```python
from dotenv import load_dotenv
load_dotenv()
import os
import ssl
import time
import json
import uuid
import threading
from datetime import datetime, timezone

import paho.mqtt.client as mqtt

# --- Config (from env, defaults for quick tests) ---
DEVICE_ID = os.getenv("DEVICE_ID", "device-1")
DEVICE_PASSWORD = os.getenv("DEVICE_PASSWORD", "")
MQTT_HOST = os.getenv("MQTT_HOST", "tvs-dev.ifactory.ai")
BROKER_PORT = int(os.getenv("MQTT_PORT", "8883"))
PUBLISH_INTERVAL = int(os.getenv("PUBLISH_INTERVAL", "5"))  # seconds

PUBLISH_TOPIC = f"/d2c/{DEVICE_ID}"
SUBSCRIBE_TOPIC = f"/c2d/{DEVICE_ID}"

# --- Message format identical to your device.js getSampleMessage() ---
def get_sample_message():
    return {
        "data": {
            "ueid": str(uuid.uuid4()),                    # unique event id
per message
            "ts": datetime.now(timezone.utc).isoformat(), # ISO 8601
timestamp in UTC
            "did": DEVICE_ID,
            "eid": 1,
            "pl": {
                "ms": 1
            }
        }
    }

# --- Callbacks ---
def on_connect(client, userdata, flags, rc, properties=None):
    if rc == 0:
        print(f"{DEVICE_ID} connected (rc={rc})")
```

```
        # mirror Node.js behavior: subscribe after 20s
        def do_subscribe():
            result, mid = client.subscribe(SUBSCRIBE_TOPIC, qos=1)
            print(f"Subscribe request sent -> result: {result}, mid:
{mid}")
        t = threading.Timer(20.0, do_subscribe)
        t.daemon = True
        t.start()
    else:
        print(f"Failed to connect, rc={rc}")

def on_message(client, userdata, msg):
    print(f"Received message: {msg.payload.decode()} on {msg.topic}
(qos={msg.qos})")

def on_disconnect(client, userdata, rc, properties=None):
    print(f"Disconnected, rc={rc}")

# --- Client setup (MQTT v5) ---
client = mqtt.Client(client_id=DEVICE_ID, protocol=mqtt.MQTTv5)
client.username_pw_set(DEVICE_ID, DEVICE_PASSWORD)

client.tls_set(cert_reqs=ssl.CERT_NONE)
client.tls_insecure_set(True)

client.on_connect = on_connect
client.on_message = on_message
client.on_disconnect = on_disconnect

print("Connecting with:")
print("  Host:", MQTT_HOST)
print("  Username:", DEVICE_ID)
print("  Password:", "(set)" if DEVICE_PASSWORD else "(empty)")
print("  Client ID:", DEVICE_ID)

client.connect(MQTT_HOST, BROKER_PORT, keepalive=60)
client.loop_start()

try:
    while True:
        message = get_sample_message()
        client.publish(PUBLISH_TOPIC, json.dumps(message), qos=1)
        print("Message sent:", message)
```

```
        time.sleep(PUBLISH_INTERVAL)
except KeyboardInterrupt:
    print("Exiting...")
finally:
    client.loop_stop()
    client.disconnect()
```