

目次

- AWS IoTを使用する** 3
- Fluentd用outputプラグイン** 3
- 事前の準備 3
- プラグインの設定 3
- 動作確認例 4
- Device Shadow ライブラリ** 7
- 事前の準備 7
- API一覧 8
- サンプルアプリケーション 9

AWS IoTを使用する

MA-E3xx for IoTファームウェア(v2.6.10rc2以降)には、[AWS IoT](#)の機能を使用するためのソフトウェアが内蔵されています。

以下の2つのソフトウェアからなります。

* Fluentd用outputプラグイン: Fluentdのログ出力をAWS IoTに出力するためのプラグイン * Device Shadowライブラリ: Device Shadowを使用するためのAPI

Fluentd用outputプラグイン

Fluentd用outputプラグインは、Fluentdのログ出力を、AWS IoTのメッセージとして出力します。

プラグインは、`/usr/lib/ruby/vendor_ruby/fluent/plugin` に `out_awsiot.rb` というファイル名で置かれています。

事前の準備

デバイス(MA-E3xx)用の「[秘密鍵 \(private key\)](#)」[証明書 \(certificate\)](#)を作成する必要があります。

これらを作成するには、[AWSのWebコンソールから作成する方法](#)や、AWSのCLIツールで作成する方法があります。

また、ルート証明書を[Symantecのサイト](#)からダウンロードします。

作成 取得したファイルを、任意のディレクトリに以下のファイル名で置いてください。

* 秘密鍵: `privkey.pem` * 証明書: `cert.pem` * ルート証明書: `aws-iot-rootCA.pem`

プラグインの設定

プラグインの動作の設定は、Fluentdの設定ファイル(`/etc/fluent/fluent.conf`)で行います。

[fluent.conf](#)

```
<match *>
  @type awsiot
  certs_dir /home/user1/certs
  host ABCDEFGHIJKLM. iot.ap-northeast-1.amazonaws.com
  mqtt_qos 1
  client_id hogeclient
</match>
```

項目	内容	Notes
@type	プラグイン使用宣言	awsiot固定。必須。
certs_dir	証明書 秘密鍵のディレクトリ	必須
host	MQTTブローカーのホスト名	AWS IoTのWebコンソールからThingを選択し、Detailの「REST API Endpoint」のホスト名部分。必須。
mqtt_qos	MQTTのQoSレベル	0 or 1省略時1。
client_id	クライアントID	同じ秘密鍵 かつ同じクライアントIDで複数の同時接続はできない。省略時centuryfluent

またBufferedOutput pluginで用いられる設定も指定できます。^{1) 2)}

動作確認例

ここではdummyプラグインを使ってダミーデータを生成し、ダミーデータを標準出力に出力しつつ、10秒に1度ダミーデータをAWS IoTに出力する設定例を示します。また、出力されたメッセージをAWS IoT Webコンソールで確認する方法を示します。

まず、設定ファイルを作成します。

sample.conf

```
<source>
  @type dummy
  tag dummy
  auto_increment_key foo_key
</source>

<match dummy>
  @type copy
  <store>
    @type stdout
  </store>
  <store>
    @type awsiot
    certs_dir /home/user1/certs
    host ABCDEFGHIJKLM.iot.ap-northeast-1.amazonaws.com
    mqtt_qos 1
    client_id hogeclient
    flush_interval 10s
    log_level debug
  </store>
</match>
```

次に、AWS IoTのWebコンソールから、以下の操作を順に行います。

+ MQTT Clientをクリックする + Generate client ID Connectを順にクリックする + Subscribe to topicをクリックする + Subscription topicに「#」(全てのtopic)を入力し、Subscribeをクリックする。

先ほど作成した設定ファイルを使ってFluentdを起動します。

```
user1@plum:~$ fluentd -c test.conf
2016-10-18 14:59:20 +0900 [info]: reading config file path="test.conf"
2016-10-18 14:59:20 +0900 [info]: starting fluentd-0.12.17
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-mixin-config-placeholders'
version '0.3.0'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-mixin-rewrite-tag-name'
version '0.1.0'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-dstat' version '0.3.1'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-filter-record-map'
version '0.1.4'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-filter_typecast'
version '0.0.2'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-flatten-hash' version
'0.2.0'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-forest' version '0.3.0'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-influxdb' version
'0.2.2'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-mqtt-io' version
'0.0.4'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-named_pipe' version
'0.1.1'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-notifier' version
'0.2.4'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-record-modifier'
version '0.3.0'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-record-reformer'
version '0.7.0'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-rewrite-tag-filter'
version '1.5.1'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-secure-forward' version
'0.3.3dev2'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-stats' version '0.3.6'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-td' version '0.10.28'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-typecast' version
'0.2.0'
2016-10-18 14:59:22 +0900 [info]: gem 'fluent-plugin-udp-stream' version
'0.0.2'
2016-10-18 14:59:22 +0900 [info]: gem 'fluentd' version '0.12.17'
2016-10-18 14:59:22 +0900 [info]: adding match pattern="dummy" type="copy"
2016-10-18 14:59:22 +0900 [info]: adding source type="dummy"
2016-10-18 14:59:22 +0900 [info]: using configuration file: <ROOT>
  <source>
    @type dummy
    tag dummy
    auto_increment_key foo_key
  </source>
  <match dummy>
    @type copy
    <store>
```

```
@type stdout
</store>
<store>
  @type awsiot
  certs_dir /home/user1/certs
  host ABCDEFGHIJKLM.iot.ap-northeast-1.amazonaws.com
  mqtt_qos 1
  client_id hogeclient
  flush_interval 10s
  log_level debug
</store>
</match>
</ROOT>
2016-10-18 14:59:23 +0900 dummy: {"message":"dummy","foo_key":0}
2016-10-18 14:59:24 +0900 dummy: {"message":"dummy","foo_key":1}
2016-10-18 14:59:25 +0900 dummy: {"message":"dummy","foo_key":2}
2016-10-18 14:59:26 +0900 dummy: {"message":"dummy","foo_key":3}
2016-10-18 14:59:27 +0900 dummy: {"message":"dummy","foo_key":4}
2016-10-18 14:59:28 +0900 dummy: {"message":"dummy","foo_key":5}
2016-10-18 14:59:29 +0900 dummy: {"message":"dummy","foo_key":6}
2016-10-18 14:59:30 +0900 dummy: {"message":"dummy","foo_key":7}
2016-10-18 14:59:31 +0900 dummy: {"message":"dummy","foo_key":8}
2016-10-18 14:59:32 +0900 dummy: {"message":"dummy","foo_key":9}
2016-10-18 14:59:33 +0900 [debug]: begin reconnect.
2016-10-18 14:59:33 +0900 dummy: {"message":"dummy","foo_key":10}
2016-10-18 14:59:33 +0900 [debug]: reconnect finished.
2016-10-18 14:59:34 +0900 dummy: {"message":"dummy","foo_key":11}
```

AWS IoT Webコンソールを見ると、10秒ごとに10個のメッセージが送られてきています。

The screenshot shows the AWS IoT console interface. The top navigation bar includes the AWS logo, 'AWS IoT' title, and links for Resources, MQTT Client, Tutorial, Settings, and 0 notifications. The main content area is split into two panes. The left pane, titled 'MQTT Client Actions', shows a list of messages received from a device. Each message is a JSON object with 'message' and 'foo_key' fields. The right pane shows the configuration for the MQTT client, including a 'Subscribe' button, 'Subscription topic' field, 'Max message capture' set to 100, and 'Quality of service (QoS)' set to 1.

Clear all messages

x

Clear messages

dummy Oct 18, 2016 2:39:24 PM

```
1 {
2   "message": "dummy",
3   "foo_key": 9
4 }
```

dummy Oct 18, 2016 2:39:24 PM

```
1 {
2   "message": "dummy",
3   "foo_key": 8
4 }
```

dummy Oct 18, 2016 2:39:24 PM

```
1 {
2   "message": "dummy",
3   "foo_key": 7
4 }
```

dummy Oct 18, 2016 2:39:23 PM

```
1 {
2   "message": "dummy",
3   "foo_key": 6
4 }
```

Device Gateway connection ✓

Subscribe to topic 📄

Publish to topic 📤

Publish log 🔄

Subscription topic

Max message capture

100

Quality of service (QoS) 0 1

Subscribe

フィードバック 日本語 © 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. プライバシーポリシー 利用規約

Device Shadowライブラリ

Device Shadowライブラリは、[Device Shadow](#)を簡単に使うことができるRubyのライブラリです。

ライブラリは、`/usr/lib/ruby/vendor_ruby` に `awsiot_shadow.rb` というファイル名で置かれています。

事前の準備

Fluentd用outputプラグインと同じ要領で、証明書 秘密鍵とルート証明書を作成 取得してください。

API一覧

Centurysys::AWSIoTShadowクラスに様々なAPIが用意されています。

new(hash)

AWSIoTShadowクラスのオブジェクトを生成します。引数のhashには以下の要素を指定できます。

key	value	note
host	MQTTブローカーのホスト名	必須
port	MQTTブローカーのポート番号	省略時 8883
ca_file	ルート証明書のファイル名	必須
key_file	秘密鍵のファイル名	必須
cert_file	証明書のファイル名	必須
thing_name	Device ShadowのThing名	必須
client_id	クライアントID	実際にはこの値に「R□□G□□U□」を後置した文字列が使われる
keep_alive	MQTT Keep-Aliveを送る秒数	省略時 0(送らない)
enable_auto_reconnect	trueならMQTTブローカーとの接続が切れたとき再接続する	省略時 false(再接続しない)

connect()

AWS IoTのMQTTブローカーに接続します。yield()を呼び出す前には必ず呼び出して下さい。

メソッド内部で、2秒のsleep()が発生します。これはMQTTブローカーにShadowの更新を知らせるTopicをSubscribeしてから、実際にそのTopicが受信できるまでの間に、2秒間必要なためです。

オブジェクト生成時にkeep_aliveを有効にしていた場合で、MQTTブローカーと切断された時には、このメソッドを呼び出したスレッドに例外(MQTT::ProtocolException)が発生します。

yield() → hash

MQTTブローカーからShadowの更新(/update/delta)を受信し、更新された部分をHashとして返します。

Shadowが更新されなければ永遠に受信待ちを行います。タイムアウト処理を行いたい場合は、Timeoutモジュールを使って呼び出して下さい。

disconnect()

MQTTブローカーとの接続を切断します。

get() → hash

AWS IoTからShadowの現在の状態(reported)を取得します。デバイスの初期状態をAWS IoTから取得して、デバイスを初期設定する場合に使用します。

状態を取得できるまで、永遠に送信処理を行います。タイムアウト処理を行いたい場合は、Timeoutモジュールを使って呼び出して下さい。

update(hash)

AWS IoTに対してShadowの状態(reported)を更新する処理を行います。

メソッド内部で、2秒のsleep()が発生する場合があります。これはupdateするためのMQTTクライアントを接続するときにupdateの結果を取得するために、新たなTopicをSubscribeする必要があるためです。

状態を更新できるまで、永遠に送信処理を行います。タイムアウト処理を行いたい場合は、Timeoutモジュールを使って呼び出して下さい。

サンプルアプリケーション

サンプルアプリケーションのコードを以下に示します。

ここではTestThingというThingのShadowを監視し、他からShadowが更新されたら、更新された状態を標準出力に表示し、MAに繋がったデバイスを制御したとみなして、制御後の状態でShadowを更新します。

[shadow_sample.ruby](#)

```
require 'awsiot_shadow'

@_host = 'ABCDEF.iot.ap-northeast-1.amazonaws.com'
_certdir = '/home/user1/certs'
@_ca_file = _certdir + '/aws-iot-rootCA.pem'
@_key_file = _certdir + '/privkey.pem'
@_cert_file = _certdir + '/cert.pem'

shadow_client = Centurysys::AWSIoTShadow.new(
  :host => @_host,
  :ca_file => @_ca_file,
  :key_file => @_key_file,
  :cert_file => @_cert_file,
  :thing_name => 'TestThing',
  :client_id => 'test',
  :keep_alive => 15
)

# AWS IoTのMQTTブローカーに接続する。
shadow_client.connect()

now_state = nil

# AWS IoTから、デバイスに設定すべきShadowの初期状態を取得する。
timeout(5) {
  now_state = shadow_client.get
}

# 実際にはここでnow_stateに従ってデバイスの状態を変える

# AWS IoT上にあるShadowの状態を、デバイスに設定した状態に更新する。
timeout(5) {
```

```
shadow_client.update(now_state)
}

p "start main loop"

while true
  delta = nil
  begin
    # AWS IoTからShadowの更新を取得する。(ここでは5秒間でtimeoutするようにして
    いる)
    timeout(5) {
      delta = shadow_client.yield
    }
    p delta

    # デバイスの状態(now_state)をdeltaに従って変える。
    delta.each_key { |key|
      if (now_state[key] != delta[key]) then
        # changing device state to delta(desired) state
        now_state[key] = delta[key]
      end
    }

    # AWS IoTのShadowの状態を、デバイスに設定した状態に更新する。
    timeout(5) {
      shadow_client.update(now_state)
    }
  rescue Timeout::Error
    # タイムアウトした(ここではそのまま次のループへ)
    p "Timeout occured. wait again"
  end
end
```

AWS IoTのWebコンソールから「TestThing」の「Update shadow」を選択し、「desired」以下にあるパラメータを書き換えると、書き換えられたパラメータが標準出力に表示された後、そのパラメータがAWS IoT Webコンソールの「reported」以下に反映されます。

1)

[Output Plugin Overview](#)

2)

[BufferedOutput pluginの代表的なoptionについて](#)

From:

<https://ma-tech.centurysys.jp/> - MA-X/MA-S/MA-E/IP-K Developers' Wiki

Permanent link:

https://ma-tech.centurysys.jp/doku.php?id=mae3xx_ope:awsiot:start

Last update: **2016/10/19 18:48**

