
Python Janus Client

Lim Meng Kiat

Nov 19, 2023

CONTENTS

- 1 Key Features 3**
- 2 Library Installation 5**
- 3 Getting Started 7**
 - 3.1 Simple Connect And Disconnect 7
 - 3.2 Make Video Calls 7
- 4 Table Of Contents 9**
 - 4.1 Session 9
 - 4.2 Plugin 9
 - 4.3 Transport 13
- Index 15**

Easily send and share WebRTC media through [Janus](#) WebRTC server.

KEY FEATURES

- Supports HTTP/s and WebSockets communication with Janus.
- **Support Admin/Monitor API:**
 - Generic requests
 - Configuration related requests
 - Token related requests
- **Supports Janus plugin:**
 - EchoTest Plugin
 - VideoCall Plugin
 - VideoRoom Plugin
- Extendable Transport class and Plugin class

LIBRARY INSTALLATION

```
$ pip install janus-client
```


GETTING STARTED

3.1 Simple Connect And Disconnect

```
import asyncio
from janus_client import JanusSession, JanusEchoTestPlugin, JanusVideoRoomPlugin

# Protocol will be derived from base_url
base_url = "wss://janusmy.josephgetmyip.com/janusbasews/janus"
# OR
base_url = "https://janusmy.josephgetmyip.com/janusbase/janus"

session = JanusSession(base_url=base_url)

plugin_handle = JanusEchoTestPlugin()

# Attach to Janus session
await plugin_handle.attach(session=session)

# Destroy plugin handle
await plugin_handle.destroy()
```

This will create a plugin handle and then destroy it.

Notice that we don't need to call connect or disconnect explicitly. It's managed internally.

3.2 Make Video Calls

```
import asyncio
from janus_client import JanusSession, JanusVideoCallPlugin
from aiortc.contrib.media import MediaPlayer, MediaRecorder

async def main():
    # Create session
    session = JanusSession(
        base_url="wss://janusmy.josephgetmyip.com/janusbasews/janus",
    )

    # Create plugin
    plugin_handle = JanusVideoCallPlugin()
```

(continues on next page)

```
# Attach to Janus session
await plugin_handle.attach(session=session)

# Prepare username and media stream
username = "testusernamein"
username_out = "testusernameout"

player = MediaPlayer(
    "desktop",
    format="gdtgrab",
    options={
        "video_size": "640x480",
        "framerate": "30",
        "offset_x": "20",
        "offset_y": "30",
    },
)
recorder = MediaRecorder("./videocall_record_out.mp4")

# Register myself as testusernameout
result = await plugin_handle.register(username=username_out)

# Call testusernamein
result = await plugin_handle.call(
    username=username, player=player, recorder=recorder
)

# Wait awhile then hangup
await asyncio.sleep(30)

result = await plugin_handle.hangup()

# Destroy plugin
await plugin_handle.destroy()

# Destroy session
await session.destroy()

if __name__ == "__main__":
    try:
        asyncio.run(main())
    except KeyboardInterrupt:
        pass
```

This example will register to the VideoCall plugin using username *testusernameout*. It will then call the user registered using the username *testusernamein*.

A portion of the screen will be captured and sent in the call media stream. The incoming media stream will be saved into *videocall_record_out.mp4* file.

TABLE OF CONTENTS

4.1 Session

Create a session object that can be shared between plugin handles

4.1.1 Session Class

```
class janus_client.JanusSession(base_url: str = ", api_secret: Optional[str] = None, token: Optional[str]  
                                = None, transport: Optional[JanusTransport] = None)
```

Janus session instance

```
async attach_plugin(plugin: JanusPlugin) → int
```

Create plugin handle for the given plugin type

Parameters

plugin – Plugin instance with janus_client.JanusPlugin as base class

```
async create() → None
```

Initialize resources

```
async destroy() → None
```

Release resources

Should be called when you don't need the session anymore.

Plugins from this session should be destroyed before this.

4.2 Plugin

4.2.1 VideoRoom Plugin

```
class janus_client.JanusVideoRoomPlugin(*args, **kwargs)
```

Janus VideoRoom plugin implementation

Implements API to interact with VideoRoom plugin.

Each plugin object is expected to have only 1 PeerConnection.

Each VideoRoom plugin instance is expected to have one of the following three uses: - Administration - Publisher
- Subscriber

An instance meant for administration can be used as publisher or subscriber, but usually there isn't a reason to share. Just create another instance. On the other hand, a publisher instance cannot subscribe to a stream and vice versa.

async allowed(*room_id: int, secret: str = "", action: AllowedAction = AllowedAction.ENABLE, allowed: List[str] = []*) → bool

Configure ACL of a room.

async create_room(*room_id: int, configuration: dict = {}*) → bool

Create a room.

Refer to documentation for description of parameters. <https://janus.conf.meetecho.com/docs/videoroom.html>

async destroy_room(*room_id: int, secret: str = "", permanent: bool = False*) → bool

Destroy a room.

All other participants in the room will also get the “destroyed” event.

async edit(*room_id: int, secret: str = "", new_description: str = "", new_secret: str = "", new_pin: str = "", new_is_private: bool = False, new_require_pvtid: bool = False, new_bitrate: Optional[int] = None, new_fir_freq: Optional[int] = None, new_publishers: int = 3, new_lock_record: bool = False, new_rec_dir: Optional[str] = None, permanent: bool = False*) → bool

Edit a room.

async exists(*room_id: int*) → bool

Check if a room exists.

async join(*room_id: int, publisher_id: Optional[int] = None, display_name: str = "", token: Optional[str] = None*) → bool

Join a room

A handle can join a room and then do nothing, but this should be called before publishing. There is an API to configure and publish at the same time, but it's not implemented yet.

Parameters

- **room_id** – unique ID of the room to join.
- **publisher_id** – unique ID to register for the publisher; optional, will be chosen by the plugin if missing.
- **display_name** – display name for the publisher; optional.
- **token** – invitation token, in case the room has an ACL; optional.

Returns

True if room is created.

async kick(*room_id: int, id: str, secret: str = ""*) → bool

Kick a participant by ID.

Only works for room administrators (i.e. you created the room).

async leave() → bool

Leave the room. Will unpublish if publishing.

Returns

True if successfully leave.

async list_participants(*room_id: int*) → list

Get participant list in a room

Get a list of publishers in the room, that are currently publishing.

Parameters

room_id – List participants in this room

Returns

A list containing the participants. Can be empty.

async list_room() → List[dict]

List all rooms created.

If `admin_key` is included, then private rooms will be listed as well. TODO: Find out how to include `admin_key`.

async moderate(*room_id: int, id: str, mid: str, mute: bool, secret: str = ""*) → bool

Moderate a participant by ID.

Only works for room administrators (i.e. you created the room).

name: `str = 'janus.plugin.videoroom'`

Plugin name

async on_receive(*response: dict*)

Handle asynchronous messages

async pause() → None

Pause media streaming

async publish(*stream_track: List[MediaStreamTrack], configuration: dict = {}*) → None

Publish video stream to the room

Should already have joined a room before this.

async start(*jsdp: Optional[dict] = None*) → bool

Signal WebRTC start.

async subscribe_and_start(*room_id: int, on_track_created, stream: dict, use_msid: bool = False, autoupdate: bool = True, private_id: Optional[int] = None*) → bool

Subscribe to a feed. Only supporting subscribe to 1 stream.

Parameters

- **room_id** – Room ID containing the feed. The same ID that you would use to join the room.
- **on_track_created** – A callback function that will be called when AIORTC PC creates a media track
- **stream** – Configuration of the stream to subscribe to. Minimum should have a feed ID.
- **use_msid** – whether subscriptions should include an msid that references the publisher; false by default.
- **autoupdate** – whether a new SDP offer is sent automatically when a subscribed publisher leaves; true by default.
- **private_id** – unique ID of the publisher that originated this request; optional, unless mandated by the room configuration.

async unpublish() → bool

Stop publishing.

Returns

True if successfully unpublished.

async unsubscribe() → None

Unsubscribe from the feed

4.2.2 VideoCall Plugin

class janus_client.JanusVideoCallPlugin

Janus Video Call plugin implementation

name: str = 'janus.plugin.videocall'

Plugin name

async on_incoming_call(jsep: dict)

Override this. This will be called when plugin receive incoming call event

async on_receive(response: dict)

Handle asynchronous events from Janus

async register(username: str) → bool

Register a username

Detach plugin to de-register the username

4.2.3 EchoTest Plugin

class janus_client.JanusEchoTestPlugin

Janus EchoTest plugin implementation

async close_stream()

Close stream

This should cause the stream to stop and a done event to be received.

name: str = 'janus.plugin.echotest'

Plugin name

Must override to match plugin name in Janus server.

async on_receive(response: dict)

Handle asynchronous events from Janus

4.2.4 Base Class

class janus_client.JanusPlugin

Base class to inherit when implementing a plugin

async **destroy**()

Destroy plugin handle

async **send**(message: dict) → MessageTransaction

Send raw message to plugin

Will auto attach plugin ID to the message.

Parameters

message – JSON serializable dictionary to send

Returns

Synchronous reply from server

async **trickle**(sdpMLineIndex, candidate)

Send WebRTC candidates to Janus

Parameters

- **sdpMLineIndex** – (I don't know what is this)
- **candidate** – Candidate payload. (I got it from WebRTC instance callback)

4.3 Transport

Transport method is detected using regex on base_url parameter passed to Session object.

4.3.1 Base Class

class janus_client.JanusTransport(base_url: str, api_secret: Optional[str] = None, token: Optional[str] = None, **kwargs: dict)

Janus transport protocol interface

Manage Sessions and Transactions

__init__(base_url: str, api_secret: Optional[str] = None, token: Optional[str] = None, **kwargs: dict)

Create connection instance

Parameters

- **base_url** – Janus server address
- **api_secret** – (optional) API key for shared static secret authentication
- **token** – (optional) Token for shared token based authentication

abstract **async** **_send**(message: Dict) → None

Really sends the message. Doesn't return a response

static create_transport(*base_url: str, api_secret: Optional[str] = None, token: Optional[str] = None, config: Dict = {}*) → *JanusTransport*

Create transport class

JanusSession will call this to create the transport class automatically using base_url parameter.

Args:

base_url (str): _description_ api_secret (str, optional): _description_. Defaults to None. token (str, optional): _description_. Defaults to None. config (Dict, optional): _description_. Defaults to {}.

Raises:

Exception: No transport class found Exception: More than 1 transport class found

Returns:

JanusTransport: Use this object to communicate with Janus server.

async dispatch_session_created(*session_id: int*) → None

Override this method to get session created event

async dispatch_session_destroyed(*session_id: int*) → None

Override this method to get session destroyed event

async info() → Dict

Get info of Janus server. Will be overridden for HTTP.

static register_transport(*protocol_matcher, transport_cls: JanusTransport*) → None

Register transport class

Pass in a regex matcher and it will be used to match base_url to the transport class.

4.3.2 HTTP

class janus_client.JanusTransportHTTP(*base_url: str, api_secret: Optional[str] = None, token: Optional[str] = None, **kwargs: dict*)

Janus transport through HTTP

async dispatch_session_created(*session_id: str*) → None

Override this method to get session created event

async dispatch_session_destroyed(*session_id: int*) → None

Override this method to get session destroyed event

async info() → Dict

Get info of Janus server. Will be overridden for HTTP.

4.3.3 Websockets

class janus_client.JanusTransportWebsocket(***kwargs: dict*)

Janus transport through HTTP

Manage Sessions and Transactions

connected: bool

Must set this property when connected or disconnected

Symbols

`__init__()` (*janus_client.JanusTransport method*), 13
`_send()` (*janus_client.JanusTransport method*), 13

A

`allowed()` (*janus_client.JanusVideoRoomPlugin method*), 10
`attach_plugin()` (*janus_client.JanusSession method*), 9

C

`close_stream()` (*janus_client.JanusEchoTestPlugin method*), 12
`connected` (*janus_client.JanusTransportWebsocket attribute*), 14
`create()` (*janus_client.JanusSession method*), 9
`create_room()` (*janus_client.JanusVideoRoomPlugin method*), 10
`create_transport()` (*janus_client.JanusTransport static method*), 13

D

`destroy()` (*janus_client.JanusPlugin method*), 13
`destroy()` (*janus_client.JanusSession method*), 9
`destroy_room()` (*janus_client.JanusVideoRoomPlugin method*), 10
`dispatch_session_created()` (*janus_client.JanusTransport method*), 14
`dispatch_session_created()` (*janus_client.JanusTransportHTTP method*), 14
`dispatch_session_destroyed()` (*janus_client.JanusTransport method*), 14
`dispatch_session_destroyed()` (*janus_client.JanusTransportHTTP method*), 14

E

`edit()` (*janus_client.JanusVideoRoomPlugin method*), 10
`exists()` (*janus_client.JanusVideoRoomPlugin method*), 10

I

`info()` (*janus_client.JanusTransport method*), 14
`info()` (*janus_client.JanusTransportHTTP method*), 14

J

`JanusEchoTestPlugin` (*class in janus_client*), 12
`JanusPlugin` (*class in janus_client*), 13
`JanusSession` (*class in janus_client*), 9
`JanusTransport` (*class in janus_client*), 13
`JanusTransportHTTP` (*class in janus_client*), 14
`JanusTransportWebsocket` (*class in janus_client*), 14
`JanusVideoCallPlugin` (*class in janus_client*), 12
`JanusVideoRoomPlugin` (*class in janus_client*), 9
`join()` (*janus_client.JanusVideoRoomPlugin method*), 10

K

`kick()` (*janus_client.JanusVideoRoomPlugin method*), 10

L

`leave()` (*janus_client.JanusVideoRoomPlugin method*), 10
`list_participants()` (*janus_client.JanusVideoRoomPlugin method*), 10
`list_room()` (*janus_client.JanusVideoRoomPlugin method*), 11

M

`moderate()` (*janus_client.JanusVideoRoomPlugin method*), 11

N

`name` (*janus_client.JanusEchoTestPlugin attribute*), 12
`name` (*janus_client.JanusVideoCallPlugin attribute*), 12
`name` (*janus_client.JanusVideoRoomPlugin attribute*), 11

O

`on_incoming_call()` (*janus_client.JanusVideoCallPlugin method*), 12

`on_receive()` (*janus_client.JanusEchoTestPlugin method*), [12](#)
`on_receive()` (*janus_client.JanusVideoCallPlugin method*), [12](#)
`on_receive()` (*janus_client.JanusVideoRoomPlugin method*), [11](#)

P

`pause()` (*janus_client.JanusVideoRoomPlugin method*), [11](#)
`publish()` (*janus_client.JanusVideoRoomPlugin method*), [11](#)

R

`register()` (*janus_client.JanusVideoCallPlugin method*), [12](#)
`register_transport()` (*janus_client.JanusTransport static method*), [14](#)

S

`send()` (*janus_client.JanusPlugin method*), [13](#)
`start()` (*janus_client.JanusVideoRoomPlugin method*), [11](#)
`subscribe_and_start()` (*janus_client.JanusVideoRoomPlugin method*), [11](#)

T

`trickle()` (*janus_client.JanusPlugin method*), [13](#)

U

`unpublish()` (*janus_client.JanusVideoRoomPlugin method*), [11](#)
`unsubscribe()` (*janus_client.JanusVideoRoomPlugin method*), [12](#)