**Classes**

*class*aiortc.**RTCPeerConnection**(*configuration=None*)

The **[RTCPeerConnection](https://aiortc.readthedocs.io/en/latest/api.html" \l "aiortc.RTCPeerConnection" \o "aiortc.RTCPeerConnection)** interface represents a WebRTC connection between the local computer and a remote peer.

*class*aiortc.**RTCSessionDescription**(*sdp*, *type*)

The **[RTCSessionDescription](https://aiortc.readthedocs.io/en/latest/api.html" \l "aiortc.RTCSessionDescription" \o "aiortc.RTCSessionDescription)** dictionary describes one end of a connection and how it’s configured.

*class*aiortc.**RTCConfiguration**(*iceServers=None*)

The **[RTCConfiguration](https://aiortc.readthedocs.io/en/latest/api.html" \l "aiortc.RTCConfiguration" \o "aiortc.RTCConfiguration)** dictionary is used to provide configuration options for an [**RTCPeerConnection**](https://aiortc.readthedocs.io/en/latest/api.html#aiortc.RTCPeerConnection).

*class*aiortc.**RTCDataChannel**(*transport*, *parameters*, *send\_open=True*)

The **[RTCDataChannel](https://aiortc.readthedocs.io/en/latest/api.html" \l "aiortc.RTCDataChannel" \o "aiortc.RTCDataChannel)** interface represents a network channel which can be used for bidirectional peer-to-peer transfers of arbitrary data.

*class*aiortc.**RTCDataChannelParameters**(*label=''*, *maxPacketLifeTime=None*, *maxRetransmits=None*, *ordered=True*, *protocol=''*, *negotiated=False*, *id=None*)

The **[RTCDataChannelParameters](https://aiortc.readthedocs.io/en/latest/api.html" \l "aiortc.RTCDataChannelParameters" \o "aiortc.RTCDataChannelParameters)** dictionary describes the configuration of an **[RTCDataChannel](https://aiortc.readthedocs.io/en/latest/api.html" \l "aiortc.RTCDataChannel" \o "aiortc.RTCDataChannel)**.

*class*aiortc.**MediaStreamTrack**

A single media track within a stream.

**Functions**

createAnswer

Create an SDP answer to an offer received from a remote peer during

the offer/answer negotiation of a WebRTC connection.

createOffer

Create an SDP offer for the purpose of starting a new WebRTC

connection to a remote peer.

addTrack

class:`MediaStreamTrack` to the set of media tracks which

will be transmitted to the remote peer.

createDataChannel

Create a data channel with the given label.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Code

import socket

SERVER = socket.gethostbyname(socket.gethostname())

PORT = 5050

ADDR = (SERVER, PORT)

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(ADDR)

def handle(conn, addr):

print('handling')

def start():

server.listen()

while True:

conn, addr = server.accept()

thread = threading.Thread(target=handle, args=(conn, addr))

thread.start()

start()