# API Reference Template

## AWS IoT Core Client API

*A client for connecting to AWS IoT Core, managing MQTT connections, and handling device shadows.*

### Base Class

* **Aws**

**class** **Aws(object):**

"""

AWS IoT Core client

"""

**def** \_\_init\_\_**(**self**,** client\_id**,** server**,** port**,** keep\_alive**=**60**,** ssl**=False,** ssl\_params**=None):**

self**.***client\_id* **=** client\_id

self**.***mqtt\_client* **=** MQTTClientWrapper**(**client\_id**,** server**,** port**,** keep\_alive**,** ssl**,** ssl\_params

self**.***shadow\_manager* **=** ShadowManager**(**self**.***mqtt\_client***,** client\_id**)**

self**.***logging* **=** log**.***getLogger***(**"AWS"**)**

**Description:** The Aws class provides an interface for connecting to AWS IoT Core, managing MQTT topics, and interacting with AWS IoT Shadows. In initizalization phase we make mqtt\_client object for using umqtt, and shadow\_manager object for using shadows feature.

**Initialization**

Aws(client\_id, server, port, keep\_alive=60, ssl=False, ssl\_params=None)

* **client\_id** (str): The unique identifier for the IoT Thing.
* **server** (str): The AWS IoT Core endpoint.
* **port** (int): The MQTT port (default 8883 for TLS, 1883 for non-TLS).
* **keep\_alive** (int, optional): The keep-alive interval in seconds (default: 60).
* **ssl** (bool, optional): Whether to use SSL/TLS (default: False).
* **ssl\_params** (dict, optional): SSL parameters for secure connection.

### Methods

**1. connect()**

**Description:** Establishes a connection to AWS IoT Core via umqtt connect method.

**Example**

aws\_client **=** Aws**(**"my\_device"**,** "a1b2c3d4e5f6.iot.amazonaws.com"**,** 8883**,** ssl**=True)**

aws\_client**.***connect***()**

**2. disconnect()**

**Description:** Disconnects from AWS IoT Core via umqtt disconnect method.

**Example**

aws\_client**.***disconnect***()**

**3. subscribe**(topic)

**Description:** Subscribes to an MQTT topic.

* **topic** (str): The MQTT topic to subscribe to.

**Example**

aws\_client**.***subscribe***(**"my/topic"**)**

**4. publish**(topic, payload)

**Description:** Publishes a message to an MQTT topic.

* **topic** (str): The MQTT topic.
* **payload** (str/dict): The message payload.

**Example**

data **=** **{**"temperature"**:** 25**,** "humidity"**:** 60**}**

aws\_client**.***publish***(**"sensor/data"**,** data**)**

## Shadows

**5. create\_shadow**(shadow\_name="", state="")

**Description:** Creates an IoT shadow with an optional name and state.

* **shadow\_name** (str, optional): The name of the shadow (default: unnamed shadow).
* **state** (dict/str, optional): The initial shadow state.

**Example**

shadow\_state **=** **{**"reported"**:** **{**"status"**:** "active"**}}**

aws\_client**.***create\_shadow***(**"deviceShadow"**,** shadow\_state**)**

**6. update\_shadow**(shadow\_name="", state="")

**Description:** Updates the state of an IoT shadow.

**Example**

shadow\_update **=** **{**"desired"**:** **{**"temperature"**:** 22**}}**

aws\_client**.***update\_shadow***(**"deviceShadow"**,** shadow\_update**)**

**7. get\_shadow**(shadow\_name="")

**Description:** Retrieves the current state of an IoT shadow.

**Example**

aws\_client**.***get\_shadow***(**"deviceShadow"**)**

**8. delete\_shadow**(shadow\_name="")

**Description:** Deletes an IoT shadow.

**Example**

aws\_client**.***delete\_shadow***(**"deviceShadow"**)**

**9. connect\_shadow**(shadow\_name="", topics=None)

**Description**: Establishes a connection to an IoT shadow and subscribes to related topics.

* **shadow\_name** (str, optional): The name of the shadow.
* **topics** (list, optional): List of topics to subscribe to.

**Example**

aws\_client**.***connect\_shadow***(**"deviceShadow"**,** **[**"shadow/update"**,** "shadow/get"**])**

**10. set\_callback**(topic\_name, callback)

**Description**: Sets a callback function for a specified MQTT topic.

* **topic\_name** (str): The topic name.
* **callback (function)**: The function to be executed when a message is received.

**Example**

**def** message\_callback**(**topic**,** msg**):**

**print(**f"Received message from {topic}: {msg}"**)**

aws\_client**.***set\_callback***(**"sensor/data"**,** message\_callback**)**

## Python example

**import** usr**.***aws* **as** aws

**import** modem

**import** ujson

**import** sim #check if pin verification is needed for your sim card, before running the script

**import** net

# certificate.pem.crt

certificate\_content **=** """

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

"""

# private key

private\_content **=** """

-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----

"""

# device name

client\_id **=** 'qpthing'

# server address

server **=** 'abgka7vzgjoa0-ats.iot.eu-west-3.amazonaws.com'

# port

port **=** 8883

**def** aws\_callback**(**data**):**

"""

aws callback

make your callback here

"""

**print(**"HELLO from 1234 topic callback"**)**

**def** shadow\_callback\_get**(**data**):**

"""

aws shadow callback

make your callback here

"""

**print(**" HELLO from get accepted callback"**)**

**def** shadow\_callback\_update**(**data**):**

"""

aws shadow callback

make your callback here

"""

**print(**" HELLO from update accepted callback"**)**

**def** shadow\_callback\_delta**(**data**):**

"""

aws shadow callback

make your callback here

"""

**print(**" ELLO from update delta callback"**)**

# create aws obj

aws\_obj **=** aws**.***Aws***(**client\_id**,** server**,** port**,** keep\_alive**=**60**,**ssl**=True,**ssl\_params**={**"cert"**:**

certificate\_content**,**"key"**:** private\_content**})**

**print(**"create aws obj"**)**

# connect mqtt server

**print(**"aws connect start"**)**

aws\_obj**.***connect***()**

**print(**"aws connect end"**)**

# register callback for '1234' topic

aws\_obj**.***set\_callback***(**"1234"**,**aws\_callback**)**

**print(**"aws set callback"**)**

#subscribe server topic

aws\_obj**.***subscribe***(**"1234"**)**

**print(**"aws subscribe"**)**

#publish to 7777 topic. Subscribe from AWS Console -> MQTT Client before running the script, to see the messsage.

aws\_obj**.***publish***(**"7777"**,**"Hello from QuecPython"**)**

aws\_obj**.***start***()**

'''

SHADOWS

'''

#creating unnamed shadow with default state

aws\_obj**.***create\_shadow***()**

#connecting shadow by subscribing it to all topics

aws\_obj**.***connect\_shadow***()**

#setting callbacks

aws\_obj**.***set\_callback***(**"$aws/things/qpthing/shadow/get/accepted"**,**shadow\_callback\_get**)**

**print(**"shadow\_callback\_get\_accepted\_set"**)**

aws\_obj**.***set\_callback***(**"$aws/things/qpthing/shadow/update/accepted"**,**shadow\_callback\_update**)**

**print(**"shadow\_callback\_update\_accepted\_set"**)**

aws\_obj**.***set\_callback***(**"$aws/things/qpthing/shadow/update/accepted"**,**shadow\_callback\_delta**)**

**print(**"shadow\_callback\_delta\_set"**)**

#using get on unnamed shadow

aws\_obj**.***get\_shadow***()**

#using update on unnamed shadow

aws\_obj**.***update\_shadow***(**state**={**"state"**:{**"reported"**:{**"welcome"**:** "change reported"**}}})**