inovonics ®

Introduction

This guide will walk through the process of connecting your E*4080 IP gateway to the Amazon Web Services (AWS) Internet of Things (IoT) platform. It will assume that you already have an account with AWS and permissions to access/change settings in the IoT dashboard.

Contact Information

For questions, contact Inovonics technical support:
- E-mail: support@inovonics.com.
- Phone: (800) 782-2709; (303) 939-9336.

Materials Needed

- An AWS account.
- The ability to connect to Amazon servers outside of your firewall.
- Traffic allowed on the following ports:
  - Port 9000 (TCP for OTA upgrades)
  - Port 123 (UDP for NTP sync)
  - Port 8883 (standard MQTT port)
  - Port 443 (HTTPS)
- USB flash drive.
- A computer.

Setup Credentials for the IP Gateway on AWS

This procedure will setup credentials for the gateway on AWS. When completed, you will have the AWS certificate files to put on your IP gateway using the USB flash drive.

1. On your computer, navigate to the AWS IoT dashboard and select "Manage."
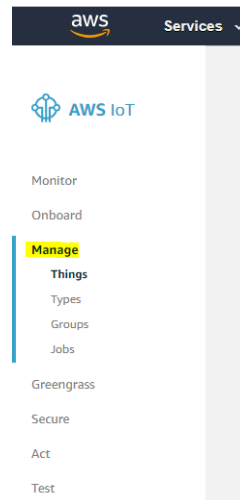

Figure 1 Select "Manage" on the AWS IoT dashboard.

2. Select "Create" in the top right of the AWS IoT dashboard.

3. Select "Create a Single Thing."

4. Give your thing a name
   • No other fields are required.

5. Select "Next."

6. Select "One-Click certificate creation."

7. Download all files except for the public key.

Figure 2 Download certificates and keys.

Note: The "Download" link opens a new tab where you select the files.

8. Click "Activate."

9. Click "Done."
   • The thing has been created with AWS.

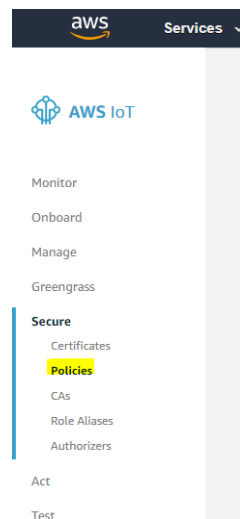**10.** Navigate to the "Secure" tab and select "Policies."

Figure 3 Select "Policies" on the AWS IoT dashboard.

11. Select "Create."

12. Give your policy a name.

13. Set policies according to your standards, or enter the following:
- Action: iot:*
- Resource ARN: *

14. Check "Allow" under "Create."

15. Click "Create."

16. Navigate to the "Secure" tab and select "Certificates."

17. Click the three dots on the certificate you generated and select "Attach Policy."
- The alphanumeric code on your certificate will match the key files you downloaded in step 7.
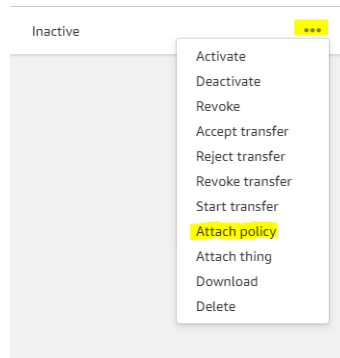

Figure 4 Attach a policy.

18. Select your new policy and click "Attach."

19. Click the three dots on the certificate you generated and select "Attach thing."

Connect the IP Gateway to AWS

This procedure will establish a connection between your IP gateway and AWS.

1. Referring to the IP gateway's installation instructions, download the configuration .yaml file and gateway license file onto a blank USB flash drive.

2. Keeping the AWS IoT dashboard open as a reference, copy the three files downloaded earlier onto the USB flash drive.
- All of the files that need to be on the USB flash drive are shown in Figure 5. These include the configuration .yaml file,

the IP gateway license file, and three files downloaded in step 7.

Name	Date modified	Type	Size
1a2b3c-certificate.pem	7/3/2018 2:18 PM	Security Certificate	2 KB
1a2b3c-private.pem.key	7/3/2018 2:18 PM	KEY File	2 KB
1a2b3c-public.pem.key	7/3/2018 2:18 PM	KEY File	1 KB
en4080-config.yaml	7/13/2018 2:48 PM	Text Document	0 KB
gateway-license.iwc	7/13/2018 2:48 PM	Text Document	0 KB

Figure 5 The files necessary on your USB flash drive.

3. Open the configuration file on the USB flash drive in any text editor.
   - Figure 6 is an example of a configuration file, with unrelated lines omitted for clarity. The reporter is specified, and then followed with a section for reporter settings:

```
plugin:
  awsmqtt:
    host: acvqxe0o75onu-ats.iot.us-west-2.amazonaws.com
    port: 8883
    root_ca: root.pem
    certificate: certificate.pem.crt
    private_key: private.pem.key
    client_id_prefix: inovonics
    client_id_fields: model-number, serial-number-unit
    reporting_format: json
    publish_message_topic: messages
    publish_discovery_topic: discovery-messages
    subscribe_conf_topic: settings
    publish_conf_topic: configuration
    batching: false
    qos: 1
```

Figure 6 Configuration file example.

4. Enter configuration information in each field of the configuration file.

Note: Any field marked with an * can be changed later remotely; any field not marked with an * must be edited using the USB flash drive.

   - host: this will be a url, which can be found in the IoT dashboard.
     – Under the "Manage" tab, select "Things", then your new thing, then "Interact."
     – Locate the url ending in ".amazonaws.com" which on the top of the page. This is your host.
   - port: This indicates which port your IP gateway communicates on and will be 8883 in almost all cases.
   - root_ca: This is the name of the root cert file placed on the USB flash drive.
   - certificate: This is the name of the device's certificate file placed on the USB flash drive.
   - private_key: This is the name of the private key placed on the USB flash drive.
   - client_id_prefix: This is the string for your client id to start with.

- client_id_fields: This is a list of values which will be appended to the base client_id_prefix with dashes. Possible values: "mac-wired", "model-number", "serial-number-board" and "serial-number-unit". See example table below:

client-id-prefix	client-id-fields	Resulting client ID
inovonics	serial-number-unit	inovonics-VP1234567
my-gateway		my-gateway
inovonics	serial-number-unit, mac-wired	inovonics-VP1234567-F8:DC:7A:13:6D:1A

- *publish_message_topic: This is the topic your device will push its messages to; it will be used to receive messages in AWS.
- subscribe_conf_topic: This is the topic your device will receive messages from, and is used to change settings remotely. If you will not be configuring settings remotely, this can be left blank.
- publish_conf_topic: The IP gateway will send an up-to-date copy of your configuration settings after each modification. This can be the same as subscribe_conf_topic.
- *discovery_mode: If true, this will activate discovery mode and send messages on your defined publish_discovery_topic. This feature times out after 30 minutes and must be reactivated.
- *publish_discovery_topic: Messages from unregistered devices will transmit on this topic.
- qos: This stands for quality of service and is a part of MQTT. It will be an integer (0 or 1). A setting of 1 is recommended in most cases. This impacts most messages, but discovery mode messages always use a QOS of 0. A QOS of 2 is not currently supported by AWS, and cannot be used here.
- *txids: This is a list of txids to collect messages from.
- batching: If true, this sends messages in groups to save resources. This is recommended, as otherwise each message will be transmitted separately.
- reporting_format: This can be JSON or bytearray. Most of the time it will be JSON.

Note: We suggest making a copy of this file once it is complete and updating the file as changes are made remotely. This will give you a backup of all settings.

5. Referring to the IP gateway's installation instructions, use the USB flash drive to boot the IP gateway. The IP gateway should now be connected to AWS.
   - The orange connection light on the IP gateway should light and remain illuminated after booting completes.

Test Connection

1. Select "Test" on the IoT dashboard.

2. Input the topic specified in your configuration file for "pub_msg_topic" in the "Subscription topic" field.

3. Click "Subscribe to topic." Messages should begin to appear, as long as transmitters are sending messages to the IP gateway.

Change Settings

This section describes how to change settings on an IP gateway that is already in use.

Your device can receive settings via messages over the subscribe_conf_topic specific in the config file. This can be done in a variety of ways, but we'll showcase the simplest method here.

1. Navigate to the "Test" tab of AWS IoT dashboard.

2. Enter the name of your subscribe_conf_topic.

3. Click "Subscribe to topic"
   • This will use your configured QOS, so be sure to match your configured QOS setting with the QOS setting here.

4. Add the amended settings in valid JSON.

5. Click "Publish to topic."

Figure 7 Update settings received

The change should take effect almost instantly. Each time a modification is made to settings via MQTT, a message will be sent on your publish_conf_topic that shows the current state of each editable setting, including the most recent change.

update-status can be:

• Success: Configuration was updated successfully.

- Warning: Configuration was updated but with potential problems.
- Failure: Configuration failed to adopt new settings.

Note: Settings changes overwrite existing settings, they do not append. For example, if you're adding a txid, send the complete list of txids, not just the new addition.
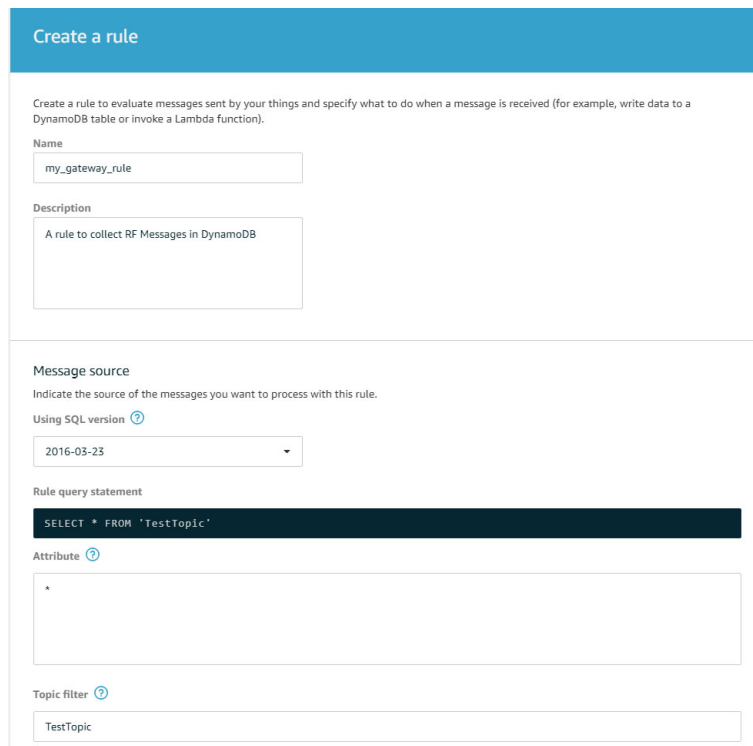

Note: Minimize topic changes as they can interrupt connectivity with the cloud and create a large queue on the IP gateway.

Note: Sending only "{}" to the device will return the current settings without any modification.

Store Message Data

This section will show a basic use case to store message data in DynamoDB. This is only an example of a possible next step. If you're familiar with using IoT devices in AWS, this section can be skipped.

1. Select the "Act" tab on the AWS IoT dashboard and click "Create" to start a new rule.

2. Configure settings as desired. Following are the minimum settings required.

Figure 8 Configure rule settings.

3. Navigate to "Set one or more actions."

Figure 9 Add an action.

4. Select "Insert a message into a DynamoDB table."

5. Set a Hash key value.

6. Select a role and update it.

7. Click "Add action."After that, you can create the rule.

8. Check the AWS "Act" tab and ensure your rule is listed as enabled.

Inovonics Transmitter and Repeater Message Examples

This section contains information that will be helpful when creating rules for processing incoming data.

All messages function like a list of dictionaries in Python. They will come wrapped in square brackets, contain at least one message, and will be separated by commas.

Note: Currently some historical messages will have a TXID of 0. These do not contain any data from the sensor and can be ignored on the receiving end.

Transmitter RF Message

```
[{
"message": {'
        "header": "InboundComplete",
        "signal": {
"margin": 38,
                    "level": 38
                },
            "metadata": {
            "traceCount": 0,
                    "originatorUid": {
                    "serialNumber": 16249316,
                        "market": {
                        "name": "Submetering"
                        }
                },
                    "firstHopUid": {
                    "serialNumber": 10894274,
                        "market": {
                        "name": "NetworkCoordinator"
                        }
                },
                    "traceUidList": [],
                    "hopCount": 0
            },
            "payload": {
            "leakDetectCount": 100,
                "mcb": 61,
                    "status": {
                    "stat1bit4": false,
                        "tamper": true,
                        "stat1bit0": false,
                        "stat1bit1": false,
                        "stat0bit1": false,
                        "stat0bit0": false,
                        "rapidTransmissionMode": false,
                        "stat1bit5": false,
                        "stat0bit7": false,
                        "lowBattery": true,
                        "shipping": false,
                        "deltaTotalizer": false,
                        "supervision": true,
                        "stat1bit3": false,
                        "stat1bit2": false,
                        "reset": false
                },
```

```
                    "count": 0,
                    "name": "SubmeteringApplicationPayload",
                    "pti": 1
              }
          },
          "market": "Submetering",
          "had_exception": false,
          "network-type": "echostream",
          "timestamp": "2018-07-11T19:14:50.065485",
      "txid": 10377745,
          "app_data_submetering": {
          "count_changed": false,
                "rapid_message": false
          }
      }]
```

Repeater RF Message

```
[{
  'app_data_submetering': {
      'count_changed': False,
      'rapid_message': False
  },
  'message': {
      'metadata': {
          'traceUidList': [],
          'hopCount': 0,
          'firstHopUid': {
              'market': {
                  'name': 'NetworkCoordinator'
              },
              'serialNumber': 10894274
          },
          'originatorUid': {
              'market': {
                  'name': 'Repeater'
              },
              'serialNumber': 8086348
          },
          'traceCount': 0
      },
      'header': 'InboundComplete',
      'signal': {
          'level': 15,
          'margin': 15
      },
      'payload': {
          'name': 'RepeaterApplicationPayload',
          'repeaterPayload': {
              'name': 'RepeaterStatusWithBroadcastRepeater',
              'pti': 0,
              'status': {
                  'tamper': True,
                  'noChange': True,
                  'neighborListAssigned': False,
                  'limitedScanningRemotes': False,
                  'lowBattery': False,
                  'onlyOneNeighbor': False,
                  'jammed': False,
                  'stat0bit0': False,
                  'layer': 0,
```

```
                                'stat0bit2': False,
                                'lossOfLinePower': False,
                                'reset': False,
                                'statusAck': False
                            }
                        },
                        'mcb': 65
                    }
                },
                'network-type': 'echostream',
                'market': 'Repeater',
                'txid': 8086348,
                'timestamp': '2018-06-26T14:54:59.190750'
        }]
```