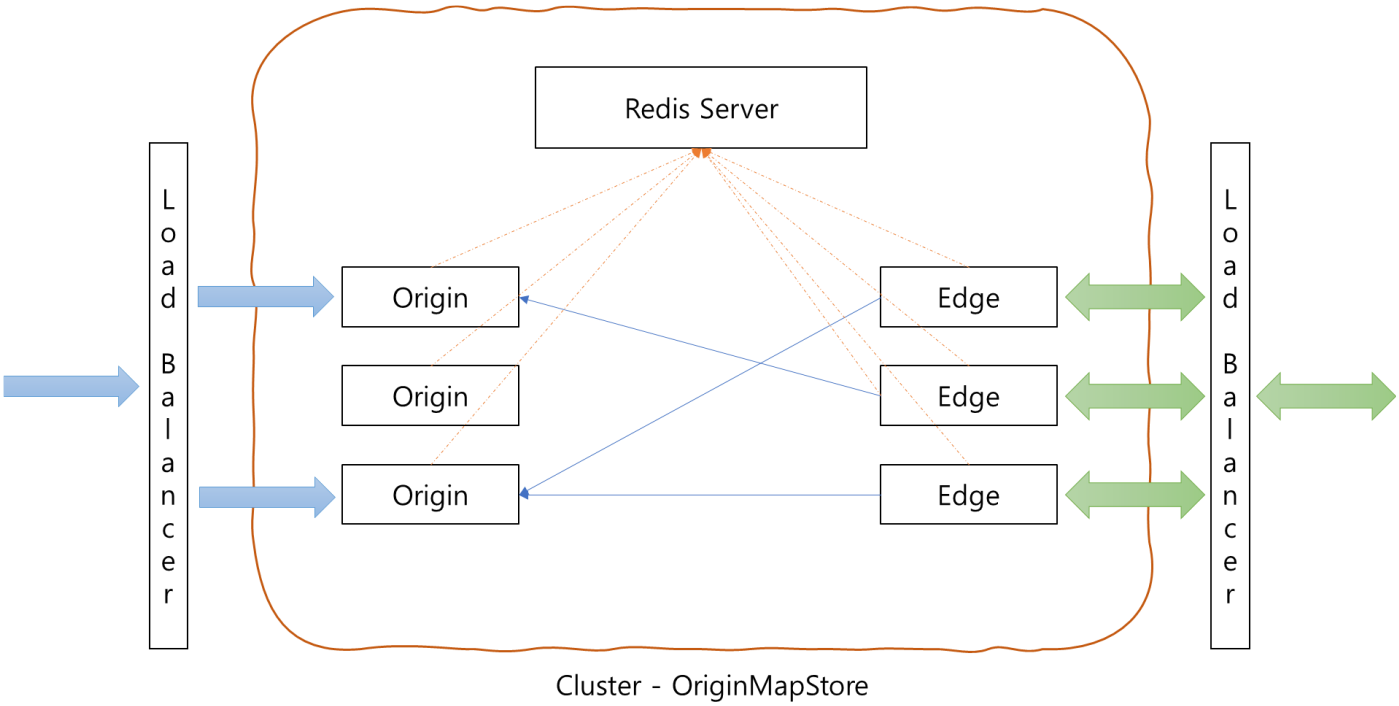
Rules for generating Origin URL

The final address to be requested by OvenMediaEngine is generated by combining the configured Url and user's request except for Location. For example, if the following is set

```
<Location>/edge_app/</Location>
<Pass>
    <Scheme>ovt</Scheme>
    <Urls><Url>origin.com:9000/origin_app/</Url></Urls>
</Pass>
```

If a user requests http://edge.com/edge_app/stream, OvenMediaEngine makes an address to ovt:

OriginMapStore



OriginMapStore is designed to make it easier to support autoscaling within a cluster. All Origin Servers and Edge Servers in the cluster share stream information and origin OVT URLs through Redis. That is, when a stream is created on the Origin server, the Origin server sets the app/stream name and OVT url to access the stream to the Redis server. Edge gets the OVT url corresponding to the app/stream from the Redis server when the user's playback request comes in.

This means that existing settings do not need to be updated when extending Origin servers and Edge servers. Therefore, all Origins can be grouped into one domain, and all Edges can be bundled with one domain. OriginMapStore allows you to expand Origins or Edges within a cluster without any additional configuration.

OriginMapStore functionality has been tested with Redis Server 5.0.7. You can enable this feature by adding the following settings to Server.xml of Origin and Edge. Note that must be set in Server.xml of the Origin server. This is used when Origin registers its own OVT url, so you just need to set a domain name or IP address that can be accessed as an OVT publisher.

```
<VirtualHost>
        ...
        <OriginMapStore>
                <!-- In order to use OriginMap, you must enable OVT Publisher in Origin and O
                <RedisServer>
                        <Host>192.168.0.160:6379</Host>
                        <Auth>!@#ovenmediaengine</Auth>
                </RedisServer>

                <!-- This is only needed for the origin server and used to register the ovt a
```

```
                <OriginHostName>ome-dev.airensoft.com</OriginHostName>
        </OriginMapStore>
        ...
</VirtualHost>
```

Load Balancer

When you are configuring Load Balancer, you need to use third-party solutions such as L4 Switch, LVS, or GSLB, but we recommend using DNS Round Robin. Also, services such as cloud-based AWS Route53, Azure DNS, or Google Cloud DNS can be a good alternative.

Thumbnail

OvenMediaEngine can generate thumbnails from live streams. This allows you to organize a broadcast list on your website or monitor multiple streams at the same time.

Configuration

Bind

Thumbnails are published via HTTP(s). Set the port for thumbnails as follows. Thumbnail publisher can use the same port number as HLS and DASH.

```
<Bind>
        <Publishers>
          ...
                <Thumbnail>
                        <Port>20080</Port>
                        <!-- If you need TLS support, please uncomment below:
                        <TLSPort>20081</TLSPort>
                        -->
                </Thumbnail>
        </Publishers>
</Bind>
```

Encoding

In order to publish thumbnails, an encoding profile for thumbnails must be set. JPG and PNG are supported as codec. And framerate and resolution can be adjusted. Framerate is the number of thumbnails extracted per second. We recommend 1 as the thumbnail framerate. Thumbnail encoding uses a lot of resources. Therefore, if you increase this value excessively, it can cause a failure due to excessive use of system

resources. The resolution can be set as desired by the user, and if the ratio is different from the input image,

```
<OutputProfiles>
        <OutputProfile>
                <Name>default_stream</Name>
                <OutputStreamName>${OriginStreamName}_preview</OutputStreamName>
                <Encodes>
                        <Image>
                                <Codec>jpeg</Codec>
                                <Framerate>1</Framerate>
                                <Width>1280</Width>
                                <Height>720</Height>
                        </Image>
                        <Image>
                                <Codec>png</Codec>
                                <Framerate>1</Framerate>
                                <Width>1280</Width>
                                <Height>720</Height>
                        </Image>
                </Encodes>
        </OutputProfile>
</OutputProfiles>
```

Publisher

Declaring a thumbnail publisher. Cross-domain settings are available as a detailed option.

```
<Publishers>
  ...
        <Thumbnail>
                <CrossDomains>
                        <Url>*</Url>
                </CrossDomains>
        </Thumbnail>
</Publishers>
```

Get thumbnails

When the setting is made for the thumbnail and the stream is input, you can view the thumbnail through the following URL.

Method	URL Pattern
GET	http(s)://<ome_host>*:<port>/<app_name>/<output_stream_name>/thun.<jpg|png>*

Recording (Beta)

OvenMediaEngine can record live streams. You can start and stop recording the output stream through REST API. When the recording is complete, a recording information file is created together with the recorded file so that the user can perform various post-recording processing.

Configuration

File Publisher

To enable recording, add the `<FILE>` publisher to the configuration file as shown below. `<FilePath>` and `<InfoPath>` are required and used as default values. <FilePath> is the setting for the file path and file name. `<InfoPath>` is the setting for the path and name of the XML file that contains information about the recorded files. If there is no file path value among parameters when requesting recording through API, recording is performed with the set default value. This may be necessary if for security reasons you do not want to specify the file path when calling the API to avoid exposing the server's internal path.
< `<RootPath>` is an optional parameter. It is used when requesting with a relative path is required when requesting an API. also, it is applied to `<FilePath>` and `<InfoPath>` as in the example below.

You must specify `.ts` or `.mp4` at the end of the FilePath string to select a container for the recording file. We recommend using .ts unless you have a special case. This is because vp8 and opus codecs are not recorded due to container limitations if you choose .mp4.

```
<Publishers>
        <FILE>
                <RootPath>/mnt/shared_volumes</RootPath>
                <FilePath>/${VirtualHost}/${Application}/${Stream}/${StartTime:YYYYMMDDhhmmss}
                <InfoPath>/${VirtualHost}/${Application}/${Stream}.xml</InfoPath>
        </FILE>
</Publishers>
```

Various macro values are supported for file paths and names as shown below.

Macro Definition

Macro	Description
${TransactionId}	Unique ID for the recording transaction. It is automatically created when recording starts. and i released when recording is stopped. In case of sp recording, it is distinguished that it is the same transaction.

${Id}	User-defined identification ID
${StartTime:YYYYMMDDhhmmss}	Recording start time YYYY - Year MM - Month DD - Days hh : Hours (0~~23~~) mm : Minutes (0059) ss : Seconds (00~59)
${EndTime:YYYYMMDDhhmmss}	Recording end time YYYY - Year MM - Month DD - Days hh : Hours (0~~23~~) mm : Minutes (0059) ss : Seconds (00~59)
${VirtualHost}	Virtual host name
${Application}	Application name
${SourceStream}	Source stream name
${Stream}	Output stream name
${Sequence}	Sequence value that increases when splitting a fil in a single transaction

Start & Stop Recording

For control of recording, use the REST API. Recording can be requested based on the output stream name (specified in the JSON body), and all/some tracks can be selectively recorded. And, it is possible to simultaneously record multiple files for the same stream. When recording is complete, an XML file is created at the path specified in InfoPath. For a sample of the recorded file information XML, refer to Appendix A.

For how to use the API, please refer to the link below.

Recording

Split Recording

Split recording methods provide **interval** and **schedule**. The interval method splits files based on the

accumulated recording time. The Schedule method then splits files according to scheduling options based on system time. The scheduling option is the same as the pattern used in crontab. However, only three options are used: seconds/minutes/hour.

> ⓘ **interval** and **schedule** methods cannot be used simultaneously.

Appendix A. Recorded File Information Specification

The following is a sample of an XML file that expresses information on a recorded file.

```
<?xml version="1.0" encoding="utf-8"?>
<files>
  <file>
    <transactionId>bcUCyJeKuOGnsah3</transactionId>
    <id>CTS_ID001</id>
    <vhost>default</vhost>
    <app>app</app>
    <stream>stream_o</stream>
    <filePath><![CDATA[/home/dev/OvenMediaEngine/records/bcUCyJeKuOGnsah3_default_app_stream_
    <recordBytes>8774737</recordBytes>
    <recordTime>60011</recordTime>
    <sequence>0</sequence>
    <interval>60000</interval>
    <lastSequence>true</lastSequence>
    <createdTime>2020-12-04T12:53:51.455+0900</createdTime>
    <startTime>2020-12-04T12:53:51.612+0900</startTime>
    <finishTime>2020-12-04T12:54:51.473+0900</finishTime>
  </file>
  <file>
    <transactionId>bcUCyJeKuOGnsah3</transactionId>
    <id>CTS_ID001</id>
    <vhost>default</vhost>
    <app>app</app>
    <stream>stream_o</stream>
    <filePath><![CDATA[/home/dev/OvenMediaEngine/records/bcUCyJeKuOGnsah3_default_app_stream_
    <recordBytes>2285797</recordBytes>
    <recordTime>60012</recordTime>
    <sequence>0</sequence>
    <schedule>0 */1 *</schedule>
    <lastSequence>false</lastSequence>
    <createdTime>2020-12-04T12:53:00.000+0900</createdTime>
    <startTime>2020-12-04T12:53:00.000+0900</startTime>
    <finishTime>2020-12-04T12:54:00.000+0900</finishTime>
  </file>
  <file>
    <transactionId>bcUCyJeKuOGnsah3</transactionId>
    <id>CTS_ID001</id>
    <vhost>default</vhost>
```

```
    <app>app</app>
    <stream>stream_o</stream>
    <filePath><![CDATA[/home/dev/OvenMediaEngine/records/bcUCyJeKuOGnsah3_default_app_stream_
    <recordBytes>4544626</recordBytes>
    <recordTime>60000</recordTime>
    <sequence>1</sequence>
    <schedule>0 */1 *</schedule>
    <lastSequence>true</lastSequence>
    <createdTime>2020-12-04T12:54:00.000+0900</createdTime>
    <startTime>2020-12-04T12:54:00.000+0900</startTime>
    <finishTime>2020-12-04T12:55:00.000+0900</finishTime>
  </file>
</files>
```

Push Publishing (Beta)

OvenMediaEngine supports **Push Publishing** function that can retransmit live streams to other systems. The protocol supported for retransmission uses RTMP or MPEGTS. Because, most services and products support this protocol. also, one output stream can be transmitted to multiple destinations at the same time. You can start and stop pushing the output stream through REST API. Note that the only codecs that can be retransmitted in RTMP and MPEGTS protocol are H264 and AAC.

Configuration

RTMPPush Publisher

To use RTMP Push Publishing, you need to declare the `<RTMPPush>` publisher in the configuration. There are no other detailed options.

```
<Applications>
  <Application>
     ...
    <Publishers>
      ...
      <RTMPPush>
      </RTMPPush>
    </Publishers>
  </Application>
</Applications>
```

MPEGTSPush Publisher

To use MPEGTS Push Publishing, you need to declare the `<MPEGTSPush>` publisher in the configuration. There are no other detailed options.

```
<Applications>
  <Application>
     ...
    <Publishers>
      ...
      <MPEGTSPush>
      </MPEGTSPush>
    </Publishers>
  </Application>
</Applications>
```

> ⓘ Only H264 and AAC are supported codecs.

Start & Stop Push

For control of push, use the REST API. RTMP, MPEGTS push can be requested based on the output stream name (specified in the JSON body), and you can selectively transfer all/some tracks. In addition, you must specify the URL and Stream Key of the external server to be transmitted. It can send multiple Pushes simultaneously for the same stream. If transmission is interrupted due to network or other problems, it automatically reconnects.

For how to use the API, please refer to the link below.

Push

REST API (Beta)

Overview

The REST APIs provided by OME allow you to query or change settings such as VirtualHost and Application/Stream.

> ⓘ The APIs are currently beta version, so there are some limitations/considerations.
>
> - Settings of VirtualHost can only be viewed and cannot be changed or deleted.
> - If you add/change/delete the settings of the App/Output Profile by invoking the API, the app will be restarted. This means that all sessions associated with the app will be disconnected.
> - The API version is fixed with v1 until the experimental stage is complete, and the detailed specification can be changed at any time.

By default, OvenMediaEngine's APIs are disabled, so the following settings are required to use the API:

Setting up for using the APIs

Port

Set the `<Port>` to use by the API server. If you omit `<Port>`, you will use the API server's default port, port `8081`.

```
<Server version="8">
        ...
        <Bind>
                <Managers>
                        <API>
                                <Port>8081</Port>
                                <!-- If you need TLS support, please uncomment below:

                                <TLSPort>8082</TLSPort>
                                -->
                        </API>
                </Managers>
                ...
        </Bind>
        ...
</Server>
```

Host and Permissions

`<Host>` sets the Host name and TLS certificate information to be used by the API server, and `<AccessToken>` sets the token to be used for authentication when calling the APIs. You must use this token to invoke the API of OvenMediaEngine.

```
<Server version="8">
        ...
        <Managers>
                <Host>
                        <Names>
                                <Name>*</Name>
                        </Names>
                        <!--
                                If you want to set up TLS, set it up by referring to the follo
                        <TLS>
                                <CertPath>airensoft_com.crt</CertPath>
                                <KeyPath>airensoft_com.key</KeyPath>
                                <ChainCertPath>airensoft_com_chain.crt</ChainCertPath>
                        </TLS>
                        -->
                </Host>
```

```
                <API>
                        <AccessToken>your_access_token</AccessToken>
                </API>
        </Managers>
</Server>
```

CrossDomains

If you face a CORS problem by calling the OME API on your browser, you can set `<CrossDomains>` as follows:

```
<Server version="10">
        ...
        <Managers>
                <Host>
                        <Names>
                                <Name>*</Name>
                        </Names>

                </Host>
                <API>
                ...
                        <CrossDomains>
                                <Url>*.airensoft.com</Url>
                                <Url>sub-domain.airensoft.com</Url>
                                <Url>http://http-only.airensoft.com</Url>
                        </CrossDomains>
                </API>
        </Managers>
</Server>
```

If protocol is omitted like `*.airensoft.com`, both HTTP and HTTPS are supported.

API Request/Response Specification

In this manual, the following format is used when describing the API.

GET http://<OME_HOST>:<API_PORT>
/<VERSION>/<API_PATH>[/...]

<VERSION>/<API_PATH>

Here is the description of the API

Request Example:
`- Method: GET`

```
- URL: http://1.2.3.4:8081/v1/vhost
- Header:
  authorization: Basic b21ldGVzdA==
```

Parameters

Path

string

Header

`authorization`	string

`Basic base64encode(AccessToken)`

if you set `<AccessToken>` to "ome-access-token" in `Server.xml`, you must set `Basic b21lLWFjY2Vzcy10b2tlbg==` in the `Authorization` header.

Responses

- **200**

<OME_HOST>

This means the IP or domain of the server on which your OME is running.

<API_PORT>

This means the port number of the API you set up in `Server.xml`. The default value is 8081.

<VERSION>

Indicates the version of the API. Currently, all APIs are v1.

<API_PATH>

Indicates the API path to be called. The API path is usually in the following form:

```
/resource-group[/resource[/resource-group[/resource/...]]][:action]
```

`resource` means an item, such as `VirtualHost` or `Application`, and `action` is used to command an action to a specific resource, such as `push` or `record`.

Response

All response results are provided in the HTTP status code and response body, and if there are multiple response results in the response, the HTTP status code will be `207 MultiStatus`. The API response data is in the form of an array of `Response` or `Response` as follows:

```
// Single data request example

// << Request >>
// Request URI: GET /v1/vhosts/default
// Header:
//   authorization: Basic b21lLWFjY2Vzcy10b2tlbg==

// << Response >>
// HTTP Status code: 200 OK
// Response Body:
{
        "statusCode": 200,
        "message": "OK",
        "response": ... // Requested data
}
```

```
// Multiple data request (Status codes are the same)

// << Request >>
// Request URI: POST /v1/vhost/default/apps
// Header:
//   authorization: Basic b21lLWFjY2Vzcy10b2tlbg==
// Request Body:
[
        { ... }, // App information to create
        { ... }, // App information to create
]

// << Response >>
// HTTP Status code: 200 OK
// Response Body:
[
        {
                "statusCode": 200,
                "message": "OK",
                "response": ... // App1
        },
        {
                "statusCode": 200,
                "message": "OK",
                "response": ... // App2
        }
]
```

```
// Multiple data request (Different status codes)

// << Request >>
// Request URI: POST /v1/vhost/default/apps
// Header:
//   authorization: Basic b21lLWFjY2Vzcy10b2tlbg==
// Request Body:
[
        { ... }, // App information to create
        { ... }, // App information to create
]

// << Response >>
// HTTP Status code: 207 MultiStatus
// Response Body:
[
        {
                "statusCode": 200,
                "message": "OK",
                "response": ... // App1
        },
        {
                "statusCode": 404,
                "message": "Not found"
        }
]
```

API Limitations

`VirtualHost` settings created by `Server.xml` cannot be modified through API. This rule also applies to `Application` / `OutputStream`, etc. within that VirtualHost. So, if you call a POST/PUT/DELETE API for `VirtualHost` / `Application` / `OutputProfile` declared in `Server.xml`, it will not work with a `403 Forbidden` error.

v1

Data Types

Primitives/Notations

Primitive data types

Type	Description	Examples
Short	16bits integer	`12345`
Int	32bits integer	`1234941932`
Long	64bits integer	`391859818923919232311`
Float	64bits real number	`3.5483`
String	A string	`"Hello"`
Bool	true/false	`true`
Timestamp (String)	A timestamp in ISO8601 format	`"2021-01-01T11:00:00.000+09:00"`
TimeInterval (Long)	A time interval (unit: milliseconds)	`349820`
IP (String)	IP address	`"127.0.0.1"`
RangedPort (String)	Port numbers with range (it can contain multiple ports and protocols) `start_port[-end_port][,start_port[-end_port][,start_port[-end_port]...]][/protocol]`	`"40000-40005/tcp"` `"40000-40005"` `"40000-40005,10000,20000/tcp"`
Port (String)	A port number `start_port[/protocol]`	`"1935/tcp"` `"1935"`

Enum/Container Notations

Enum<`Type`> (String)

- An enum value named `Type`
- Examples
  - `"value1"`
  - `"value2"`

List< `Type` >

- An ordered list of `Type`
- Examples
  - `[ Type, Type, ... ]`

Map< `KeyType` , `ValueType` >

- An object consisting of `Key` - `Value` pairs
- Examples
  - `{ Key1: Value1, Key2: Value2 }`

Enums

Enum< `ApplicationType` >

- Application type
- Examples
  - `"live"`
  - `"vod"`

Enum< `Codec` >

- Codecs
- Examples
  - `"h264"`
  - `"h265"`
  - `"vp8"`
  - `"opus"`
  - `"aac"`

Enum< `StreamSourceType` >

- A type of input stream
- Examples
  - `"Ovt"`
  - `"Rtmp"`
  - `"Rtspc"`
  - `"RtspPull"`
  - `"MpegTs"`

Enum<`MediaType`>

- type
- Examples
  - `"video"`
  - `"audio"`

Enum<`SessionState`>

- A state of the session
- Examples
  - `"Ready"`
  - `"Started"`
  - `"Stopping"`
  - `"Stopped"`
  - `"Error"`

Enum<`AudioLayout`>

- Audio layout
- Examples
  - `"stereo"`
  - `"mono"`

Classes

Response<`[TYPE]`>

Type	Name	Optional	Description	Examples
Int	statusCode	N	Status code	`200`
String	message	N	A message describing the value returned	`"OK"`
`[TYPE]`	response	Y	A response data	`{}`

VirtualHost

Type	Name	Optional	Description	Examples
String	name	N	A name of Virtual Host	`"default"`
`Host`	td	Y	Host	`Host`
`SignedPolicy`	signedPolicy	Y	SignedPolicy	`SignedPolicy`
`SignedToken`	signedToken	Y	SignedToken	`SignedToken`
List< `OriginMap` >	td	Y	A list of Origin map	`[OriginMap, OriginMap, ...]`

Host

Type	Name	Optional	Description	Examples
List<String>	td	N	A list of hosts	`["airensoft.com", "*.test.com", ...]`
Tls	td	Y	TLS	`TLS`

TLS

Type	Name	Optional	Description	Examples
String	certPath	N	A path of cert file	`"a.crt"`
String	keyPath	N	A path of private key file	`"a.key"`
String	chainCertPath	Y	A path of chain cert file	`"c.crt"`

SignedPolicy

Type	Name	Optional	Description	Examples
String	policyQueryKey	N		
String	signatureQueryKey	N		
String	secretKey	N		

SignedToken

Type	Name	Optional	Description	Examples
String	cryptoKey	N		
String	queryStringKey	N		

OriginMap

Type	Name	Optional	Description	Examples
String	location	N	A pattern to map origin	`"/"`
`Pass`	pass	N	What to request with Origin if the pattern matches	`Pass`

Pass

Type	Name	Optional	Description	Examples
String	scheme	N	Scheme to distinguish the provider	`"ovt"`
List<String>	urls	N	An address list to pull from provider	`["origin:900", "origin2:9000", ...]`

Application

Type	Name	Optional	Description	Examples
String	name	N	App name (You cannot change this value after you create it)	`"app"`
Bool	dynamic	N	Whether the app was created using `PullStream()`	`true`
Enum<`ApplicationType`>	type	N	App type	`"live"`
`Providers`	providers	Y	A list of `Provider`s	`Providers`
`Publishers`	publishers	Y	A list of `Publisher`s	`Publishers`
List<`OutputProfile`>	outputProfiles	Y	A list of `OutputProfile`s	`[OutputProfile, OutputProfie, ...]`

Providers

Type	Name	Optional	Description	Examples
`RtmpProvider`	rtmp	Y		`RtmpProvider`
`RtspPullProvider`	rtspPull	Y		`RtspPullProvider`
`RtspProvider`	rtsp	Y		`RtspProvider`
`OvtProvider`	ovt	Y		`OvtProvider`
`MpegtsProvider`	mpegts	Y		`MpegtsProvider`

RtmpProvider

Type	Name	Optional	Description	Examples
(Reserved for future use)	-	-	-	

RtspPullProvider

Type	Name	Optional	Description	Examples
(Reserved for future use)	-	-	-	

RtspProvider

Type	Name	Optional	Description	Examples
(Reserved for future use)	-	-	-	

OvtProvider

Type	Name	Optional	Description	Examples
(Reserved for future use)	-	-	-	

MpegtsProvider

Type	Name	Optional	Description	Examples
List<`MpegtsStream`>	streams	Y	MPEG-TS Stream map	`[MpegtsStream, MpegtsStrea, ...]`

MpegtsStream

Type	Name	Optional	Description	Examples
String	name	N	A name to generate when MPEG-TS stream is received	`"stream"`
`RangedPort`	port	Y	MPEG-TS Port	`"40000-40001/udp"`

Publishers

Type	Name	Optional	Description	Examples
Int	threadCount	N	Number of threads	`4`

`RtmpPushPublisher`	rtmpPush	Y		`RtmpPushPublisher`
`HlsPublisher`	hls	Y		`HlsPublisher`
`DashPublisher`	dash	Y		`DashPublisher`
`LlDashPublisher`	llDash	Y		`LlDashPublisher`
`WebrtcPublisher`	webrtc	Y		`WebrtcPublisher`
`OvtPublisher`	ovt	Y		`OvtPublisher`
`FilePublisher`	file	Y		`FilePublisher`
`ThumbnailPublisher`	thumbnail	Y		`ThumbnailPublisher`

RtmpPushPublisher

Type	Name	Optional	Description	Examples
(Reserved for future use)	-	-	-	

HlsPublisher

Type	Name	Optional	Description	Examples
Int	segmentCount	N	Segment count in the playlist.m3u8	`3`
Int	segmentDuration	N	Segment duration (unit: seconds)	`4`
List<String>	crossDomains	Y	Cross domain	`["*"]`

			URLs	

DashPublisher

Type	Name	Optional	Description	Examples
Int	segmentCount	N	Segment count in the manifest.mpd	`3`
Int	segmentDuration	N	Segment duration (unit: seconds)	`4`
List<String>	crossDomains	Y	Cross domain URLs	`["*"]`

LlDashPublisher

Type	Name	Optional	Description	Examples
Int	segmentDuration	N	Segment duration (unit: seconds)	`3`
List<String>	crossDomains	Y	Cross domain URLs	`["*"]`

WebrtcPublisher

Type	Name	Optional	Description	Examples
`TimeInterval`	timeout	Y	ICE timeout (unit: seconds)	`30`

OvtPublisher

Type	Name	Optional	Description	Examples
(Reserved for future use)	-	-	-	

FilePublisher

Type	Name	Optional	Description	Examples
String	filePath	Y	A path to store recorded file You can use the following macros: `${TransactionId}`: An identifier of transaction `${Id}`: An identifier to distinguish files `${StartTime:YYYYMMDDhhmmss}`: Start time of recording `${EndTime:YYYYMMDDhhmmss}`: End time of of recording `${VirtualHost}`: A name of virtual host `${Application}`: A name of application `${SourceStream}`: A name of input	`"/tmp/${StartTime:YYYYMMDDhhmmss}_${Stream}.mp4`

			`stream ${Stream}: A name of output stream ${Sequence}: A sequence number`	
String	fileInfoPath	Y	A path of recorded files	`"/tmp/${StartTime:YYYYMDDhhmmss}_${Stream}.xml`

ThumbnailPublisher

Type	Name	Optional	Description	Examples
List<String>	crossDomains	Y	Cross domain URLs	`["*"]`

OutputProfile

Type	Name	Optional	Description	Examples
String	name	N	A name of `OutputProfile`	`"bypass_stream"`
String	outputStreamName	N	A name of output stream	`"${OriginStreamName}"`
`Encodes`	encodes	Y		`[Encodes, Encodes, ...]`

Encodes

Type	Name	Optional	Description	Examples
List<`Video`>	videos	Y		`[Video, Video, ...]`
List<`Audio`>	audios	Y		`[Audio, Audio, ...]`
List<`Image`>	images	Y		`[Image, Image, ...]`

Video

Type	Name	Optional	Description	Examples
Bool	bypass	Y		`true`
Enum<`Codec`>	codec	Conditional	Video codec	`"h264"`
Int	width	Conditional		`1280`
Int	height	Conditional		`720`
String	bitrate	Conditional	bitrate (You can use "K" or "M" suffix like `100K`, `3M`)	`"3000000"` `"2.5M"`
Float	framerate	Conditional		`29.997`

Audio

Type	Name	Optional	Description	Examples
Bool	bypass	Y		`true`
Enum<`Codec`>	codec	Conditional	Audio codec	`"opus"`
Int	samplerate	Conditional		`48000`
Int	channel	Conditional		`2`
			bitrate (You can	

String	bitrate	Conditional	use "K" or "M" suffix like `128K`, `0.1M`)	`"128000"` `"128K"`

Image

Type	Name	Optional	Description	Examples
Enum<`Codec`>	codec	N		`"jpeg" | "png"`
Int	width	Conditional		`854`
Int	height	Conditional		`480`
Float	framerate	N	An interval of image creation	`1`

Stream

Type	Name	Optional	Description	Examples
String	name	N	A name of stream	`"stream"`
`InputStream`	input	N	An information of input stream	`InputStream`
List<`OutputStream`>	outputs	N	An information of output streams	`[OutputStream, OutputStrea, ...]`

NewStream

Type	Name	Optional	Description	Examples
String	name	N	A name of stream to create	`"stream"`
`PullStream`	pull	Y	pull	`PullStream`

`MpegtsStream`	mpegts	Y	Creates a `prestream`	`MpegtsStream`

PullStream

Type	Name	Optional	Description	Examples
String	url	N	URL to pull	`"rtsp://host.com/resource"`

InputStream

Type	Name	Optional	Description	Examples
String	agent	Y	A name of broadcast tool	`"OBS 12.0.4"`
String	from	N	URI stream created	`"tcp://192.168.0.200:3399"`
String	to	Y	URI represents connection with the input	`"rtmp://devairensoft.com:1935"`
List< `Track` >	tracks	N	A list of tracks in input stream	`[Track, Track, ...]`
`Timestamp`	createdTime	N	Creation time	`"2020-10-30T11:00:0009:00"`

OutputStream

Type	Name	Optional	Description	Examples
String	name	N	An name of `OutputStrea`	`"stream_o"`

			`m`	
List<`Track`>	tracks	N	A list of tracks in `OutputStream`	`[Track, Track, ...]`

Track

Type	Name	Optional	Description	Examples
Enum<`MediaType`>	type	Y	Media type	`"video"`
`Video`	video	Conditional	A configuration of video encoding	`Video`
`Audio`	audio	Conditional	A configuration of audio encoding	`Audio`

VideoTrack

Type	Name	Optional	Description	Examples
(Extends `Video`)	-	-		
`Timebase`	timebase	Y	Timebase	`Timebase`

Timebase

Type	Name	Optional	Description	Examples
Int	num	N	Numerator	`1`
Int	den	N	Denominator	`90000`

AudioTrack

Type	Name	Optional	Description	Examples
(Extends `Audio`)	-	-		`true`
`Timebase`	timebase	Y	Timebase	`Timebase`

Record

Type	Name	Optional	Description	Examples
String	id	Y	Unique identifier	
`OutputStream`	streams	N	A combination of output stream's track name and track id	
Enum<`SessionState`>	state	N	Record state	
String	filePath	N	A path of recorded files	
String	fileInfoPath	N	A path of recorded file informations	
String	recordedBytes	N	Recorded bytes	
Int	recordedTime	N	Recorded time	
`Timestamp`	startTime	N	Started time	
`Timestamp`	finishTime	N	Finished time	
Int	bitrate	N	Average bitrate	

Push

Type	Name	Optional	Description	Examples

String	id	N	Unique identifier	
`OutputStream`	stream	Y	A combination of output stream's track name and track id	
Enum<`StreamSourceType`>	protocol	Y	Protocol of input stream	
String	url	Y	Destination URL	
String	streamKey	Conditional	Stream key of destination	
Enum<`SessionState`>	state	N	Push state	
Int	sentBytes	N	Sent bytes	
Int	sentPackets	N	Sent packets count	
Int	sentErrorBytes	N	Error bytes	
Int	sentErrorPackets	N	Error packets count	
Int	reconnect	N	Reconnect count	
`Timestamp`	startTime	N	Started time	
`Timestamp`	finishTime	N	Finished time	
Int	bitrate	N	Average bitrate	

CommonMetrics

Type	Name	Optional	Description	Examples
`Timestamp`	createdTime	N	Creation time	`"2020-10-30T11:00:0009:00"`
`Timestamp`	lastUpdatedTime	N	Modified time	`"2020-10-30T11:00:0009:00"`

Long	totalBytesIn	N	Received bytes	`3109481213`
Long	totalBytesOut	N	Sent bytes	`1230874123`
Int	totalConnections	N	Current connections	`10`
Int	maxTotalConnections	N	Max connections since the stream is created	`293`
`Timestamp`	maxTotalConnectionTime	N	When the maximum number of concurrent connections has been updated.	`"2020-10-30T11:00:0009:00"`
`Timestamp`	lastRecvTime	N	Last time data was received	`"2020-10-30T11:00:0009:00"`
`Timestamp`	lastSentTime	N	Last time data was sent	`"2020-10-30T11:00:0009:00"`

StreamMetrics

Type	Name	Optional	Description	Examples
(Extends `CommonMetrics`)	-	-	Includes all fields of `CommonMetrics`	
`TimeInterval`	requestTimeToOrigin	Y	A elapsed time to connect to Origin	`1000`
`TimeInterval`	responseTimeFromOrigin	Y	A elapsed time from Origin to respond	`10000`

VirtualHost

POST http://<OME_HOST>:<API_PORT>

/v1/vhosts

/v1/vhosts

Creates `VirtualHost`s

Request Example:
`POST http://1.2.3.4:8081/v1/vhosts`

[{ "name": "default" }]

Parameters

Header

`authorization` string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`(json body)`* array

A list of `VirtualHost`

Responses

- **200: OK**
  Returns the specified virtual host information
- **404: Not Found**
  The specified VirtualHost was not found.
- **409: Conflict**
  The virtual host already exists.

GET http://<OME_HOST>:<API_PORT>

/v1/vhosts

/v1/vhosts

Lists all virtual host names

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts`

Parameters

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

- **200: OK**

Returns a list of virtual host names

GET http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}

/v1/vhosts/{vhost_name}

Gets the configuration of the `VirtualHost`

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts/default`

Parameters

Path

`vhost_name`	string

A name of `VirtualHost`

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

- **200**

- Return type: Response<VirtualHost>
- Description

Returns the specified virtual host information

- **404**

- Return type: Response<>

- Description
The specified VirtualHost was not found.

Application

POST http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps

/v1/vhosts/{vhost_name}/apps

Creates `Application`s in the `VirtualHost`

Request Example:
`POST http://1.2.3.4:8081/v1/vhosts/default/apps`
[{ "name": "app", "type": "live", "outputProfiles": { "outputProfile": [{ "name": "bypass_profile", "outputStreamName": "${OriginStreamName}", "encodes": { "videos": [{ "bypass": true }], "audios": [{ "bypass": true }] } }] }]

Parameters

Path

`vhost_name`	string

A name of `VirtualHost`

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`(json body)`	array

A list of `Application`

Responses

200
- Return type: List<Response<Application>>
- Description
Returns a list of created application informations

404
- Return type: Response<>

- Description
Not Found

GET http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps

/v1/vhosts/{vhost_name}/apps

Lists all application names in the `VirtualHost`

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts/default/apps`

Parameters

Path

`vhost_name`	string

A name of VirtualHost

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

200
- Return type: Response<List<string>>
- Description
Returns a list of application names

404
- Return type: Response<>
- Description
Not Found

GET http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}

/v1/vhosts/{vhost_name}/apps/{app_name}

Gets the configuration of the `Application`

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts/default/apps/app`

Parameters

Path

`vhost_name`	string

A name of `VirtualHost`

`app_name`	string

A name of `Application`

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

- **200**
  - Return type: Response<Application>
  - Description

  Returns the specified application information

- **404**
  - Return type: Response<>
  - Description

  Not Found

PUT http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}

/v1/vhosts/{vhost_name}/apps/{app_name}

Changes the configuration of the `Application`

Request Example:
`PUT http://1.2.3.4:8081/v1/vhosts/default/apps/app`

```
{
  "type": "live"
}
```

Parameters

Path

`vhost_name`	string
A name of `VirtualHost`	
`app_name`	string
A name of `Application`	

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`(json body)`	object

`Application`

Responses

200
- Return type: Response<Application>
- Description
Returns the modified application information

404
- Return type: Response<>
- Description
Not Found

DELETE http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}

/v1/vhosts/{vhost_name}/apps/{app_name}

Deletes the `Application`

Request Example:
`DELETE http://1.2.3.4:8081/v1/vhosts/default/apps/app`

Parameters

Path

`vhost_name`	string
A name of `VirtualHost`	
`app_name`	string

A name of `Application`

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

200
- Return type: Response<>
- Description
Returns the result

404
- Return type: Response<>
- Description
Not Found

Stream

GET http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}/streams

/v1/vhosts/{vhost_name}/apps/{app_name}/streams

Lists all stream names in the `Application`

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts/default/apps/app/streams`

Parameters

Path

`vhost_name`	string

A name of `VirtualHost`

`app_name`	string

A name of `Application`

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

200
- Return type: Response<List<string>>
- Description
Returns a list of stream names

404
- Return type: Response<>
- Description
Not Found

GET http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}/streams/{stream_name}

/v1/vhosts/{vhost_name}/apps/{app_name}/streams/{stream_name}

Gets the configuration of the `Stream`

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts/default/apps/app/streams/stream`

Parameters

Path

`vhost_name`	string
A name of `VirtualHost`	
`app_name`	string
A name of `Application`	
`stream_name`	string
A name of `Stream`	

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

200
- Return type: Response<Stream>

- Description
Returns the specified stream information

404
- Return type: Response<>
- Description
Not Found

Output Profile

POST http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles

/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles

Creates `OutputProfile`s in the `Application`

Request Example:
`POST http://1.2.3.4:8081/v1/vhosts/default/apps/app/outputProfiles`

```
[
  {
    "name": "bypass_profile",
    "outputStreamName": "${OriginStreamName}",
    "encodes": {
      "videos": [
        {
          "bypass": true
        }
      ],
      "audios": [
        {
          "bypass": true
        }
      ]
    }
  }
]
```

Parameters

Path

`vhost_name` A name of `VirtualHost`	string
`app_name` A name of `Application`	string

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`(json body)`	array

List< `OutputProfile` >

Responses

- **200**
  - Return type: List<Response<OutputProfile>>
  - Description

  Returns a list of created output profile informations

- **404**
  - Return type: Response<>
  - Description

  Not Found

GET http://<OME_HOST>:<API_PORT>

/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles

/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles

Lists all output profile names in the `Application`

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts/default/apps/app/outputProfiles`

Parameters

Path

`vhost_name` A name of `VirtualHost`	string
`app_name` A name of `Application`	string

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

200
- Return type: Response<List<string>>
- Description
Returns a list of output profile names

404
- Return type: Response<>
- Description
Not Found

GET http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles/{profile_name}

/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles/{profile_name}

Gets the configuration of the `OutputProfile`

Request Example:
`GET http://1.2.3.4:8081/v1/vhosts/default/apps/app/outputProfiles/bypass_profile`

Parameters

Path

`vhost_name`	string

A name of `VirtualHost`

`app_name`	string

A name of `Application`

`profile_name`	string

A name of `OutputProfile`

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

200
- Return type: Response<OutputProfile>
- Description
Returns the specified output profile information

404
- Return type: Response<>
- Description
Not Found

PUT http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles/{profile_name}

/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles/{profile_name}

Changes the configuration of the `OutputProfile`

Request Example:
`PUT http://1.2.3.4:8081/v1/vhosts/default/apps/app/outputProfiles/bypass_profile`

```
{
  "outputStreamName": "${OriginStreamName}",
  "encodes": {
    "videos": [
      {
        "codec": "h264",
        "bitrate": "3M",
        "width": 1280,
        "height": 720,
        "framerate": 30
      }
    ],
    "audios": [
      {
        "bypass": true
      }
    ]
  }
}
```

Parameters

Path

`vhost_name`	string

A name of `VirtualHost`

`app_name`	string

A name of `Application`

`profile_name`	string

A name of `OutputProfile`

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`(json body)`	object

`OutputProfile`

Responses

- **200**
- Return type: Response<OutputProfile>
- Description

Returns the modified output profile information

- **404**
- Return type: Response<>
- Description

Not Found

DELETE http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles/{profile_name}

/v1/vhosts/{vhost_name}/apps/{app_name}/outputProfiles/{profile_name}

Deletes the `OutputProfile`

Request Example:
`DELETE http://1.2.3.4:8081/v1/vhosts/default/apps/app/outputProfiles/bypass_profile`

Parameters

Path

`vhost_name`	string
A name of `VirtualHost`	
`app_name`	string
A name of `Application`	
`profile_name`	string
A name of `OutputProfile`	

Header

`authorization`	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Responses

- **200**
  - Return type: Response<>
  - Description
  Returns the result

- **404**
  - Return type: Response<>
  - Description
  Not Found

Recording

POST http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}:startRecord

/v1/vhosts/{vhost_name}/apps/{app_name}:startRecord

This API performs a recording start request operation. for recording, the output stream name must be specified. file path, information path, recording interval and schedule parameters can be specified as options.

Request Example:

```
POST http://1.2.3.4:8081/v1/vhosts/default/apps/app:startRecord
{
  "id": "custom_id",
  "stream": {
```

```
    "name": "stream_o",
    "tracks": [ 100, 200 ]
  },
  "filePath" : "/path/to/save/recorded/file_${Sequence}.ts",
  "infoPath" : "/path/to/save/information/file.xml",
  "interval" : 60000,    # Split it every 60 seconds
  "schedule" : "0 0 */1" # Split it at second 0, minute 0, every hours.
  "segmentationRule" : "continuity"

}
```

Parameters

Path

`vhost_name` string

A name of `VirtualHost`

`app_name` string

A name of `Application`

Header

`authorization` string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`

Body

`segmentationRule` string

Define the policy for continuously or discontinuously generating timestamp in divided recorded files.

- continuity
- discontinuity (default)

`id` string

An unique identifier for recording job.

`stream` string

Output stream.

`name` string

Output stream name.

`tracks` array

Default is all tracks. It is possible to record only a specific track using the track Id.

- default is all tracks

`schedule` string

Schedule based split recording. set only <second minute hour> using crontab method.

It cannot be used simultaneously with interval.

`interval` number

Interval based split recording. It cannot be used simultaneously with schedule.

`filePath` string

Set the path of the file to be recorded. same as setting macro pattern in Config file.

`infoPath` string

Set the path to the information file to be recorded. same as setting macro pattern in Config file.

Responses

- **200**
- **400**

POST http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}:stopRecord

/v1/vhosts/{vhost_name}/apps/{app_name}:stopRecord

This API performs a recording stop request.

Request Example:

```
POST http://1.2.3.4:8081/v1/vhosts/default/apps/app:stopRecord
{
  "id": "custom_id"
}
```

Parameters

Path

`vhost_name` string

A name of `VirtualHost`

`app_name` string

A name of `Application`

Header

`authorization` string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`id` string

An unique identifier for recording job.

Responses

- **200**
- **400**
- **404**

POST http://<OME_HOST>:<API_PORT>

/v1/vhosts/{vhost_name}/apps/{app_name}:records

/v1/vhosts/{vhost_name}/apps/{app_name}:records

This API performs a query of the job being recorded. Provides job inquiry function for all or custom Id.

Request Example:

```
POST http://1.2.3.4:8081/v1/vhosts/default/apps/app:records
{
   "id" : "custom_id"
}
```

Parameters

Path

`vhost_name` string

A name of `VirtualHost`

`app_name` string

A name of `Application`

Header

`authorization` string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`id` string

An unique identifier for recording job. If no value is specified, the entire recording job is requested.

Responses

- **200**

204

Push

POST http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}:startPush

/v1/vhosts/{vhost_name}/apps/{app_name}:startPush

This is an action to request a push of a selected stream. Please refer to the "Push" document for detail setting.

\

\

Request Example:

\

```
POST http://1.2.3.4:8081/v1/vhosts/default/apps/app:startPush
```

\

```
{
```

\

```
"id": "{UserDefinedUniqueId}",
```

\

```
"stream": {
```

\

```
"name": "output_stream_name",
```

\

```
"tracks": [ 101, 102 ]
```

\

```
},
```

\

```
"protocol": "rtmp",
```

```
\
"url":"rtmp://{host}[:port]/{appName}",
\
"streamKey":"{streamName}"
\
}
POST http://1.2.3.4:8081/v1/vhosts/default/apps/app:startPush
\
{
\
"id": "{UserDefinedUniqueId}",
\
"stream": {
\
"name": "output_stream_name",
\
"tracks": [ 101, 102 ]
\
},
\
"protocol": "mpegts",
\
"url":"udp://{host}[:port]",
\
"streamKey":""
\
}
```

Parameters

Path

`vhost_name`* — string

A name of `VirtualHost`

`app_name`* — string

A name of `Application`

Header

`authorization`* — string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`

Body

`id`* — string

Unique identifier for push management. if there is no value, automatically created and returned

`stream`* — string

Output stream for push

`name`* — string

Output stream name

`tracks` — string

Track id for want to push, if there is no value, all tracks are push

`protocol`* — string

Transport protocol [rtmp | mpegts]

`url`* — string

Destination URL

`streamKey`* — object

Destination stream key

Responses

200
Success

400
Invalid Parameters

POST http://<OME_HOST>:<API_PORT>

/v1/vhosts/{vhost_name}/apps/{app_name}:stopPush

/v1/vhosts/{vhost_name}/apps/{app_name}:stopPush

Request to stop pushing

\

\

Request Example:

\

`POST http://1.2.3.4:8081/v1/vhosts/default/apps/app:stopRecord`

\

``

\

`{`

\

`"id": "{userDefinedUniqueId}"`

\

`}`

Parameters

Path

`vhost_name`*	string
A name of `VirtualHost`	
`app_name`*	string
A name of `Application`	

Header

`authorization`*	string

A string for authentication in `Basic Base64(AccessToken)` format.
For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is `ome-access-token`.

Body

`id`*	string

Unique identifier for push management

Responses

- **200**
  Success
- **400**
  Invalid Parameters
- **404**
  No content

POST http://<OME_HOST>:<API_PORT>
/v1/vhosts/{vhost_name}/apps/{app_name}:pushes

/v1/vhosts/{vhost_name}/apps/{app_name}:pushes

Get all push lists for a specific application

\

\

Request Example:

\

```
POST http://1.2.3.4:8081/v1/vhosts/default/apps/app:pushes
```

Parameters

Path

`vhost_name`*	string
A name of `VirtualHost`	
`app_name`*	string
A name of `Application`	

Header

`authorization`*	string
A string for authentication in `Basic Base64(AccessToken)` format. \ For example, `Basic b21lLWFjY2Vzcy10b2tlbg==` if access token is	

`ome-access-token`

Responses

- **200**
  Success
- **204**
  Not Found

Statistics

Current

GET http://<OME_HOST>:<API_PORT>
/v1/stats/current/vhosts/{vhost_name}

/v1/stats/current/vhosts/{vhost_name}

Usage statistics of the `VirtualHost`

Request Example:
`GET http://1.2.3.4:8081/v1/stats/current/vhosts/default`

Parameters

Path

`vhost_name`	string

A name of `VirtualHost`

Query

`access_token`	string

A token for authentication

Responses

- **200**
  Returns a statistics of the VirtualHost.
- **404**

Not Found

GET http://<OME_HOST>:<API_PORT>
/v1/stats/current/vhosts/{vhost_name}/apps/{app_name}

/v1/stats/current/vhosts/{vhost_name}/apps/{app_name}

Usage statistics of the `Application`

Request Example:
`GET http://1.2.3.4:8081/v1/stats/current/vhosts/default/apps/app`

Parameters

Path

`vhost_name`	string
A name of `VirtualHost`	
`app_name`	string
A name of `Application`	

Query

`access_token`	string
A token for authentication	

Responses

- **200**
  Returns a statistics of the Application.
- **404**
  Not Found

GET http://<OME_HOST>:<API_PORT>
/v1/stats/current/vhosts/{vhost_name}/apps/{app_name}/streams/{stream}

/v1/stats/current/vhosts/{vhost_name}/apps/{app_name}/streams/{stream}

Usage statistics of the `Stream`

Request Example:
`GET`

`http://1.2.3.4:8081/v1/stats/current/vhosts/default/apps/app/streams/{stream}`

Parameters

Path

`vhost_name`	string
A name of `VirtualHost`	
`app_name`	string
A name of `Application`	
`stream_name`	string
A name of `Stream`	

Query

`access_token`	string
A token for authentication	

Responses

- **200** Returns a statistics of the Stream.
- **404** Not Found

Performance Tuning

Performance Test

OvenMediaEngine provides a tester for measuring WebRTC performance called OvenRtcTester. It is developed in Go language and uses the pion/webrtc/v3 and gorilla/websocket modules. Many thanks to the pion/webrtc and gorilla/websocket teams for contributing this wonderful project.

Getting Started OvenRtcTester

Install GO

Since OvenRtcTester is developed in Go language, Go must be installed on your system. Install Go from the following URL: https://golang.org/doc/install

OvenRtcTester was tested with the latest version of go 1.17.

Run

You can simply run it like this: -url is required. If the -life option is not used, it will run indefinitely until the user presses `ctrl+c`.

```
$ cd OvenMediaEngine/misc/oven_rtc_tester
$ go run OvenRtcTester.go
-url parameter is required and must be vaild. (input : undefined)
  -cint int
        [Optional] PeerConnection connection interval (milliseconds) (default 100)
  -life int
        [Optional] Number of times to execute the test (seconds)
  -n int
        [Optional] Number of client (default 1)
  -sint int
        [Optional] Summary information output cycle (milliseconds) (default 5000)
  -url string
        [Required] OvenMediaEngine's webrtc streaming URL (default "undefined")
```

You can also use `go build` or `go install` depending on your preference.

> OvenRtcTester must test OvenMediaEngine 0.12.4 or higher as the target system. OvenMediaEngine versions below 0.12.4 have a problem with incorrectly calculating the RTP timestamp, so OvenRtcTester calculates the `Video Delay` value incorrectly.

```
$ go run OvenRtcTester.go -url ws://192.168.0.160:13333/app/stream -n 5
client_0 connection state has changed checking
client_0 has started
client_1 connection state has changed checking
client_1 has started
client_0 connection state has changed connected
client_1 connection state has changed connected
client_1 track has started, of type 100: video/H264
client_0 track has started, of type 100: video/H264
client_1 track has started, of type 101: audio/OPUS
client_0 track has started, of type 101: audio/OPUS
client_2 connection state has changed checking
client_2 has started
client_2 connection state has changed connected
client_2 track has started, of type 100: video/H264
client_2 track has started, of type 101: audio/OPUS
client_3 connection state has changed checking
client_3 has started
client_3 connection state has changed connected
client_3 track has started, of type 100: video/H264
client_3 track has started, of type 101: audio/OPUS
client_4 connection state has changed checking
client_4 has started
```

```
client_4 connection state has changed connected
client_4 track has started, of type 100: video/H264
client_4 track has started, of type 101: audio/OPUS
<Summary>
Running time : 5s
Number of clients : 5
ICE Connection State : New(0), Checking(0) Connected(5) Completed(0) Disconnected(0) Failed(0)
Avg Video Delay(54.20 ms) Max Video Delay(55.00 ms) Min Video Delay(53.00 ms)
Avg Audio Delay(37.00 ms) Max Audio Delay(55.00 ms) Min Audio Delay(26.00 ms)
Avg FPS(30.15) Max FPS(30.25) Min FPS(30.00)
Avg BPS(4.1 Mbps) Max BPS(4.1 Mbps) Min BPS(4.0 Mbps)
Total Bytes(11.6 MBytes) Avg Bytes(2.3 MBytes)
Total Packets(13897) Avg Packets(2779)
Total Packet Losses(0) Avg Packet Losses(0)

<Summary>
Running time : 10s
Number of clients : 5
ICE Connection State : New(0), Checking(0) Connected(5) Completed(0) Disconnected(0) Failed(0)
Avg Video Delay(43.60 ms) Max Video Delay(45.00 ms) Min Video Delay(42.00 ms)
Avg Audio Delay(36.60 ms) Max Audio Delay(55.00 ms) Min Audio Delay(25.00 ms)
Avg FPS(30.04) Max FPS(30.11) Min FPS(30.00)
Avg BPS(4.0 Mbps) Max BPS(4.0 Mbps) Min BPS(4.0 Mbps)
Total Bytes(24.3 MBytes) Avg Bytes(4.9 MBytes)
Total Packets(28832) Avg Packets(5766)
Total Packet Losses(0) Avg Packet Losses(0)

<Summary>
Running time : 15s
Number of clients : 5
ICE Connection State : New(0), Checking(0) Connected(5) Completed(0) Disconnected(0) Failed(0)
Avg Video Delay(36.60 ms) Max Video Delay(38.00 ms) Min Video Delay(35.00 ms)
Avg Audio Delay(49.20 ms) Max Audio Delay(68.00 ms) Min Audio Delay(38.00 ms)
Avg FPS(30.07) Max FPS(30.07) Min FPS(30.07)
Avg BPS(4.0 Mbps) Max BPS(4.0 Mbps) Min BPS(4.0 Mbps)
Total Bytes(36.8 MBytes) Avg Bytes(7.4 MBytes)
Total Packets(43717) Avg Packets(8743)
Total Packet Losses(0) Avg Packet Losses(0)

^CTest stopped by user
**************************
Reports
**************************
<Summary>
Running time : 15s
Number of clients : 5
ICE Connection State : New(0), Checking(0) Connected(5) Completed(0) Disconnected(0) Failed(0)
Avg Video Delay(23.60 ms) Max Video Delay(25.00 ms) Min Video Delay(22.00 ms)
Avg Audio Delay(11.20 ms) Max Audio Delay(18.00 ms) Min Audio Delay(5.00 ms)
Avg FPS(30.07) Max FPS(30.07) Min FPS(30.07)
Avg BPS(4.0 Mbps) Max BPS(4.0 Mbps) Min BPS(4.0 Mbps)
Total Bytes(38.6 MBytes) Avg Bytes(7.7 MBytes)
Total Packets(45662) Avg Packets(9132)
```

```
Total Packet Losses(0) Avg Packet Losses(0)
<Details>
[client_0]
        running_time(15s) connection_state(connected) total_packets(9210) packet_loss(0)
        last_video_delay (22.0 ms) last_audio_delay (52.0 ms)
        total_bytes(7.8 Mbytes) avg_bps(4.0 Mbps) min_bps(3.6 Mbps) max_bps(4.3 Mbps)
        total_video_frames(463) avg_fps(30.07) min_fps(28.98) max_fps(31.00)

client_0 connection state has changed closed
client_0 has stopped
[client_1]
        running_time(15s) connection_state(connected) total_packets(9210) packet_loss(0)
        last_video_delay (22.0 ms) last_audio_delay (52.0 ms)
        total_bytes(7.8 Mbytes) avg_bps(4.0 Mbps) min_bps(3.6 Mbps) max_bps(4.3 Mbps)
        total_video_frames(463) avg_fps(30.07) min_fps(28.98) max_fps(31.00)

client_1 has stopped
[client_2]
        running_time(15s) connection_state(connected) total_packets(9145) packet_loss(0)
        last_video_delay (23.0 ms) last_audio_delay (63.0 ms)
        total_bytes(7.7 Mbytes) avg_bps(4.0 Mbps) min_bps(3.6 Mbps) max_bps(4.5 Mbps)
        total_video_frames(460) avg_fps(30.07) min_fps(28.97) max_fps(31.02)

client_1 connection state has changed closed
client_2 has stopped
[client_3]
        running_time(15s) connection_state(connected) total_packets(9081) packet_loss(0)
        last_video_delay (25.0 ms) last_audio_delay (65.0 ms)
        total_bytes(7.7 Mbytes) avg_bps(4.0 Mbps) min_bps(3.6 Mbps) max_bps(4.3 Mbps)
        total_video_frames(457) avg_fps(30.07) min_fps(29.00) max_fps(31.03)

client_2 connection state has changed closed
client_3 has stopped
client_3 connection state has changed closed
[client_4]
        running_time(15s) connection_state(connected) total_packets(9016) packet_loss(0)
        last_video_delay (26.0 ms) last_audio_delay (36.0 ms)
        total_bytes(7.6 Mbytes) avg_bps(4.0 Mbps) min_bps(3.6 Mbps) max_bps(4.3 Mbps)
        total_video_frames(454) avg_fps(30.07) min_fps(28.99) max_fps(31.02)

client_4 has stopped
```

Performance Tuning

Monitoring the usage of threads

Linux has various tools to monitor CPU usage per thread. We will check the simplest with the top command. If you issue the top -H -p [pid] command, you will see the following screen.

```
top - 21:37:55 up 2 days, 23:31,  2 users,  load average: 10.87, 4.35, 1.78
Threads:  41 total,   0 running,  41 sleeping,   0 stopped,   0 zombie
%Cpu(s): 68.2 us,  4.3 sy,  0.0 ni, 27.3 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem : 128796.8 total,   5753.5 free,  36148.0 used,  86895.3 buff/cache
MiB Swap:  31250.0 total,  31248.7 free,      1.2 used.  91450.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
3489038 getroot   20   0 2074776 219132  16860 S  25.7   0.2   0:26.89 StreamWorker
3350981 getroot   20   0 2074776 219132  16860 S   9.9   0.2   0:11.86 SPICE
3351004 getroot   20   0 2074776 219132  16860 S   2.0   0.2   0:16.86 AppWorker
3351000 getroot   20   0 2074776 219132  16860 S   1.0   0.2   0:05.00 SPRTMP
3351002 getroot   20   0 2074776 219132  16860 S   1.0   0.2   0:04.91 OutboundWorker
3351007 getroot   20   0 2074776 219132  16860 S   1.0   0.2   0:02.53 AppWorker
3489033 getroot   20   0 2074776 219132  16860 S   1.0   0.2   0:00.26 Decaac
3489035 getroot   20   0 2074776 219132  16860 S   1.0   0.2   0:01.13 Encopus
3350966 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.02 OvenMediaEngine
3350975 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.01 SPDefUdp
3350976 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 DelayQueue
3350977 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.01 SRT:GC
3350978 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 DelayQueue
3350979 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.21 SPRtcSignalling
3350980 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.02 DelayQueue
3350982 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SPICE
3350983 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 MessageThread
3350984 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SPSegPub
3350985 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SegWorker
3350986 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SegPubReq
3350987 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SegWorker
3350988 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SegPubReq
3350989 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SegWorker
3350990 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SegPubReq
3350991 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.01 SPOvtPub
3350992 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.11 FilePubWorker
3350993 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.11 RtmpPubWorker
3350994 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 DelayQueue
3350995 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SPMPEGTS
3350996 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.01 PProviderTimer
3350997 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.95 SPSRT
3350998 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 SRT:SndQ:w1
3350999 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:01.35 SRT:RcvQ:w1
3351001 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:01.08 InboundWorker
3351003 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:02.22 AppWorker
3351005 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:06.91 AppWorker
3351006 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:01.98 AppWorker
3351008 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.01 StreamCollector
3351009 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.00 StreamCollector
3489037 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.40 Resampler
3489039 getroot   20   0 2074776 219132  16860 S   0.0   0.2   0:00.20 StreamWorker
```

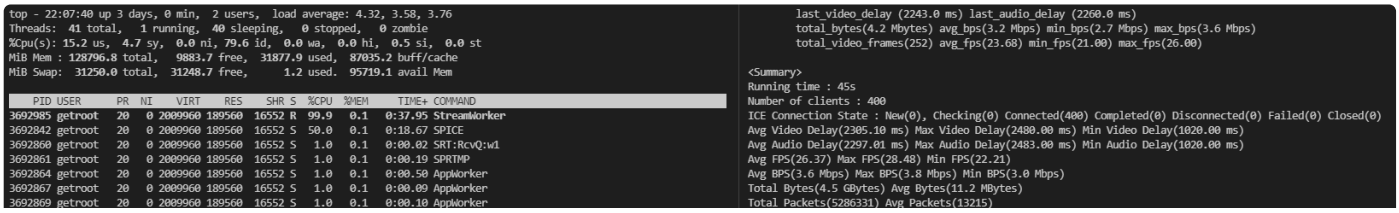
You can use OvenRtcTester to test the capacity of the server as shown below. When testing the maximum performance, OvenRtcTester also uses a lot of system resources, so test it separately from the system where OvenMediaEngine is running. Also, it is recommended to test OvenRtcTester with multiple servers. For example, simulate 500 players with -n 500 on one OvenRtcTester, and simulate 2000 players with four servers.

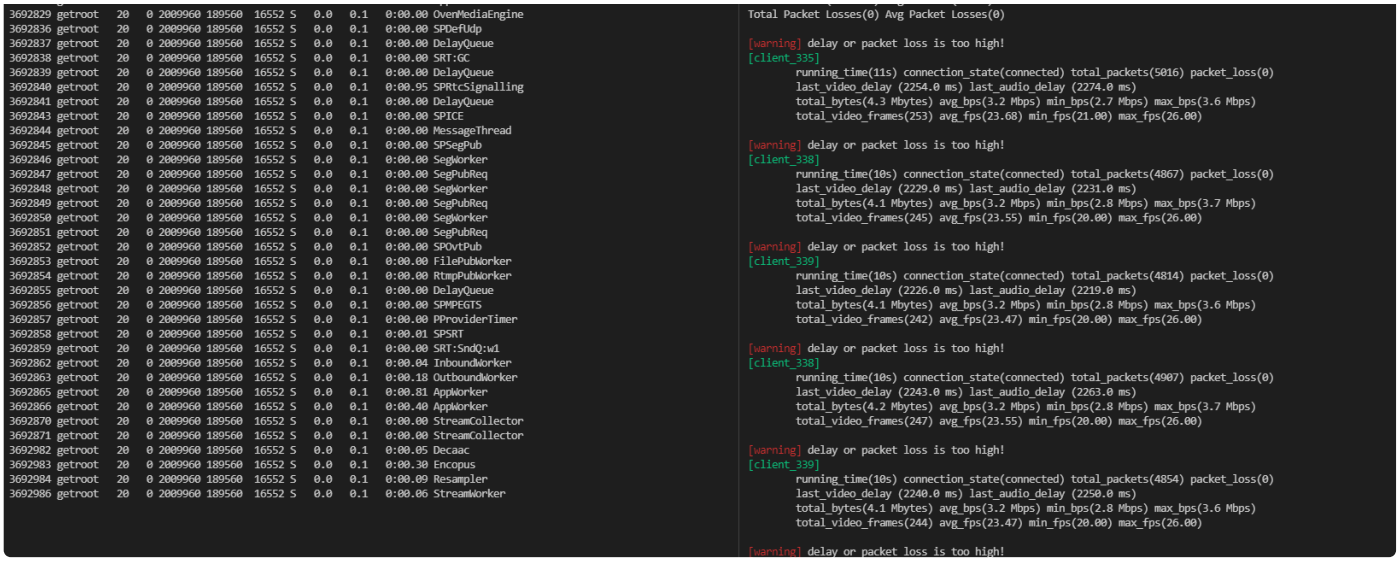
> ⚠ Building and running OvenMediaEngine in debug mode results in very poor performance. Be sure to test the maximum performance using the binary generated by make release && make

> install .

```
$ go run OvenRtcTester.go -url ws://192.168.0.160:13333/app/stream -n 100
client_0 connection state has changed checking
client_0 has started
client_0 connection state has changed connected
client_0 track has started, of type 100: video/H264
client_0 track has started, of type 101: audio/OPUS
client_1 connection state has changed checking
client_1 has started
client_1 connection state has changed connected
client_1 track has started, of type 100: video/H264
client_1 track has started, of type 101: audio/OPUS
client_2 connection state has changed checking
client_2 has started
client_2 connection state has changed connected
client_2 track has started, of type 100: video/H264
client_2 track has started, of type 101: audio/OPUS
....
client_94 connection state has changed checking
client_94 has started
client_94 connection state has changed connected
client_94 track has started, of type 100: video/H264
client_94 track has started, of type 101: audio/OPUS
client_95 connection state has changed checking
client_95 has started
client_95 connection state has changed connected
client_95 track has started, of type 100: video/H264
client_95 track has started, of type 101: audio/OPUS
client_96 connection state has changed checking
client_96 has started
<Summary>
Running time : 10s
Number of clients : 97
ICE Connection State : New(0), Checking(1) Connected(96) Completed(0) Disconnected(0) Failed(
Avg Video Delay(13.51 ms) Max Video Delay(47.00 ms) Min Video Delay(0.00 ms)
Avg Audio Delay(22.42 ms) Max Audio Delay(67.00 ms) Min Audio Delay(0.00 ms)
Avg FPS(27.20) Max FPS(32.51) Min FPS(0.00)
Avg BPS(3.7 Mbps) Max BPS(4.6 Mbps) Min BPS(0bps)
Total Bytes(238.7 MBytes) Avg Bytes(2.5 MBytes)
Total Packets(285013) Avg Packets(2938)
Total Packet Losses(306) Avg Packet Losses(3)
```

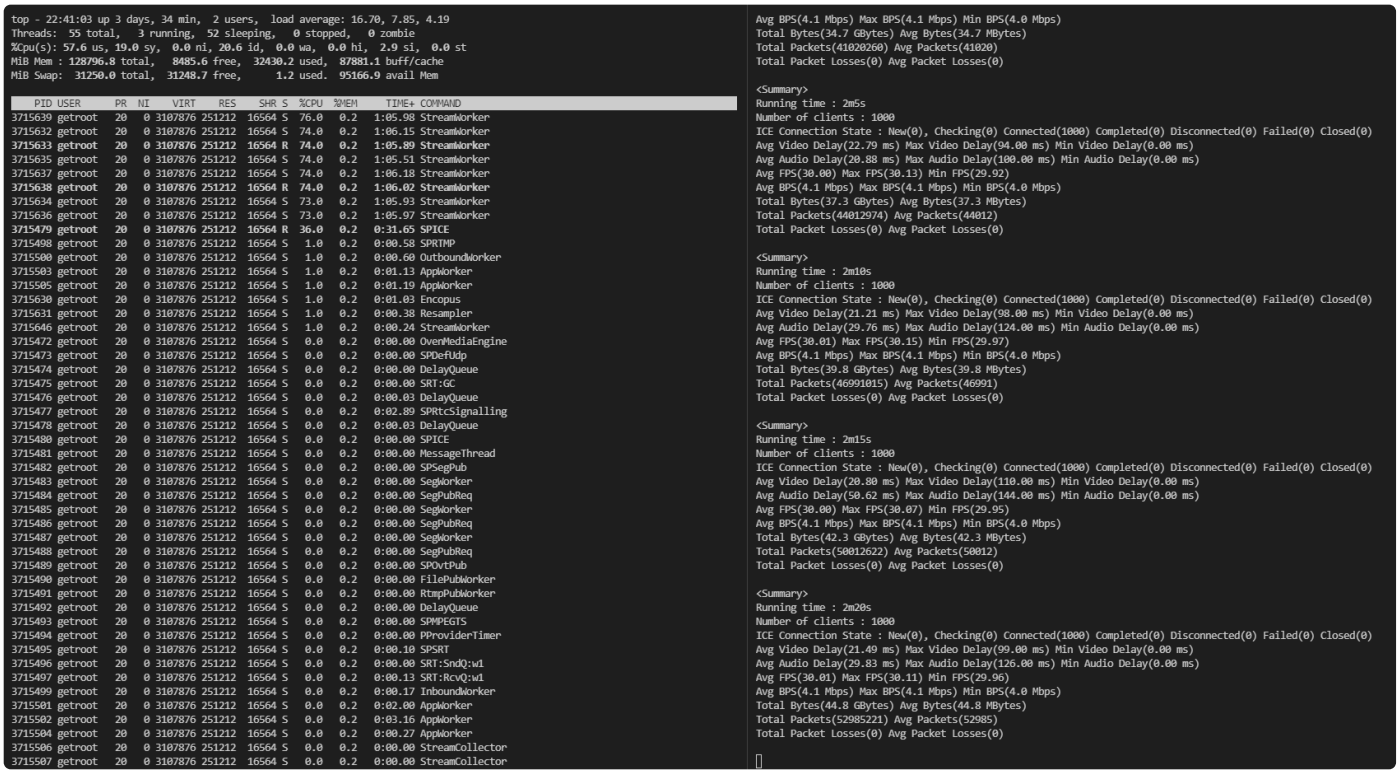
If the OvenMediaEngine's capacity is exceeded, you will notice it in OvenRtcTester's Summary report with `Avg Video Delay` and `Avg Audio Delay` or `Packet loss`.





On the right side of the above capture screen, we simulate 400 players with OvenRtcTester. <Summary> of OvenRtcTester shows that `Avg Video Delay` and `Avg Audio Delay` are very high, and `Avg FPS` is low.

And on the left, you can check the CPU usage by thread with the `top -H -p` command. This confirms that the StreamWorker threads are being used at 100%, and now you can scale the server by increasing the number of StreamWorker threads. If OvenMediaEngine is not using 100% of all cores of the server, you can improve performance by tuning the number of threads.



This is the result of tuning the number of StreamWorkerCount to 8 in config. This time, we simulated 1000 players with OvenRtcTester, and you can see that it works stably.

Tuning the number of threads

The WorkerCount in `<Bind>` can set the thread responsible for sending and receiving over the socket. Publisher's AppWorkerCount allows you to set the number of threads used for per-stream processing such as RTP packaging, and StreamWorkerCount allows you to set the number of threads for per-session processing such as SRTP encryption.

```
<Bind>
                <Providers>
                        <RTMP>
                                <Port>1935</Port>
                                <WorkerCount>1</WorkerCount>
                        </RTMP>
                        ...
                </Providers>
                ...
                <Publishers>
                        <WebRTC>
                                <Signalling>
                                        <Port>3333</Port>
                                        <WorkerCount>1</WorkerCount>
                                </Signalling>
                                <IceCandidates>
                                        <TcpRelay>*:3478</TcpRelay>
                                        <IceCandidate>*:10000/udp</IceCandidate>
                                        <TcpRelayWorkerCount>1</TcpRelayWorkerCount>
                                </IceCandidates>
                ...
</Bind>

<Application>
<Publishers>
  <AppWorkerCount>1</AppWorkerCount>
  <StreamWorkerCount>8</StreamWorkerCount>
</Publishers>
</Application>
```

Scalable Threads and Configuration

Thread name	Element in the configuration
AW-XXX	<Application><Publishers><AppWorkerCount>
StreamWorker	<Application><Publishers><StreamWorkerCount>
SPICE-XXX	<Bind><Provider><WebRTC><IceCandidates> <TcpRelayWorkerCount> <Bind><Pubishers><WebRTC><IceCandidates> <TcpRelayWorkerCount>
SPRtcSignalling	<Bind><Provider><WebRTC><Signalling> <WorkerCount>

	<Bind><Pubishers><WebRTC><Signalling><WorkerCount>
SPSegPub	<Bind><Pubishers><HLS><WorkerCount> <Bind><Pubishers><DASH><WorkerCount>
SPRTMP-XXX	<Bind><Providers><RTMP><WorkerCount>
SPMPEGTS	<Bind><Providers><MPEGTS><WorkerCount>
SPOvtPub	<Bind><Pubishers><OVT><WorkerCount>
SPSRT	<Bind><Providers><SRT><WorkerCount>

AppWorkerCount

Type	Value
Default	1
Minimum	1
Maximum	72

With `AppWorkerCount`, you can set the number of threads for distributed processing of streams when hundreds of streams are created in one application. When an application is requested to create a stream, the stream is evenly attached to one of created threads. The main role of Stream is to packetize raw media packets into the media format of the protocol to be transmitted. When there are thousands of streams, it is difficult to process them in one thread. Also, if StreamWorkerCount is set to 0, AppWorkerCount is responsible for sending media packets to the session.

It is recommended that this value does not exceed the number of CPU cores.

StreamWorkerCount

Type	Value
Default	8
Minimum	0
Maximum	72

It may be impossible to send data to thousands of viewers in one thread. StreamWorkerCount allows sessions to be distributed across multiple threads and transmitted simultaneously. This means that resources required for SRTP encryption of WebRTC or TLS encryption of HLS/DASH can be distributed and processed by multiple threads. It is recommended that this value not exceed the number of CPU cores.

Use-Case

If a large number of streams are created and very few viewers connect to each stream, increase AppWorkerCount and lower StreamWorkerCount as follows.

```
<Publishers>
  <AppWorkerCount>32</AppWorkerCount>
  <StreamWorkerCount>0</StreamWorkerCount>
</Publishers>
```

If a small number of streams are created and a very large number of viewers are connected to each stream, lower AppWorkerCount and increase StreamWorkerCount as follows.

```
<Publishers>
  <AppWorkerCount>1</AppWorkerCount>
  <StreamWorkerCount>32</StreamWorkerCount>
</Publishers>
```

Logs and Statistics

Logs

To monitor the OvenMediaEngine, you can view in real-time the log files generated by itself. You can configure a log type and level by creating the `Logger.xml` configuration file in the same location as Server.xml.

You can set up Logger.xml as shown in the following example: OvenMediaEngine prints logs separated by many tag names and levels. Set `<Tag name=".*" level="debug">` to have OvenMediaEngine print all logs and read the logs. And then it's better to disable tags that you don't need.

```
<Logger version="2">
        <!-- Log file location -->
        <Path>/var/log/ovenmediaengine</Path>

        <!-- Disable some SRT internal logs -->
        <Tag name="SRT" level="critical" />
        <Tag name="Monitor" level="critical" />

        <!-- Log level: [debug, info, warn, error, critical] -->
        <Tag name=".*" level="info" />
</Logger>
```

OvenMediaEngine generates log files. If you start OvenMediaEngine by `systemctl start ovenmediaengine`, the log file is generated to the following path.

```
/var/log/ovenmediaengine
```

If you run it directly from the command line, it will be generated to the following location:

```
/<OvenMediaEngine Binary Path>/log/
```

If you run it in the Docker container, the log file is in the following path:

```
# For Origin mode
/opt/ovenmediaengine/bin/log/
# For Edge mode
/opt/ovenmediaengine/bin/log/
```

Following is the example of real logs.

```
getroot@Jeheon-Main:/var/log/ovenmediaengine$ cat ovenmediaengine.log
[03-27 19:59:13.221] I 10996 Config | config_manager.cpp:144  | Trying to set logfile in direc
[03-27 19:59:13.221] I 10996 Config | config_manager.cpp:47   | Trying to load configurations
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:211  | OvenMediaEngine v0.9.5 (v0.9.1-
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:213  | With modules:
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:214  |   FFmpeg 3.4.2
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:215  |     Configuration: --prefix=/op
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:216  |     libavformat: 57.83.100
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:217  |     libavcodec: 57.107.100
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:218  |     libavutil: 55.78.100
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:219  |     libavfilter: 6.107.100
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:220  |     libswresample: 2.9.100
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:221  |     libswscale: 4.8.100
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:222  |   SRT: 1.3.3
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:223  |   SRTP: libsrtp2 2.2.0
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:224  |   OpenSSL: OpenSSL 1.1.0g  2 No
[03-27 19:59:13.235] I 10996 OvenMediaEngine | main.cpp:225  |     Configuration: compiler: gc
[03-27 19:59:13.240] I 10996 Monitor | monitoring.cpp:35   | Create HostMetrics(default) for r
[03-27 19:59:13.240] I 10996 OvenMediaEngine | main.cpp:148  | Trying to create a module Media
[03-27 19:59:13.240] I 10996 MediaRouter | media_router.cpp:40   | MediaRouter has been starte
[03-27 19:59:13.240] I 10996 OvenMediaEngine | main.cpp:151  | Trying to create a module RTMP
[03-27 19:59:13.244] I 10996 RtmpProvider | rtmp_provider.cpp:63   | RTMP Server has started
[03-27 19:59:13.246] I 10996 Provider | provider.cpp:40   | RtmpProvider has been started.
[03-27 19:59:13.246] I 10996 OvenMediaEngine | main.cpp:152  | Trying to create a module OVT P
[03-27 19:59:13.248] I 10996 Provider | provider.cpp:40   | OvtProvider has been started.
[03-27 19:59:13.248] I 10996 OvenMediaEngine | main.cpp:153  | Trying to create a module RTSPC
[03-27 19:59:13.250] I 10996 Provider | provider.cpp:40   | RtspcProvider has been started.
[03-27 19:59:13.250] I 10996 OvenMediaEngine | main.cpp:154  | Trying to create a module RTSP
[03-27 19:59:13.250] I 10996 RtspProvider | rtsp_provider.cpp:40   | RTSP is disabled in the c
[03-27 19:59:13.251] I 10996 OvenMediaEngine | main.cpp:157  | Trying to create a module Trans
[03-27 19:59:13.251] I 10996 Transcoder | transcoder.cpp:38   | Transcoder has been started.
[03-27 19:59:13.251] I 10996 OvenMediaEngine | main.cpp:160  | Trying to create a module WebRT
[03-27 19:59:13.251] I 10996 Signalling | rtc_signalling_server.cpp:74   | P2P is disabled in
[03-27 19:59:13.258] I 10996 Ice | ice_port.cpp:89   | ICE port is bound to 0.0.0.0:10000/UDP
[03-27 19:59:13.260] I 10996 Ice | ice_port.cpp:89   | ICE port is bound to 0.0.0.0:10001/UDP
[03-27 19:59:13.261] I 10996 Ice | ice_port.cpp:89   | ICE port is bound to 0.0.0.0:10002/UDP
```

```
[03-27 19:59:13.263] I 10996 Ice | ice_port.cpp:89   | ICE port is bound to 0.0.0.0:10003/UDP
[03-27 19:59:13.264] I 10996 Ice | ice_port.cpp:89   | ICE port is bound to 0.0.0.0:10004/UDP
[03-27 19:59:13.266] I 10996 Ice | ice_port.cpp:89   | ICE port is bound to 0.0.0.0:10005/UDP
[03-27 19:59:13.266] I 10996 Publisher | publisher.cpp:15   | WebRTC Publisher has been starte
[03-27 19:59:13.266] I 10996 WebRTC | webrtc_publisher.cpp:89   | WebRTC Publisher has started
[03-27 19:59:13.266] I 10996 Publisher | publisher.cpp:15   | WebRTC Publisher has been starte
[03-27 19:59:13.266] I 10996 OvenMediaEngine | main.cpp:161  | Trying to create a module HLS P
[03-27 19:59:13.273] I 10996 Publisher | segment_publisher.cpp:65   | HLS Publisher has starte
[03-27 19:59:13.273] I 10996 Publisher | publisher.cpp:15   | HLS Publisher has been started.
[03-27 19:59:13.275] I 10996 OvenMediaEngine | main.cpp:162  | Trying to create a module MPEG-
[03-27 19:59:13.281] I 10996 Publisher | segment_publisher.cpp:65   | DASH Publisher has start
[03-27 19:59:13.281] I 10996 Publisher | publisher.cpp:15   | DASH Publisher has been started
[03-27 19:59:13.282] I 10996 OvenMediaEngine | main.cpp:163  | Trying to create a module Low-L
[03-27 19:59:13.289] I 10996 Publisher | segment_publisher.cpp:65   | LLDASH Publisher has sta
[03-27 19:59:13.289] I 10996 Publisher | publisher.cpp:15   | LLDASH Publisher has been starte
[03-27 19:59:13.291] I 10996 OvenMediaEngine | main.cpp:164  | Trying to create a module OVT P
[03-27 19:59:13.294] I 10996 OVT | ovt_publisher.cpp:49   | Ovt Publisher has started listeni
[03-27 19:59:13.294] I 10996 Publisher | publisher.cpp:15   | OVTPublisher has been started.
[03-27 19:59:13.294] I 10996 OvenMediaEngine | main.cpp:169  | All modules are initialized suc
[03-27 19:59:13.294] I 10996 Orchestrator | orchestrator.cpp:856  | Trying to create an applic
[03-27 19:59:13.294] I 10996 Monitor | host_metrics.cpp:52   | Create ApplicationMetrics(#defa
[03-27 19:59:13.297] I 10996 Provider | application.cpp:30   | [#default#app] RTMP Provider ap
[03-27 19:59:13.297] I 10996 Provider | application.cpp:30   | [#default#app] OVT Provider app
[03-27 19:59:13.297] I 10996 Provider | application.cpp:30   | [#default#app] RTSP Pull Provid
[03-27 19:59:13.297] I 10996 Provider | application.cpp:30   | [#default#app] RTSP Provider ap
[03-27 19:59:13.298] I 10996 TranscodeApplication | transcode_application.cpp:36   | [#default
[03-27 19:59:13.300] I 10996 Publisher | application.cpp:26   | [#default#app] WebRTC Publishe
[03-27 19:59:13.302] I 10996 Publisher | application.cpp:26   | [#default#app] HLS Publisher a
[03-27 19:59:13.304] I 10996 Publisher | application.cpp:26   | [#default#app] DASH Publisher
[03-27 19:59:13.305] I 10996 Publisher | application.cpp:26   | [#default#app] LLDASH Publishe
[03-27 19:59:13.307] I 10996 Publisher | application.cpp:26   | [#default#app] OVT Publisher a
[03-27 19:59:14.706] I 11002 RtmpProvider | rtmp_server.cpp:126  | A RTMP client has connected
[03-27 19:59:14.835] I 11002 RtmpProvider | rtmp_server.cpp:226  | [#default#app/stream] RTMP
[03-27 19:59:14.835] I 11002 MediaRouter.App | media_route_application.cpp:184  | Trying to cr
[03-27 19:59:14.836] I 11002 Monitor | stream.cpp:240  |
[Stream Info]
id(2921228900), name(stream), SourceType(Rtmp), Created Time (Fri Mar 27 19:59:14 2020)

        Video Track #0: Bypass(false) Bitrate(2.50Mb) codec(1, avc) resolution(1280x720) frame
        Audio Track #1: Bypass(false) Bitrate(160.00Kb) codec(5, aac) samplerate(44.1K) format
[03-27 19:59:14.836] I 11002 Monitor | application_metrics.cpp:56   | Create StreamMetrics(str
[03-27 19:59:14.836] I 11002 TranscodeStream | transcode_stream.cpp:353  | [#default#app/strea
[03-27 19:59:14.839] I 11002 FFmpeg | third_parties.cpp:115  |  [AVCodecContext] using SAR=1/1
[03-27 19:59:14.841] I 11002 FFmpeg | third_parties.cpp:115  |  [AVCodecContext] using cpu cap
[03-27 19:59:14.846] I 11002 FFmpeg | third_parties.cpp:115  |  [AVCodecContext] profile Const
[03-27 19:59:14.849] I 11002 FFmpeg | third_parties.cpp:115  |  [AVCodecContext] v1.7.0
[03-27 19:59:14.864] I 11048 MediaRouter.App | media_route_application.cpp:184  | Trying to cr
[03-27 19:59:14.864] I 11002 TranscodeStream | transcode_stream.cpp:108  | [#default#app/strea
[03-27 19:59:14.865] I 11048 Monitor | stream.cpp:240  |
[Stream Info]
id(3169746412), name(stream_medium_o), SourceType(Transcoder), Created Time (Fri Mar 27 19:59
        >> Origin Stream Info
        id(2921228900), name(stream), SourceType(Rtmp), Created Time (Fri Mar 27 19:59:14 2020
```

```
        Video Track #0: Bypass(false) Bitrate(700.00Kb) codec(2, vp8) resolution(640x360) fra
        Video Track #1: Bypass(false) Bitrate(700.00Kb) codec(1, avc) resolution(640x360) fra
        Audio Track #2: Bypass(false) Bitrate(48.00Kb) codec(7, opus) samplerate(48.0K) forma
        Audio Track #3: Bypass(false) Bitrate(48.00Kb) codec(5, aac) samplerate(48.0K) format
[03-27 19:59:14.865] I 11048 Monitor | application_metrics.cpp:56   | Create StreamMetrics(str
[03-27 19:59:14.865] I 11048 WebRTC | rtc_stream.cpp:181  | Unsupported codec(Audio/AAC) is be
[03-27 19:59:14.880] I 11048 Publisher | stream.cpp:192  | [stream_medium_o(3169746412)] WebR
[03-27 19:59:14.881] I 11048 Publisher | stream.cpp:192  | [stream_medium_o(3169746412)] HLS
[03-27 19:59:14.881] I 11048 Publisher | stream.cpp:192  | [stream_medium_o(3169746412)] DASH
[03-27 19:59:14.881] I 11048 Publisher | stream.cpp:192  | [stream_medium_o(3169746412)] LLDAS
[03-27 19:59:14.897] I 11048 Publisher | stream.cpp:192  | [stream_medium_o(3169746412)] OVT
[03-27 19:59:14.898] I 11048 TranscodeCodec | transcode_codec_dec_aac.cpp:49   | [#default#app
[03-27 19:59:14.985] I 11048 TranscodeCodec | transcode_codec_dec_avc.cpp:48   | [#default#app
```

Statistics

OvenMediaEngine collects the following metrics for each host, application, and stream.

- Bytes in/out by protocol
- Connections by protocol
- Maximum connections and time
- Time is taken to connect to origin

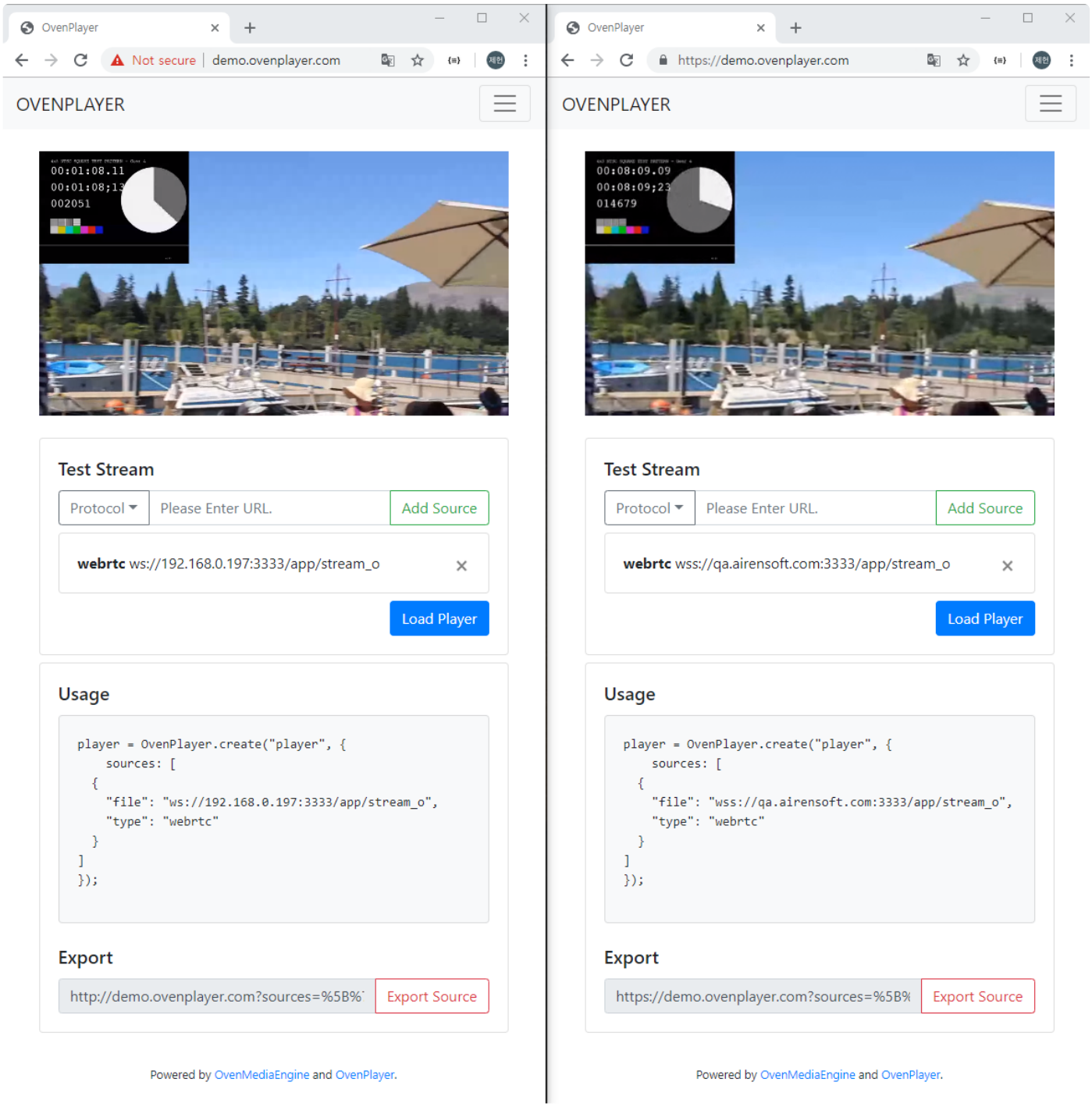
You can get the current statistics using the REST API. See Stat API for the statistics REST API.

> Files such as webrtc_stat.log and hls_rtsp_xxxx.log that were previously output are deprecated in the current version. We are developing a formal stats file, which will be open in the future.

Test Player

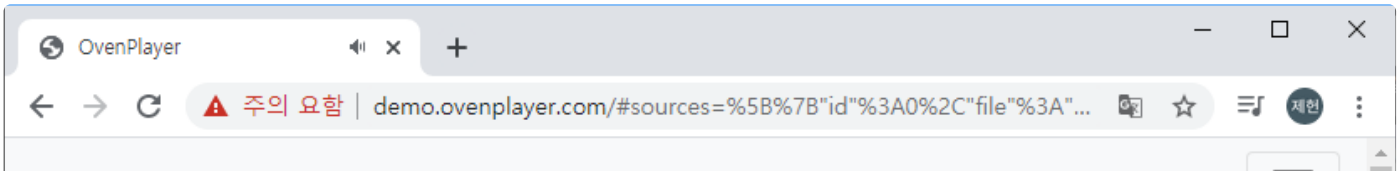
We provide you our test player to make sure that OvenMediaEngine works well. Most browsers prohibit access to the TLS-based HTTPS site through unsecured HTTP or WebSocket (WS) for security reasons. Thus, we have prepared the HTTP or HTTPS based player as follows:

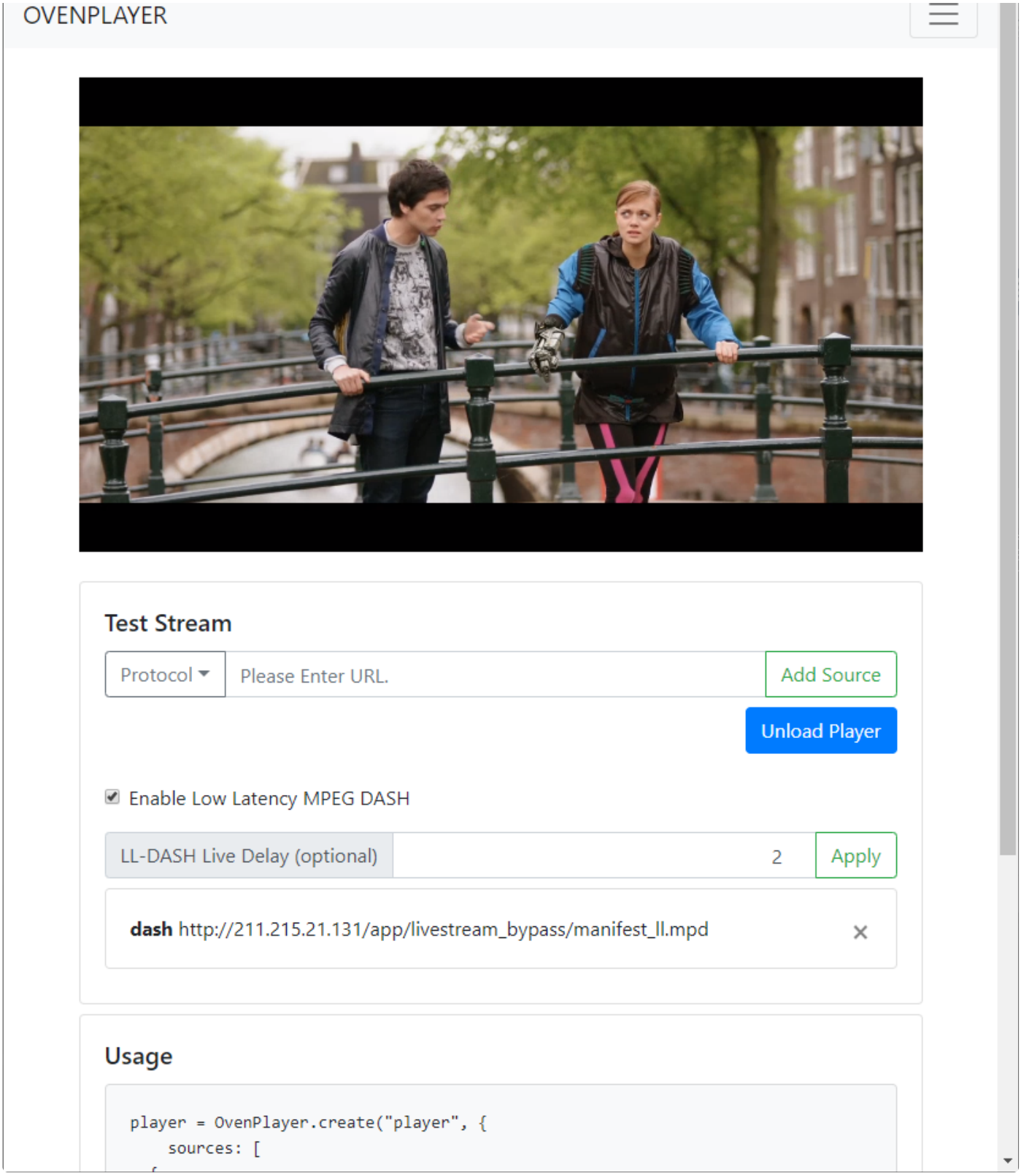
Site URL	Description
http://demo.ovenplayer.com	A player without TLS that can test all streaming URLs such as `WS://`, `WSS://`, `HTTP://`, `HTTPS://`
https://demo.ovenplayer.com	A player with TLS that can only test `WSS://` and `HTTPS://` based streaming URLs



Delay option for Low-Latency DASH

When playing Low-Latency DASH, you can control the delay time in the player as shown below. Delay time is closely related to the buffering size. The smaller the value, the shorter the latency, but if it is too small, there is no buffer and playback may not be smooth. In a typical network environment, it is most stable to give 2 as the delay value.





Troubleshooting

We will update this document as we gather troubleshooting examples. (Written in Nov 04, 2021)

`prerequisites.sh` Script Failed

If you have problems with the `prerequisites.sh` the script we have provided, please install it manually

Platform Specific Installation

Ubuntu 18

```
sudo apt install -y build-essential nasm autoconf libtool zlib1g-dev tclsh cmake cu
```

Fedora 28

```
sudo yum install -y gcc-c++ make nasm autoconf libtool zlib-devel tcl cmake
```

CentOS 7

```
# for downloading latest version of nasm (x264 needs nasm 2.13+ but centos provides
sudo curl -so /etc/yum.repos.d/nasm.repo https://www.nasm.us/nasm.repo
sudo yum install centos-release-scl
sudo yum install -y bc gcc-c++ cmake nasm autoconf libtool glibc-static tcl bzip2 z
source scl_source enable devtoolset-7
```

Common Installation

Install OpenSSL

```
PREFIX=/opt/ovenmediaengine && \
OPENSSL_VERSION=1.1.0g && \
DIR=/tmp/openssl && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://www.openssl.org/source/openssl-${OPENSSL_VERSION}.tar.gz | tar -xz --strip-
./config --prefix="${PREFIX}" --openssldir="${PREFIX}" -Wl,-rpath="${PREFIX}/lib" shared no-i
make -j 4 && \
sudo make install_sw && \
rm -rf ${DIR} && \
sudo rm -rf ${PREFIX}/bin
```

Install SRTP

```
PREFIX=/opt/ovenmediaengine && \
SRTP_VERSION=2.2.0 && \
```

```
DIR=/tmp/srtp && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/cisco/libsrtp/archive/v${SRTP_VERSION}.tar.gz | tar -xz --strip-
./configure --prefix="${PREFIX}" --enable-shared --disable-static --enable-openssl --with-oper
make shared_library -j 4 && \
sudo make install && \
rm -rf ${DIR}
```

Install SRT

```
PREFIX=/opt/ovenmediaengine && \
SRT_VERSION=1.3.3 && \
DIR=/tmp/srt && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/Haivision/srt/archive/v${SRT_VERSION}.tar.gz | tar -xz --strip-co
PKG_CONFIG_PATH=${PREFIX}/lib/pkgconfig:${PKG_CONFIG_PATH} ./configure --prefix="${PREFIX}" --
make -j 4 && \
sudo make install && \
rm -rf ${DIR} && \
sudo rm -rf ${PREFIX}/bin
```

Install Opus

```
PREFIX=/opt/ovenmediaengine && \
OPUS_VERSION=1.1.3 && \
DIR=/tmp/opus && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://archive.mozilla.org/pub/opus/opus-${OPUS_VERSION}.tar.gz | tar -xz --strip-
autoreconf -fiv && \
./configure --prefix="${PREFIX}" --enable-shared --disable-static && \
make -j 4&& \
sudo make install && \
sudo rm -rf ${PREFIX}/share && \
rm -rf ${DIR}
```

Install x264

```
PREFIX=/opt/ovenmediaengine && \
X264_VERSION=20190513-2245-stable && \
DIR=/tmp/x264 && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://download.videolan.org/pub/videolan/x264/snapshots/x264-snapshot-${X264_VERSI
./configure --prefix="${PREFIX}" --enable-shared --enable-pic --disable-cli && \
make -j 4&& \
sudo make install && \
rm -rf ${DIR}
```

Install VPX

```
PREFIX=/opt/ovenmediaengine && \
VPX_VERSION=1.7.0 && \
DIR=/tmp/vpx && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://codeload.github.com/webmproject/libvpx/tar.gz/v${VPX_VERSION} | tar -xz --s
./configure --prefix="${PREFIX}" --enable-vp8 --enable-pic --enable-shared --disable-static --
make -j 4 && \
sudo make install && \
rm -rf ${DIR}
```

Install FDK-AAC

```
PREFIX=/opt/ovenmediaengine && \
FDKAAC_VERSION=0.1.5 && \
DIR=/tmp/aac && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/mstorsjo/fdk-aac/archive/v${FDKAAC_VERSION}.tar.gz | tar -xz --s
autoreconf -fiv && \
./configure --prefix="${PREFIX}" --enable-shared --disable-static --datadir=/tmp/aac && \
make -j 4&& \
sudo make install && \
rm -rf ${DIR}
```

Install FFMPEG

```
PREFIX=/opt/ovenmediaengine && \
FFMPEG_VERSION=3.4 && \
DIR=/tmp/ffmpeg && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/AirenSoft/FFmpeg/archive/ome/${FFMPEG_VERSION}.tar.gz | tar -xz
PKG_CONFIG_PATH=${PREFIX}/lib/pkgconfig:${PKG_CONFIG_PATH} ./configure \
--prefix="${PREFIX}" \
--enable-gpl \
--enable-nonfree \
--extra-cflags="-I${PREFIX}/include"  \
--extra-ldflags="-L${PREFIX}/lib -Wl,-rpath,${PREFIX}/lib" \
--extra-libs=-ldl \
--enable-shared \
--disable-static \
--disable-debug \
--disable-doc \
--disable-programs \
--disable-avdevice --disable-dct --disable-dwt --disable-error-resilience --disable-lsp --disa
--disable-everything \
--enable-zlib --enable-libopus --enable-libvpx --enable-libfdk_aac --enable-libx264 \
--enable-encoder=libvpx_vp8,libvpx_vp9,libopus,libfdk_aac,libx264 \
```

```
--enable-decoder=aac,aac_latm,aac_fixed,h264 \
--enable-parser=aac,aac_latm,aac_fixed,h264 \
--enable-network --enable-protocol=tcp --enable-protocol=udp --enable-protocol=rtp --enable-d
--enable-filter=asetnsamples,aresample,aformat,channelmap,channelsplit,scale,transpose,fps,se
make && \
sudo make install && \
sudo rm -rf ${PREFIX}/share && \
rm -rf ${DIR}
```

Install JEMALLOC

```
PREFIX=/opt/ovenmediaengine && \
JEMALLOC_VERSION=5.2.1 && \
DIR=${TEMP_PATH}/jemalloc && \
mkdir -p ${DIR} && \
cd ${DIR} && \
curl -sLf https://github.com/jemalloc/jemalloc/releases/download/${JEMALLOC_VERSION}/jemalloc-
./configure --prefix="${PREFIX}" && \
make && \
sudo make install_include install_lib && \
rm -rf ${DIR}
```

`systemctl start ovenmediaengine` failed

Check SELinux

If SELinux is running on your system, SELinux can deny the execution of OvenMediaEngine.

```
# Example of SELinux disallow OvenMediaEngine execution
$ systemctl start ovenmediaengine
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'ovenmediaengine.service'.
Authenticating as: Jeheon Han (getroot)
Password:
==== AUTHENTICATION COMPLETE ====
Failed to start ovenmediaengine.service: Unit ovenmediaengine. service not found.
# Check if SELinux is enabled
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      31
```

```
# Check if SELinux denies execution
$ sudo tail /var/log/messages
...
May 17 12:44:24 localhost audit[1]: AVC avc:  denied  { read } for  pid=1 comm="systemd" name=
May 17 12:44:24 localhost audit[1]: AVC avc:  denied  { read } for  pid=1 comm="systemd" name=
```

You can choose between two methods of adding a policy to SELinux or setting SELinux to permissive mode. To add a policy, you must apply the SELinux policy file for the OvenMediaEngine service to your system as follows:

```
$ cd <OvenMediaEngine Git Clone Root Path>
$ sudo semodule -i misc/ovenmediaengine.pp
$ sudo touch /.autorelabel
# If you add a policy to SELinux, you must reboot the system.
$ sudo reboot
```

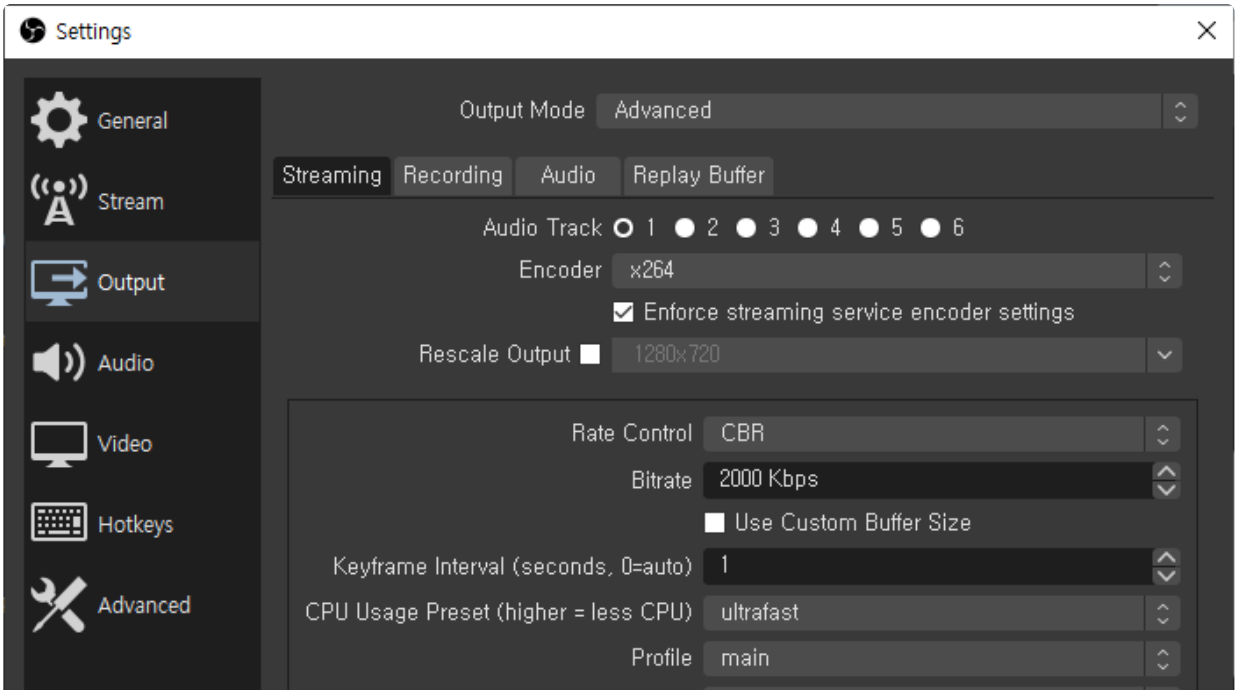
Setting SELinux to permissive mode is as simple as follows. But we don't recommend this method.

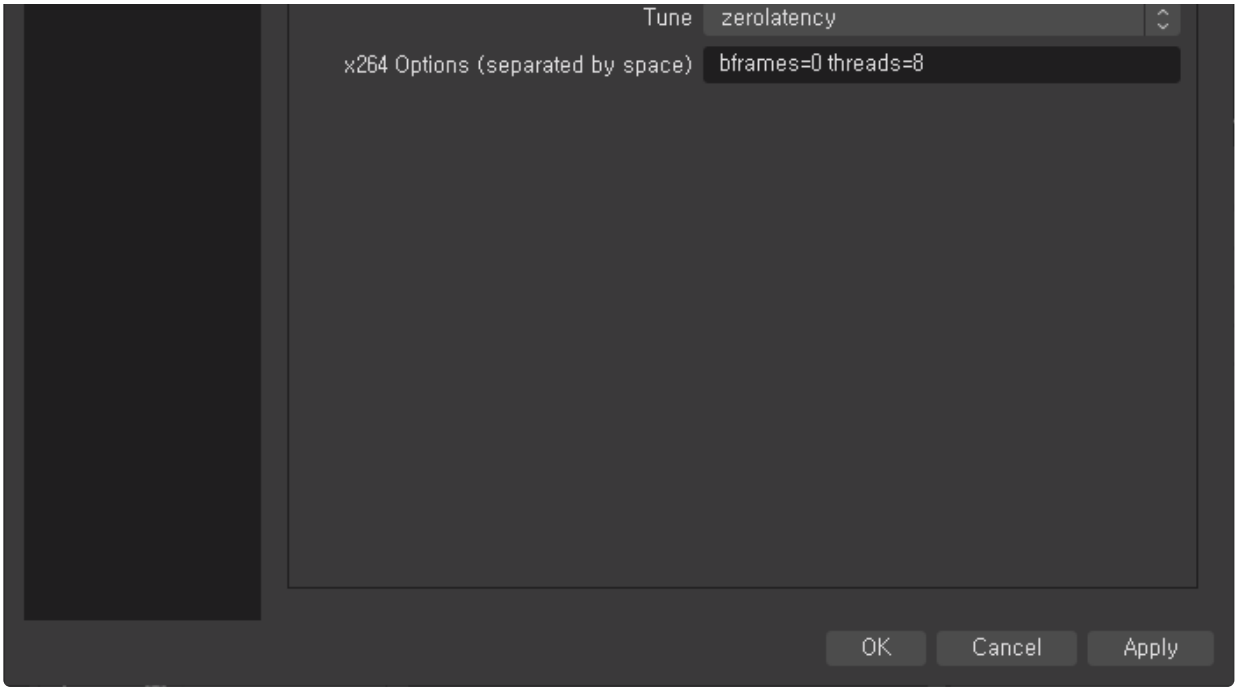
```
$ sudo setenforce 0
```

Streaming is not smooth

1. If you are using Transcoding as Bypass in OvenMediaEngine, and streaming does not work in all players

WebRTC does not support b-frame of H.264. But if your encoder sends b-frames the video will be stuttered in the player. In this case, you can solve the problem by disabling the b-frame function in your encoder. For OBS, you can set bframes=0 option as below.





Or by **activating the encoding options** in OvenMediaEngine.

> ⓘ Setting up Transcoding options in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/transcoding#encodes

2. When streaming does not work in some players

In this case, you are probably trying to stream with UDP in an environment where packet loss is high due to network performance, connection problems, etc., the interruption during stream playback may more and more worsen. This problem can be solved simply by playing with WebRTC/TCP.

If you want to monitor packet loss in your Chrome browser, you can access it by typing '**chrome://webrtc-internals**' in the address bar.

> ⓘ Setting up WebRTC over TCP in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/streaming/webrtc-publishing#webrtc-over-tcp

Also, if the device's network speed, which is running the player, isn't fast enough to accommodate the stream's BPS, the stuttering during streaming won't resolve and will eventually drop the connection. In this case, there is no other way than to speed up your network.

3. When streaming fails due to excessive CPU/Memory/Network usage of Origin in OvenMediaEngine

If the Origin server uses excessive CPU/Memory/Network, all players may experience stuttering during streaming.

When you see Origin is CPU intensive on your Origin-Edge structure, the transcoding options in the OvenMediaEngine may be the primary cause. That is, you may have set the quality of the input stream too

high, or the output stream to exceed the capabilities of your hardware significantly. In this case, it can be

> ⓘ Setting up GPU Acceleration in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/transcoding/gpu-usage

4. When streaming fails due to excessive CPU/Memory/Network usage of Edge in OvenMediaEngine

If the edge server excessively uses CPU/Memory/Network, the player connected to that Edge may experience stuttering during streaming. In this case, it can be solved by **expanding Edge**.

5. If you have enough CPU/Memory/Network, but streaming is not smooth

5-1. When a specific thread is using the CPU excessively

When you see a specific thread overuses the CPU, the video may not stream smoothly. Please refer to the manual below for more information on this.

> ⓘ Tuning OvenMediaEngine Performance:
> https://airensoft.gitbook.io/ovenmediaengine/performance-tuning#performance-tuning

5-2. Tuning your Linux kernel

The Linux kernel, which is set by default, **cannot handle 1Gbps output**, so put it as follows:

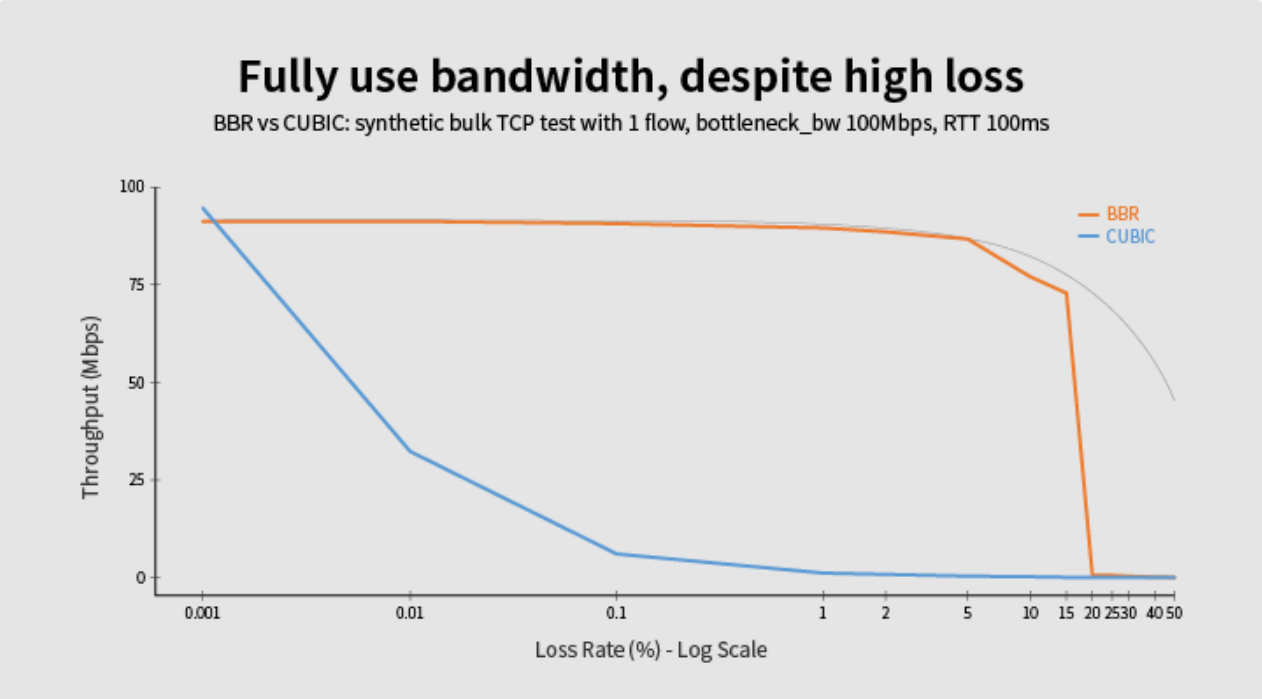
```
[ec2-user@ip-172-31-56-213 ~]$ cat /etc/sysctl.conf
fs.file-max = 100000
net.core.somaxconn = 65535
net.ipv4.tcp_max_tw_buckets = 1440000
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_fin_timeout = 15
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_max_syn_backlog = 3240000
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.rmem_default = 16777216
net.core.wmem_default = 16777216
net.core.optmem_max = 40960
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.core.netdev_max_backlog = 50000
net.ipv4.tcp_max_syn_backlog = 30000
net.ipv4.tcp_max_tw_buckets = 2000000
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_fin_timeout = 10
net.ipv4.tcp_slow_start_after_idle = 0
```

```
[ec2-user@ip-172-31-56-213 ~]$ cat /etc/security/limits.conf
* soft nofile 1048576
* hard nofile 1048576
```

5-3. Congestion Control: Change to BBR

The mobile environment used by many people uses a **wireless network**. It has a high network speed but, conversely, can cause high packet loss.

Look, **CUBIC**, the Congestion Control set by default in your Linux, adjusts the TCP Window by packet loss, so it is not suitable to provide stable streaming in such an environment.



Source: iccrg-bbr-congestion-control-02.pdf (Page 18)

So our suggestion is to use Google's **BBR**. This setting is even more important if you mainly provide WebRTC services to mobile users who use a wireless network. Change the Congestion Control from CUBIC to BBR on your Linux.

Player connection fails

1. Due to the Mixed Contents

If you try to access OvenMediaEngine's WebRTC URL starting with **ws://** *(Non-TLS)* from an **HTTPS** *(HTTP/TLS)* site, the connection may be rejected due to a mixed content problem depending on the browser.

In this case, you can solve this by installing a certificate in OvenMediaEngine and trying to connect with the **wss://** *(WebSocket/TLS)* URL.

> ⓘ Setting up TLS Encryption in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/streaming/tls-encryption

2. Due to a Cross-Origin Resource Sharing (CORS) Error

As of October 2021, most browsers have enforced the CORS policy, and CORS errors often occur when requesting access to other domains if it is not a TLS site. In this case, you can solve the problem by **installing a certificate** on the site that loads the player.

3. When the message "Too many open files" appears in the log, the player cannot connect

At some point, when the message "**Too many open files**" is output in your OvenMediaEngine log, it may not be able to handle any more player connections. In this case, you can solve the problem by setting it as follows:

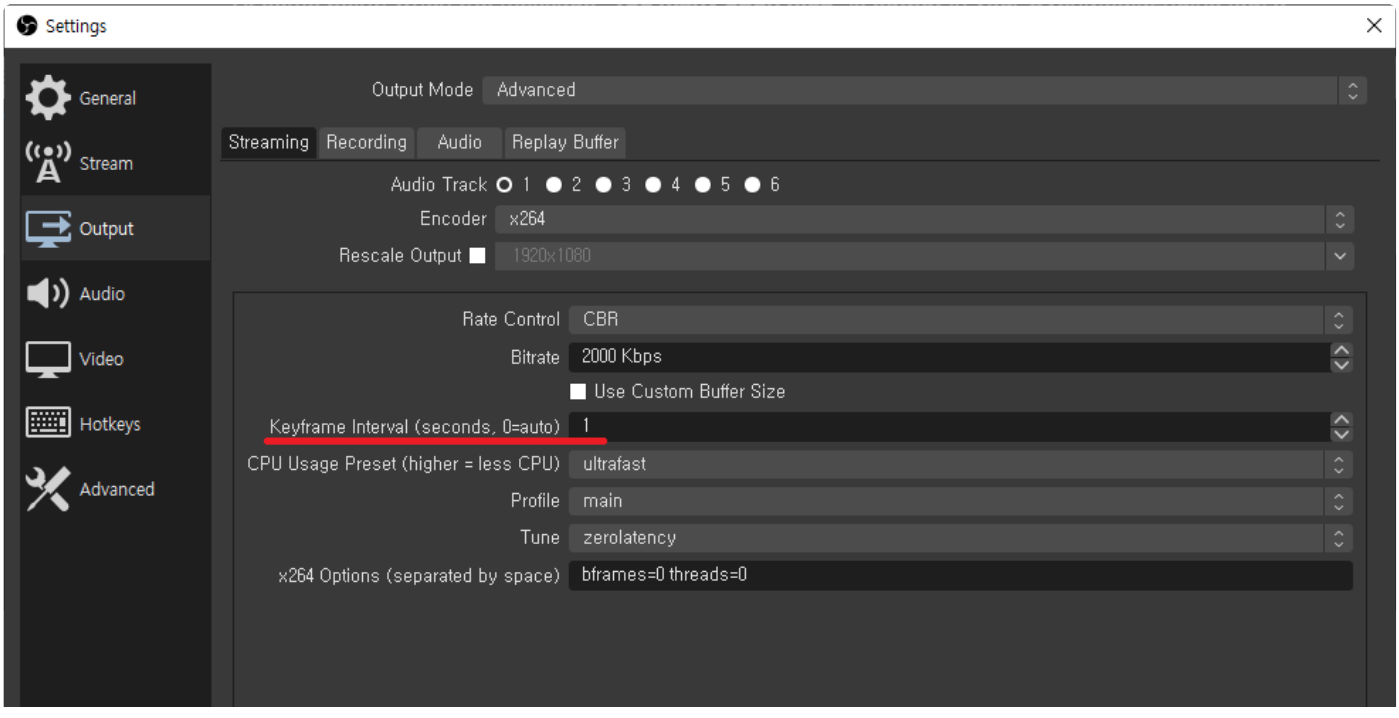
```
[ec2-user@ip-172-31-56-213 ~]$ cat /etc/security/limits.conf
* soft nofile 1048576
* hard nofile 1048576
```

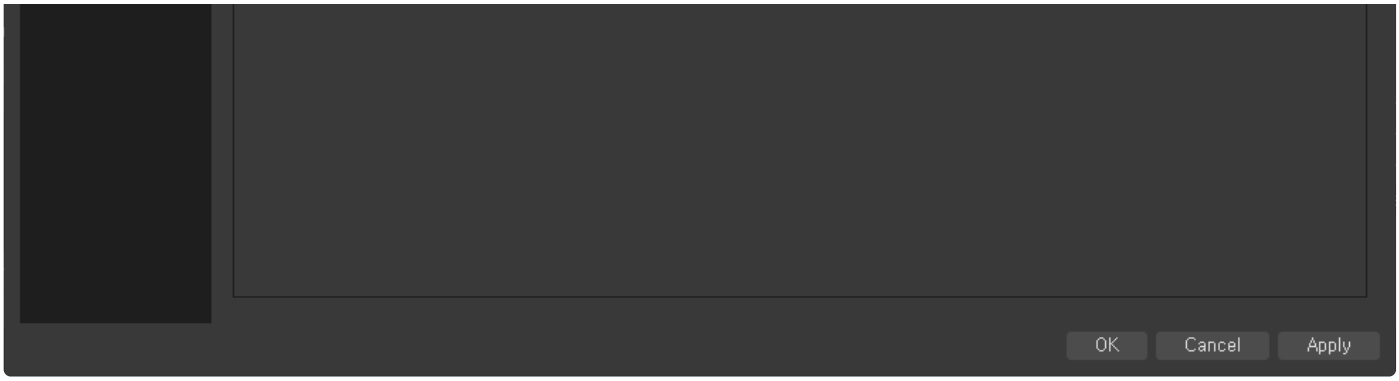
Player takes a long time first to load while trying to stream

1. Due to the Keyframe Interval

If you use Transcoding as Bypass in OvenMediaEngine and set a **long keyframe interval** in the encoder, the WebRTC player cannot start streaming until a keyframe is an input.

In this case, you can solve this by setting the keyframe interval in the encoder to **1-2 seconds**,





How to set the keyframe intverval in OBS, which is the most used encoder

Or by **enabling the encoding options** in OvenMediaEngine.

> Setting up Transcoding options in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/transcoding#encodes

---

A/V is out of sync

1. When the A/V sync does not match during initial streaming, and it gradually fits

A/V may not be input evenly from the encoder. There are some encoders with policies for reliable streaming that they decide, for example, sending audio first and video very later, or video first and audio very late.

OvenMediaEngine outputs the input received from the encoder as-is for sub-second latency streaming. The WebRTC player also streams the received input as-is, so the A/V sync may not match during the initial playback due to the policy of specific encoders.

However, this can be resolved naturally as the player will sync A/V while streaming based on Timestamp. Still, if this work looks like an error, you can also solve it by **enabling JitterBuffer** in OvenMediaEngine.

Also, suppose you are using a transcoder in OvenMediaEngine and trying to input with b-frames of H264. Audio is encoded fast, but a video is buffered at the decoder because of b-frames. Therefore, there is a time difference at the start of each encoding, which may cause the A/V to be out of sync. Even in this case, **enabling JitterBuffer** will solve this problem.

> Setting up WebRTC JitterBuffer in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/streaming/webrtc-publishing#publisher

2. Time has passed, but A/V is out of sync

There may be cases where the A/V sync is not corrected even after a certain amount of time has elapsed after playback. This problem is caused by **small internal buffers** in some browsers such as Firefox, which

causes the player to give up calibration if the A/V sync differs too much. But this can also be solved by

> Setting up WebRTC JitterBuffer in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/streaming/webrtc-publishing#publisher

Nevertheless, if the A/V sync is not corrected, you should suspect an error in the original video file, which can be checked by playing as HLS.

However, if A/V sync is well during streaming with HLS, this is OvenMediaEnigne's bug. If you find any bugs, please feel free to report them to **OvenMediaEngine GitHub Issues**.

No audio is output

1. When Opus is not set in the encoding options

WebRTC supports Opus, not AAC, as an audio codec. Because RTMP and other protocols mainly use and transmit AAC as the audio codec, you may not have set up Opus, but WebRTC cannot output audio without Opus. This can be solved by **setting Opus** in OvenMediaEnigne.

> Setting up Opus Codec in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/transcoding#audio

---

Not the video quality you want

1. When video encoding is enabled in OvenMediaEngine

If you are using video encoding in OME, the video bitrate may be set low. In this case, the video quality can be improved by **increasing the unit of video bitrate**.

However, since OvenMediaEngine has the default to the fastest encoding option for sub-second latency streaming, the video quality may not be as good as the set video bitrate. In this case, OvenMediaEngine provides an **output profile preset** that can control the quality, so you can choose to solve it.

> Choosing an Encoding Preset in OvenMediaEngine:
> https://airensoft.gitbook.io/ovenmediaengine/transcoding#video

2. If you are using Transcoding as Bypass in OvenMediaEngine

Since the encoder is transmitting video to OvenMediaEngine in low quality, you can solve it by increasing the input quality in the encoder settings.

P2P Delivery (Experiment)

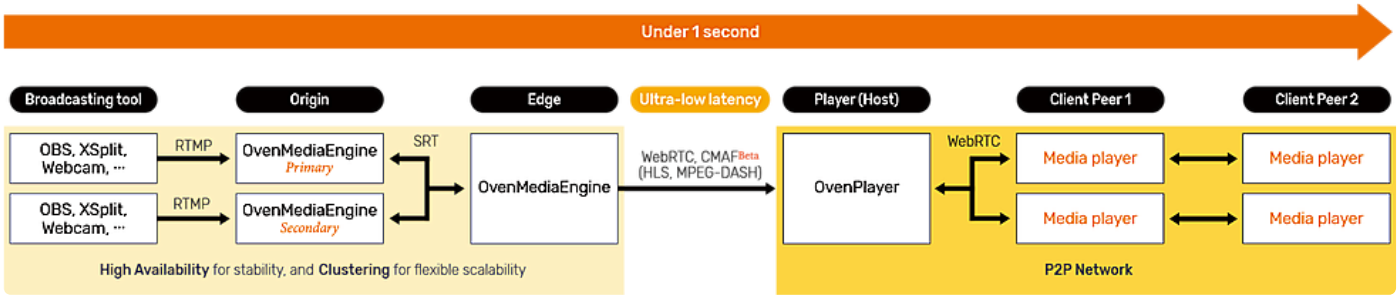
OvenMediaEngine provides P2P Delivery to be able to distribute Edge Traffic to Player. This feature is currently the Preview version, and if you want to use it, you need only to use OvenPlayer. Moreover, we plan to perform more experiments in various real-world and then upgrade it to the full version in OvenMediaEngine.

First of all, we have rules. The peer that sends the Traffic in the P2P network is called a Host Peer, and the peer that receives the Traffic from the Host Peer is called a Client Peer. Also, P2P Delivery in OvenMediaEngine doesn't designate the Client Peer as the Host Peer again. In other words, it only operates as 1 Depth.

What are the benefits of using P2P Delivery?

According to our experiments so far, P2P Delivery provides the best performance and stability when using 1 Depth to connect between Players and connecting up to two Players to one Player.

In other words, P2P Delivery has distributed two-thirds of existing Traffic. So, this means that it can expand the Capacity of the Edge Network by three times and reduce Traffic costs by two-thirds.



How does it work?

You can use the P2P function of OvenMediaEngine by adding the `<P2P>` element as the following settings:

Server.xml

```
<Server version="...">
        ...
        <P2P>
                <MaxClientPeersPerHostPeer>2</MaxClientPeersPerHostPeer>
        </P2P>
        ...
</Server>
```

Also, If you want to use P2P Delivery when your OvenMediaEngine is running in Origin-Edge Cluster-Mode, you need to apply this setting to all the Edges. You can instantly test P2P Delivery with OvenPlayer.

- `<MaxClientPeersPerHostPeer>` sets the number of Client Peers connecting to one Host Peer.

How does it classify Peers?

When OvenMediaEngine receives a WebRTC connection request from a new player, it determines the Host Peer or Client Peer according to the following rules:

Qualification for Host Peer	Qualification for Client Peer
• The device isn't Mobile • OS isn't Linux • Browser isn't MS Edge Browser • Browser isn't Unknown Browser	• One of the Host Peers uses the same kind of browser • Host Peer is vacant

> When any Host Peer is disconnected, OvenMediaEngine detects this situation and immediately reconnects the Client Peer connected to that Host Peer to the Edge to ensure stability.
>
> Also, we are preparing a smarter algorithm based on user location, platform performance, and network statistical information for classifying Host Peers or Client Peers.

If you have a better idea, we hope that you improve our code and contribute to our project. Please visit OvenMediaEngine GitHub.

https://github.com/AirenSoft/OvenMediaEngine/blob/master/src/projects/rtc_signalling/p2p/
github.com

Introduction

The Enterprise Edition of OvenMediaEnigne has been released to provide commercial codecs. The Enterprise version is distributed for a fee because it has a commercial codec installed, and you must agree to the EULA to use it.

In addition, it provides valuable features to managers in a commercial environment, such as the Web Console, and some additional functions not available in the Community Edition of OvenMediaEngine.

All features of OvenMediaEngine are identical to the Open Source version and the Enterprise version, with the following exceptions.

Enterprise adds:

- Commercial library
- The features that the customer does not allow to be disclosed as open source among the features developed by the customer.

Advanced Features

Summary

Cateroty	Feature	Description	Link
API	Reload Virtual Host	Reload Virtual Host	**REST API (Beta)**

Admin	Reload Application Web Console		
Codec	Beamr H.264 Encoder		

Advanced Features

Advanced Codec

The Enterprise Edition of OvenMediaEnigne includes commercial H.264/H.265 codecs.

Web Console

The Web Console is a web application that allows you to monitor and operate OvenMediaEngine.

Overview

The main features of Web Console are:

- Read and display the config file of OvenMediaEngine.
- Interpret the OvenMediaEngine configuration file and work with the REST API.
- Use the REST API to display a list of Virtual Hosts, Applications, Streams, and configuration information.
- You can reload Virtual Host and Application.
- You can load and unload Virtual Hosts that have been added or removed from the OvenMediaEngine configuration file.
- Provide a Test Player that can play on the generated stream.

Installation & Configuration

The Web Console is distributed as part of the OvenMediaEngine Enterprise package. The default installation path and running port for the Web Console are:

Installation path	`/usr/share/ovenmediaengine/console`
Configuration file path	`/usr/share/ovenmediaengine/console/conf/config.yaml`
Log file path	`/var/log/ovenmediaengine/console/ovenmediaengine-console.log`
Running port	`5000`
Initial account	username: `admin` password: `admin`

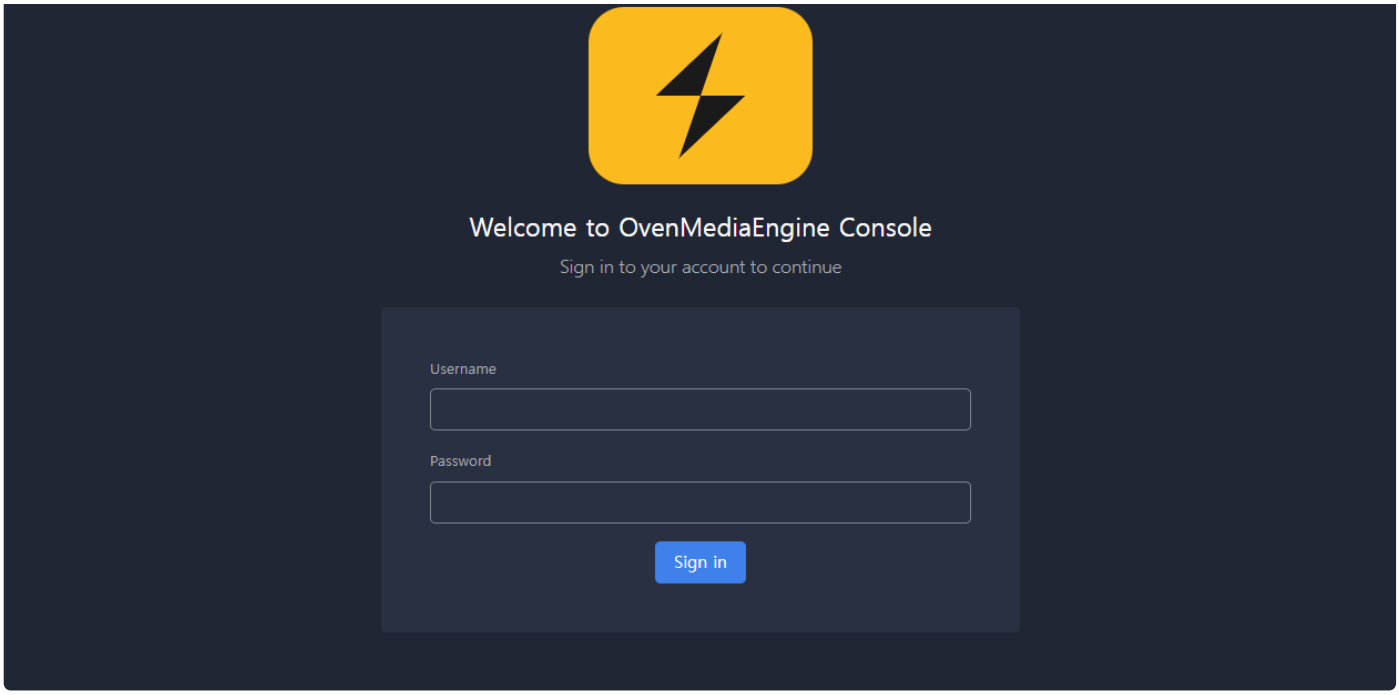
The default configuration for the Web Console is as follows:

```
# Config about the OvenMediaEngine Console.
ovenmediaengineconsole:
  # Port for serve OvenMediaEngine Console. Default is 5000.
  port: 5000
  # Log dir of OvenMediaEngine Console.
  # When using a different value, be careful that the directory exists.
  logDir: /var/log/ovenmediaengine/console
  # Log level: CRITICAL, FATAL, ERROR, WARN, WARNING, INFO, DEBUG, NOTSET
  logLevel: DEBUG


# Config about the OvenMediaEngine.
ovenmediaengine:
  # Directory where the OvenMediaEngine binary is installed.
  # Default is /usr/share/ovenmediaengine.
  installedDir: /usr/share/ovenmediaengine
```

Common User Interfaces

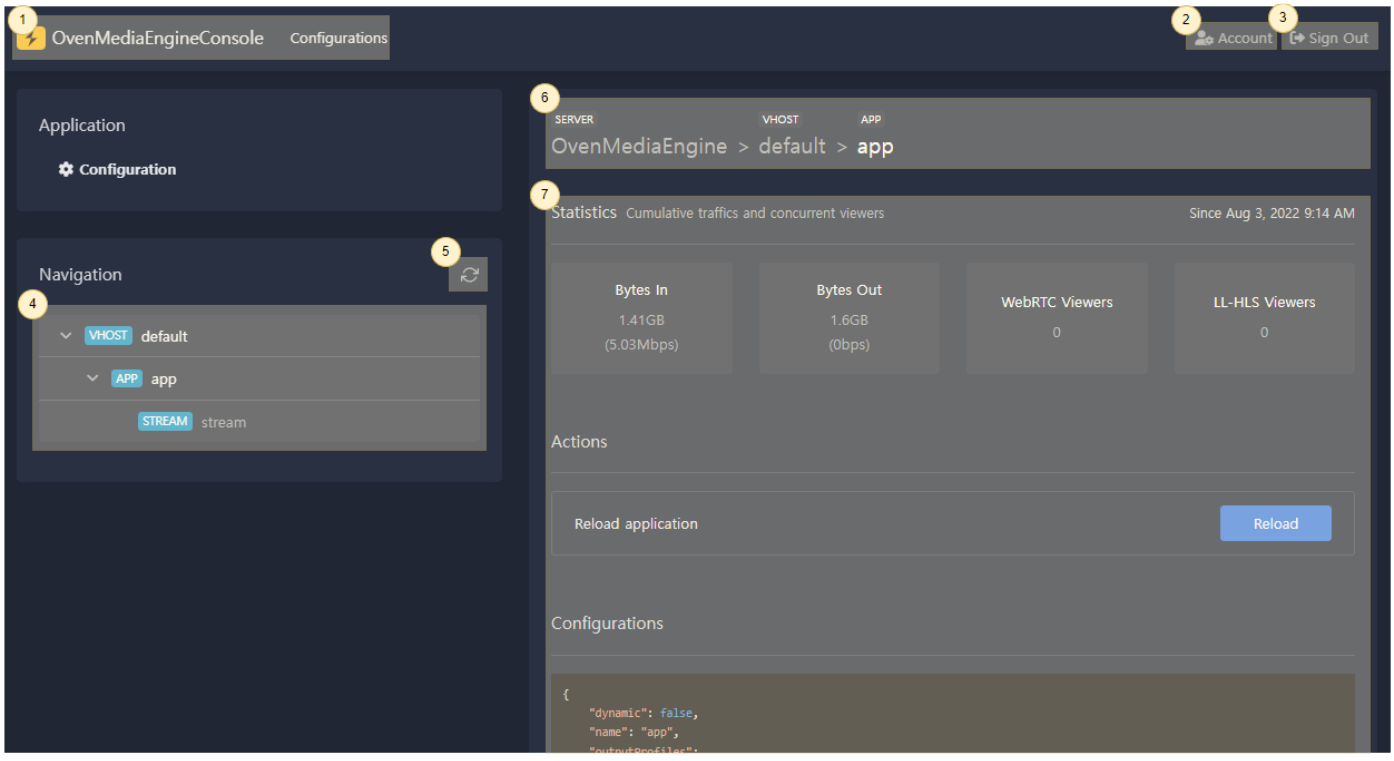
This section describes the common user interface of Web Console.

Sign in



You can sign in from this login screen. A Username and Password of the initial account are `admin`.

Common Layout



① Home Link (Server page)

This logo is a link to the first entry screen. You can check the settings of OvenMediaEngine.

② Account Information Link

This tab is a link to a page where you can edit your account information.

③ Sign Out

Sign out and then expire the session.

④ Virtual Host, Application, Stream Navigation

Tree-structured navigation with access to Virtual Hosts, Applications, and Streams running on OvenMediaEngine. The selected resource is highlighted.

> ⓘ You can load and unload virtual hosts or applications when added or removed from the OvenMediaEngine configuration.

⑤ Update Virtual Host, Application, Stream Navigation

Reload the Virtual Host, Application, and Stream information. If a new stream is added to your application, you can view the new stream.

⑥ Current Path

Shows the path of the selected Virtual Host, Application, and Stream. You can click the parent path to choose that resource.

⑦ Virtual Host, Application, Stream Details

Displays Statistics, Actions, and Setting information of the selected resource.

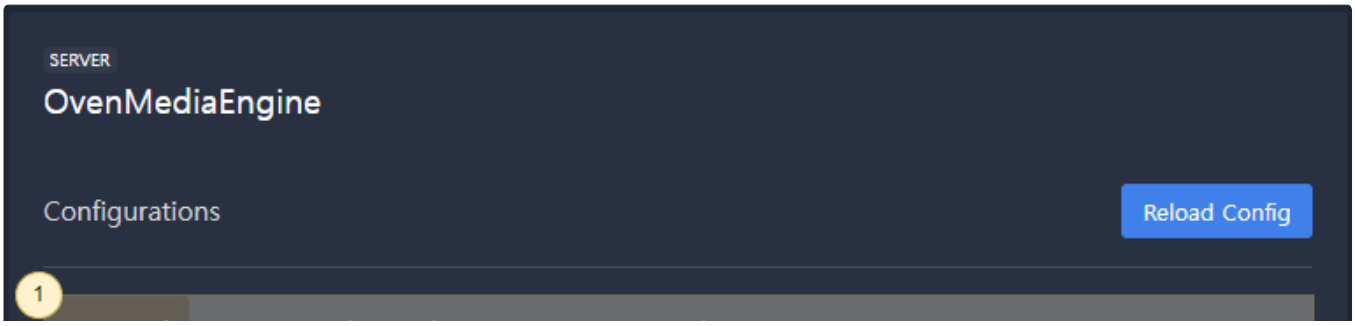
Features

Describe the main features of Web Console.

Display the Configuration

The OvenMediaEngine Console displays each Server, Virtual Host, Application, and Stream configuration.

Server Configuration



Server.xml VHost-OvenPlayer.xml VHost-OvenSpace.xml

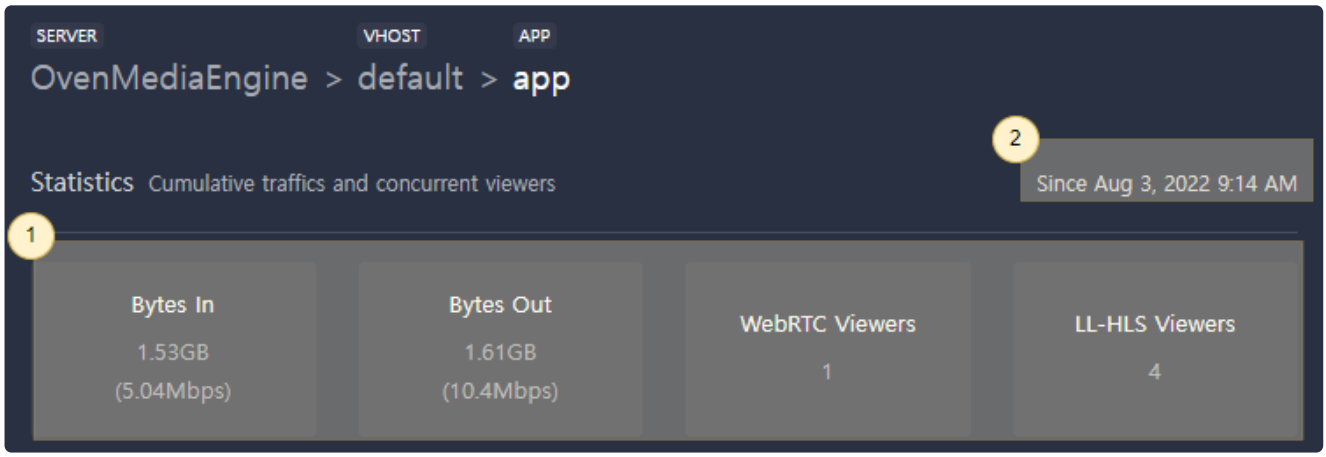
```
<?xml version="1.0" encoding="UTF-8"?>

<Server version="8">
    <Name>OvenMediaEngine</Name>
    <!-- Host type (origin/edge) -->
    <Type>origin</Type>
    <!-- Specify IP address to bind (* means all IPs) -->
    <IP>*</IP>
    <StunServer>stun.l.google.com:19302</StunServer>
    <Bind>
        <Providers>
            <RTMP>
                <Port>1935</Port>
            </RTMP>
            <WebRTC>
                <Signalling>
                    <Port>3333</Port>
                    <!-- <TLSPort>3333</TLSPort> -->
                </Signalling>
                <IceCandidates>
                    <TcpRelay>*:3478</TcpRelay>
                    <IceCandidate>*:10000-10005/udp</IceCandidate>
```

① Display Configuration Files

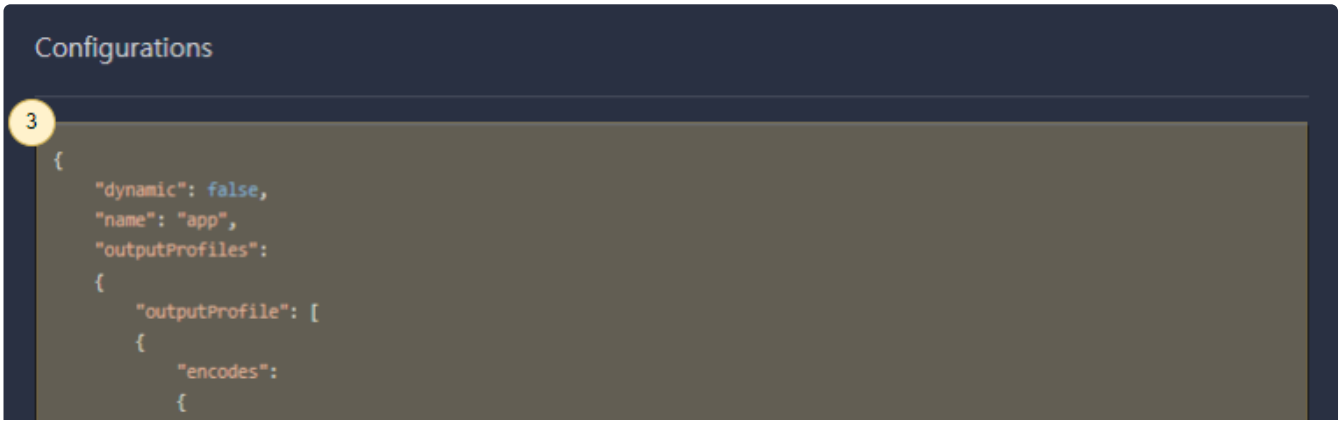
If you select Server, you will see all the configuration files used by OvenMediaEngine.

Virtual Host, Application, Stream Statistics, and Information

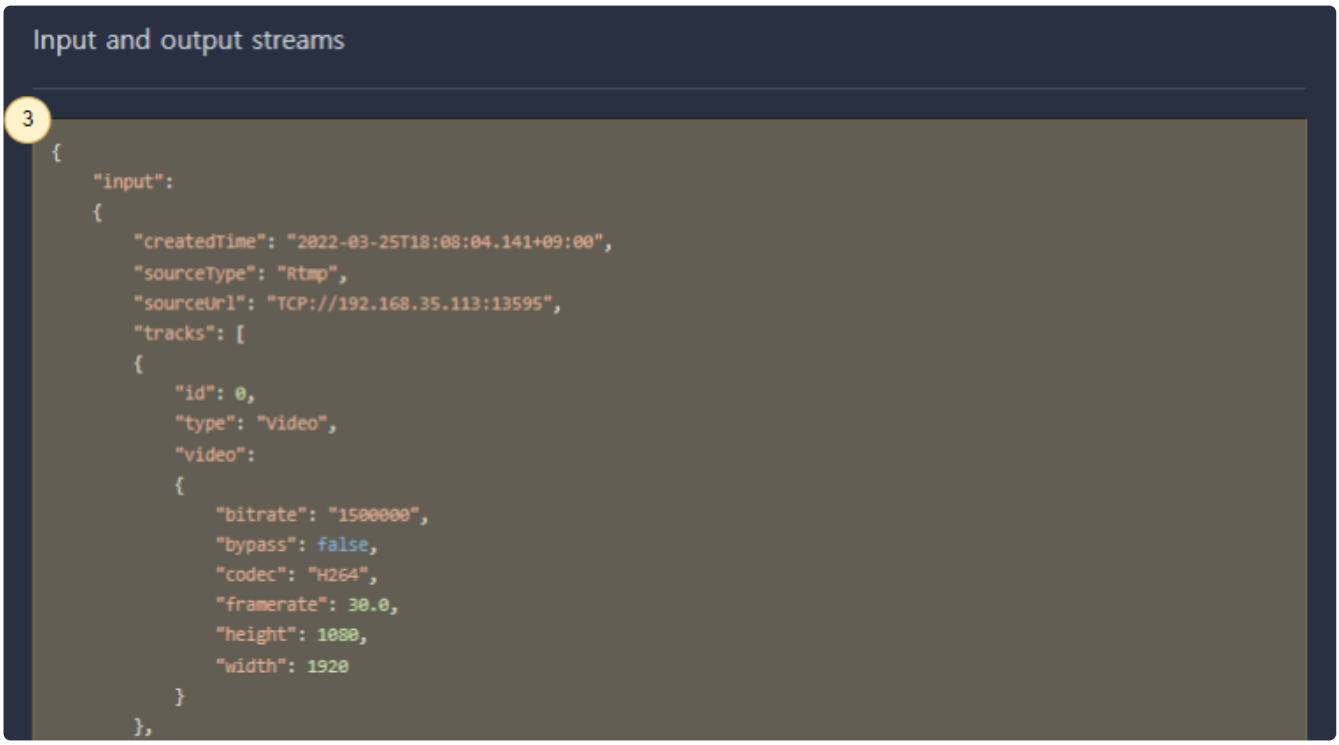
If you select Virtual Host, Application, or Stream, you can see traffic/connection statistics and configuration.



Display statistics



Display of configuration of Virtual Host, Application



Displaying input/output stream information

① Statistics

You can check the cumulative bytes in/out and the number of connected sessions for each protocol.

② Creation Time

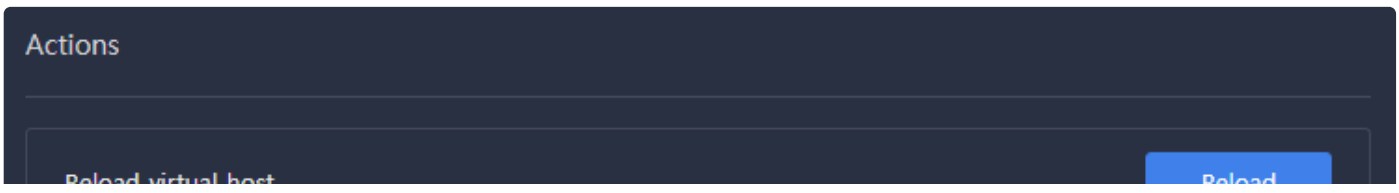
You can check the creation time of the selected resource.

③ Detailed Information

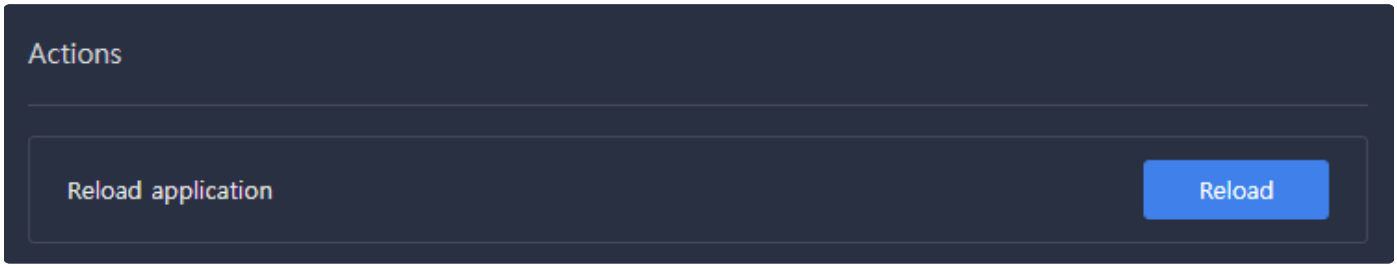
Display detailed information about the selected resource. When Virtual Host or Application is selected, the configuration is displayed, and when Stream is selected, the input stream and output stream information is displayed.

Reload Virtual Host or Application Dynamically

If you change the Virtual Host or Application settings directly in the Server.xml or include the Virtual Host configuration file, you can apply the changes to OvenMediaEngine without restarting OvenMediaEngine. Use the Actions → Reload button on the details page of Virtual Host or Application.



Reload Virtual Host



Reload Application

Add or Remove Virtual Hosts Dynamically

Suppose you modify the OvenMediaEngine Server.xml directly to add a new Virtual Host or remove an existing Virtual Host. In that case, you can reflect the added or removed Virtual Host to the OvenMediaEngine without restarting the OvenMediaEngine. Likewise, if you add or delete the separated Virtual Host configuration files, it will work similarly.

① Reload the OvenMediaEngine Configuration

To reload the modified configuration file, refresh each page or use the `Refresh` button in the left navigation.

② Add a new Virtual Host

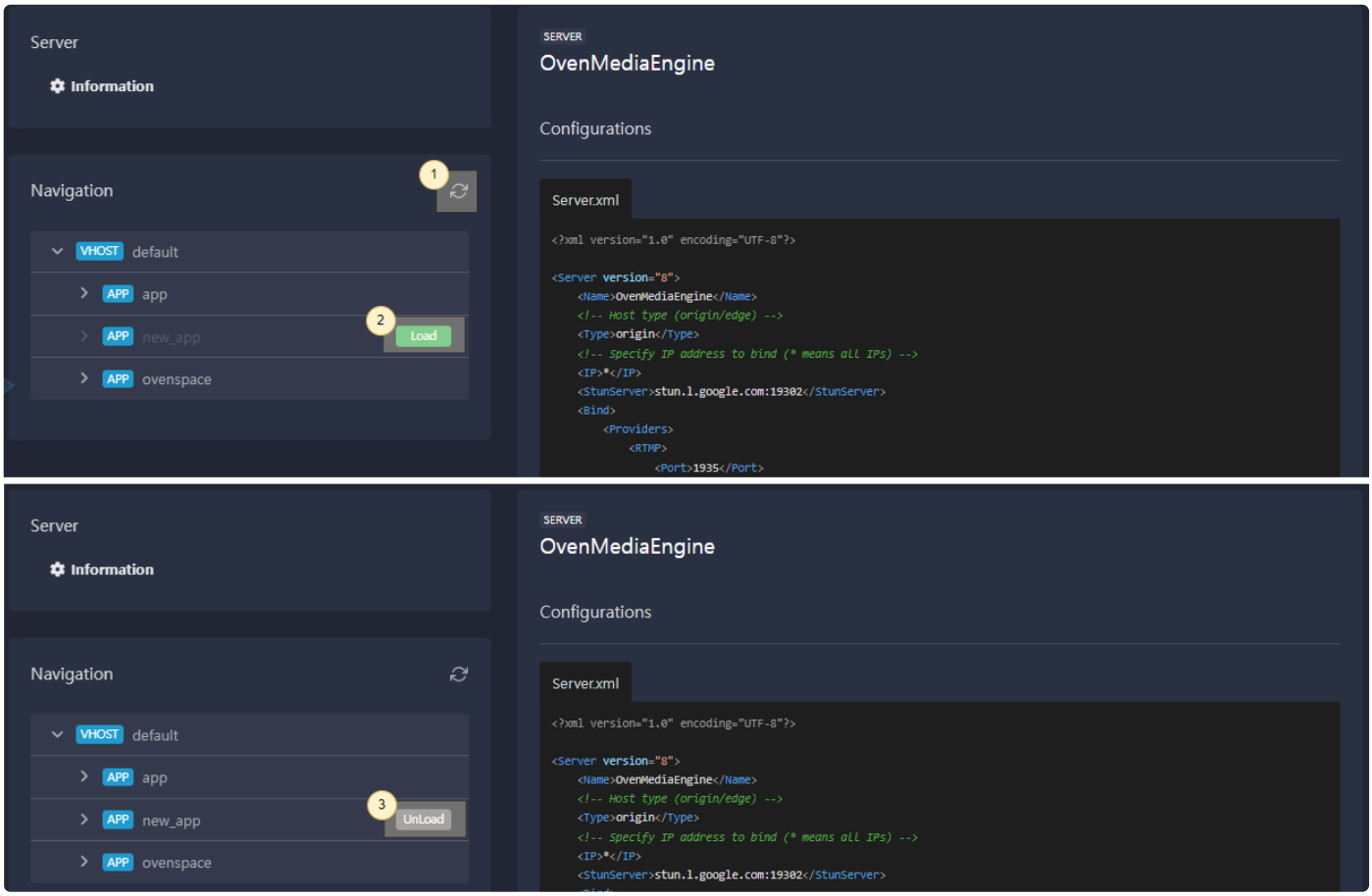
Suppose a new Virtual Host is added to the configuration file. In that case, the new Virtual Host will be added to the left navigation, and you will see a `Load` button that can be immediately reflected in the OvenMediaEngine.

③ Remove the existing Virtual Host

Similarly, if an existing Virtual Host is removed from the configuration file, you will see an `Unload` button in the left navigation. You can use the Unload button to remove a current Virtual Host from OvenMediaEngine.

Add or Remove Applications Dynamically

As with Virtual Host, if you modify the OvenMediaEngine configuration file directly to add or remove an Application, you can add or remove the Application to OvenMediaEngine without restarting OvenMediaEngine.



① Reload OvenMediaEngine Configuration

To reread the modified configuration file, refresh each page or use the `Refresh` button in the left navigation.

② Add a new Application

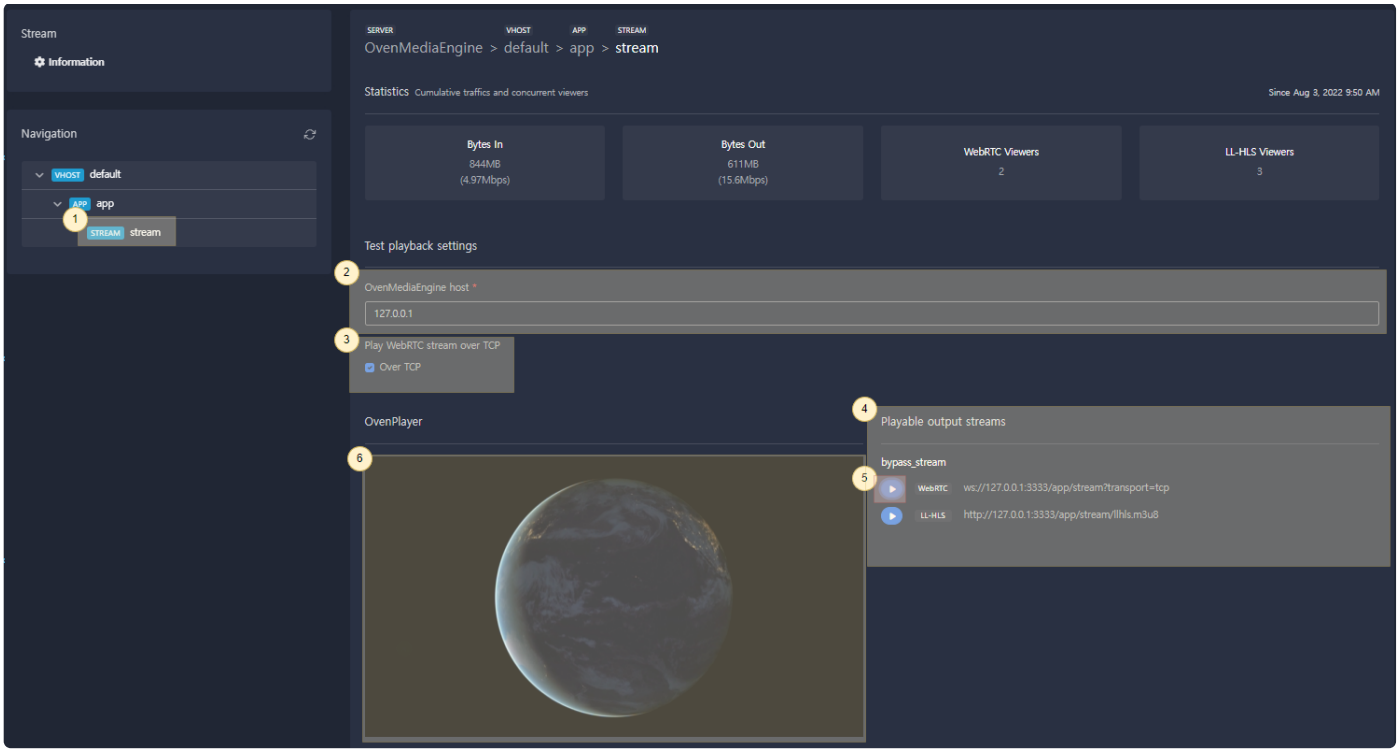
When a new Application is added to the config file, the new Application is added to the left navigation with a Load button that can be immediately applied to the OvenMediaEngine. You can use the Load button to apply the new Application to OvenMediaEngine.

③ Remove the existing Application

Similarly, if an existing Application is removed from the configuration file, an `Unload` button will appear in the left navigation.

Test Player

We provide a Test Player that can play that Stream.



① Test Playback Link

If you select a Stream, you can find the section where you can play the stream with the `Test Playback` link.

② OvenMediaEngine Host Settings

Set the IP address or domain of OvenMediaEngine (required).

> ⓘ Host information is different depending on the operating environment of OvenMediaEngine, so enter it manually.

③ TCP Playback Settings

Check if you want to play the WebRTC stream with TCP.

④ Output Stream List

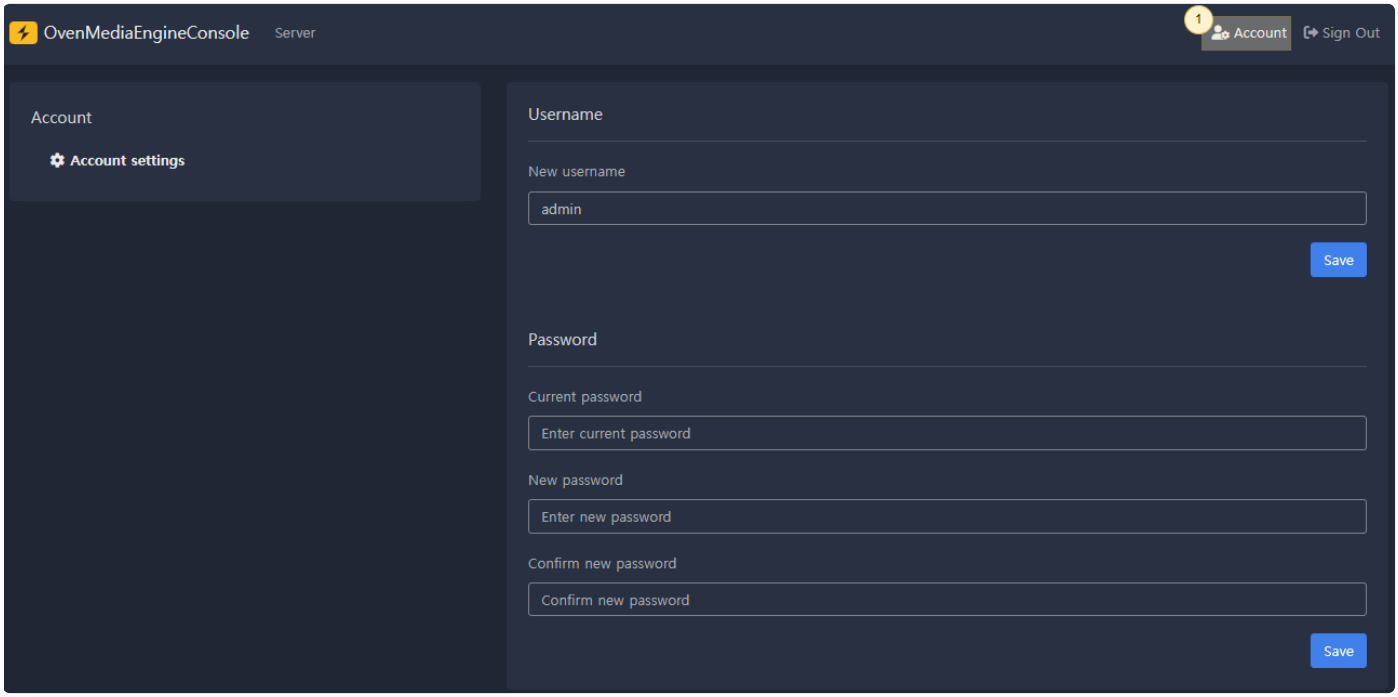
It interprets the information set in ② and ③ and the configurations of OvenMediaEngine and display the list of playable output streams.

⑤ Output Stream Playback

If you click the `Play` button, the stream is played with ⑥ Test Player (OvenPlayer).

Account Settings

You can change the account information to sign in to the Web Console.



① Account Setting Link

You can enter the account information page through the `Account` link.

The default account is set to username: `admin`, password: `admin`. If you change your account information, your session will be expired, and you will need to sign in again.

Release Note