```
Help on module cayenne.client in cayenne:

NAME
    cayenne.client

CLASSES
    builtins.object
        CayenneMQTTClient
        CayenneMessage

    class CayenneMQTTClient(builtins.object)
     |  Cayenne MQTT Client class.
     |
     |  This is the main client class for connecting to Cayenne and sending and
receiving data.
     |
     |  Standard usage:
     |  * Set on_message callback, if you are receiving data.
     |  * Connect to Cayenne using the begin() function.
     |  * Call loop() at intervals (or loop_forever() once) to perform message
processing.
     |  * Send data to Cayenne using write functions: virtualWrite(),
celsiusWrite(), etc.
     |  * Receive and process data from Cayenne in the on_message callback.
     |
     |  The on_message callback can be used by creating a function and assigning
it to CayenneMQTTClient.on_message member.
     |  The callback function should have the following signature:
on_message(message)
     |  The message variable passed to the callback is an instance of the
CayenneMessage class.
     |
     |  Methods defined here:
     |
     |  begin(self, username, password, clientid, hostname='mqtt.mydevices.com',
port=1883)
     |      Initializes the client and connects to Cayenne.
     |
     |      username is the Cayenne username.
     |      password is the Cayenne password.
     |      clientID is the Cayennne client ID for the device.
     |      hostname is the MQTT broker hostname.
     |      port is the MQTT broker port. Use port 8883 for secure connections.
     |
     |  celsiusWrite(self, channel, value)
     |      Send a Celsius value to Cayenne.
     |
     |      channel is the Cayenne channel to use.
     |      value is the data value to send.
     |
     |  fahrenheitWrite(self, channel, value)
     |      Send a Fahrenheit value to Cayenne.
     |
     |      channel is the Cayenne channel to use.
     |      value is the data value to send.
     |
     |  getCommandTopic(self)
     |      Get the command topic string.
     |
     |  getDataTopic(self, channel)
     |      Get the data topic string.
     |
     |      channel is the channel to send data to.
     |
```

```
 |  getResponseTopic(self)
 |      Get the response topic string.
 |
 |  hectoPascalWrite(self, channel, value)
 |      Send a hectopascal value to Cayenne.
 |
 |      channel is the Cayenne channel to use.
 |      value is the data value to send.
 |
 |  kelvinWrite(self, channel, value)
 |      Send a kelvin value to Cayenne.
 |
 |      channel is the Cayenne channel to use.
 |      value is the data value to send.
 |
 |  loop(self)
 |      Process Cayenne messages.
 |
 |      This should be called regularly to ensure Cayenne messages are sent
and received.
 |
 |  loop_forever(self)
 |      Process Cayenne messages in a blocking loop that runs forever.
 |
 |  luxWrite(self, channel, value)
 |      Send a lux value to Cayenne.
 |
 |      channel is the Cayenne channel to use.
 |      value is the data value to send.
 |
 |  mqttPublish(self, topic, payload)
 |      Publish a payload to a topic
 |
 |      topic is the topic string.
 |      payload is the payload data.
 |
 |  pascalWrite(self, channel, value)
 |      Send a pascal value to Cayenne.
 |
 |      channel is the Cayenne channel to use.
 |      value is the data value to send.
 |
 |  responseWrite(self, msg_id, error_message)
 |      Send a command response to Cayenne.
 |
 |      This should be sent when a command message has been received.
 |      msg_id is the ID of the message received.
 |      error_message is the error message to send. This should be set to
None if there is no error.
 |
 |  virtualWrite(self, channel, value, dataType='', dataUnit='')
 |      Send data to Cayenne.
 |
 |      channel is the Cayenne channel to use.
 |      value is the data value to send.
 |      dataType is the type of data.
 |      dataUnit is the unit of the data.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
```

```
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  client = None
 |
 |  connected = False
 |
 |  on_message = None
 |
 |  reconnect = False
 |
 |  rootTopic = ''

class CayenneMessage(builtins.object)
 |  CayenneMessage(msg)
 |
 |  This is a class that describes an incoming Cayenne message. It is
 |  passed to the on_message callback as the message parameter.
 |
 |  Members:
 |
 |  client_id : String. Client ID that the message was published on.
 |  topic : String. Topic that the message was published on.
 |  channel : Int. Channel that the message was published on.
 |  msg_id : String. The message ID.
 |  value : String. The message value.
 |
 |  Methods defined here:
 |
 |  __init__(self, msg)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

FUNCTIONS
    on_connect(client, cayenne, flags, rc)
        # The callback for when the client receives a CONNACK response from the
server.

    on_disconnect(client, cayenne, rc)
        # The callback for when the client disconnects from the server.

    on_message(client, cayenne, msg)
        # The callback for when a PUBLISH message is received from the server.

DATA
    COMMAND_TOPIC = 'cmd'
    DATA_TOPIC = 'data'
    PROTOCOL_TLSv1_2 = <_SSLMethod.PROTOCOL_TLSv1_2: 5>
    RESPONSE_TOPIC = 'response'
    TYPE_BAROMETRIC_PRESSURE = 'bp'
```

```
    TYPE_BATTERY = 'batt'
    TYPE_LUMINOSITY = 'lum'
    TYPE_PROXIMITY = 'prox'
    TYPE_RELATIVE_HUMIDITY = 'rel_hum'
    TYPE_TEMPERATURE = 'temp'
    TYPE_VOLTAGE = 'voltage'
    UNIT_CELSIUS = 'c'
    UNIT_CENTIMETER = 'cm'
    UNIT_DIGITAL = 'd'
    UNIT_FAHRENHEIT = 'f'
    UNIT_HECTOPASCAL = 'hpa'
    UNIT_KELVIN = 'k'
    UNIT_LUX = 'lux'
    UNIT_METER = 'm'
    UNIT_MILLIVOLTS = 'mv'
    UNIT_PASCAL = 'pa'
    UNIT_PERCENT = 'p'
    UNIT_RATIO = 'r'
    UNIT_UNDEFINED = 'null'
    UNIT_VOLTS = 'v'

VERSION
    1.1.0

FILE
    c:\program files (x86)\python37-32\lib\site-packages\cayenne\client.py
```