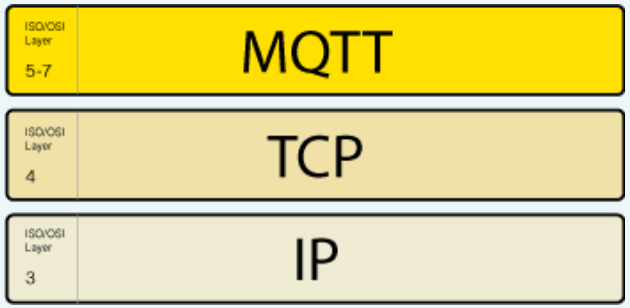
Internet of Things
class 8

MQTT, AWS-IoT

MQTT Protocol

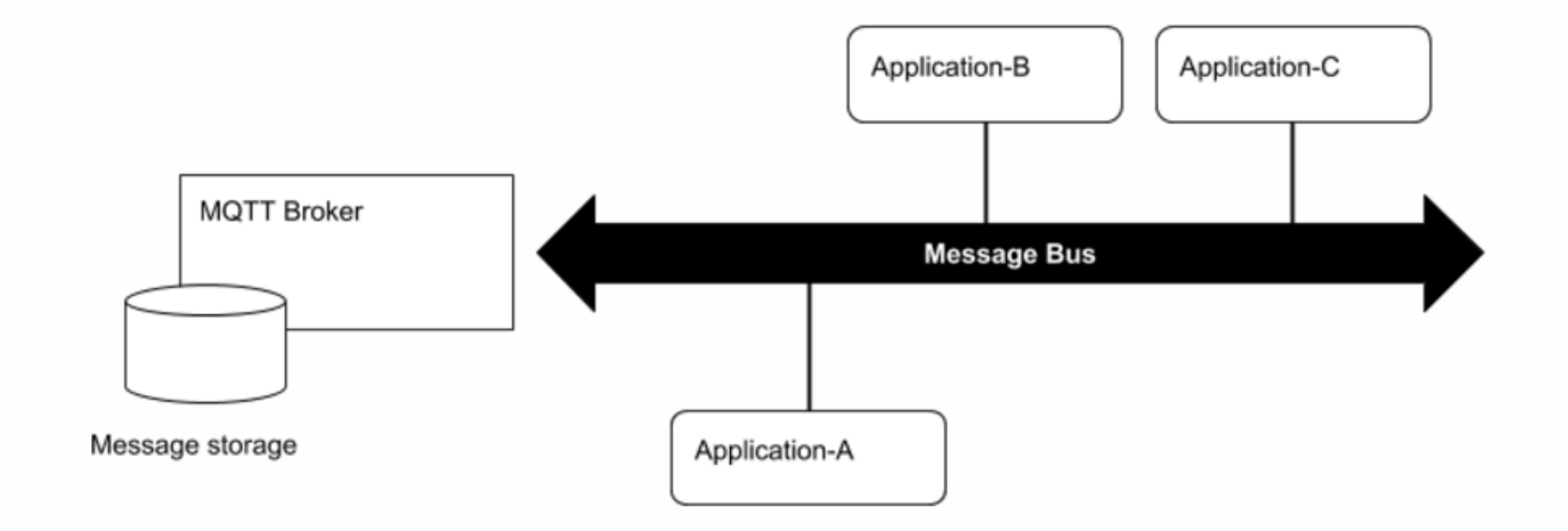
MQTT Protocol

- Message Queue Telemetry Transport (MQTT)
  - ***publish/subscribe*** messaging protocol designed for lightweight M2M communications
  - Server/Client Model over TCP
    - Client: sensors, Server: broker
  - MQTT Stack
    - The MQTT protocol is based on top of TCP/IP and both client and broker need to have a TCP/IP stack



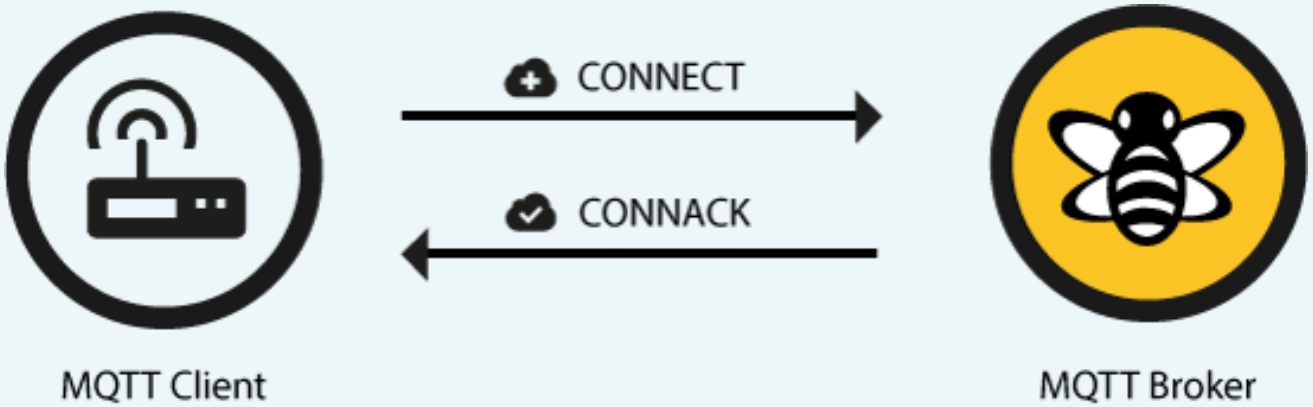
MQTT

- Message oriented Usage
  - Every MSG is a discrete chunk of data
  - Every message is published to an address (called topic)
  - Clients may subscribe to multiple topics
  - Every client subscribed to a topic receives every message published to the topic



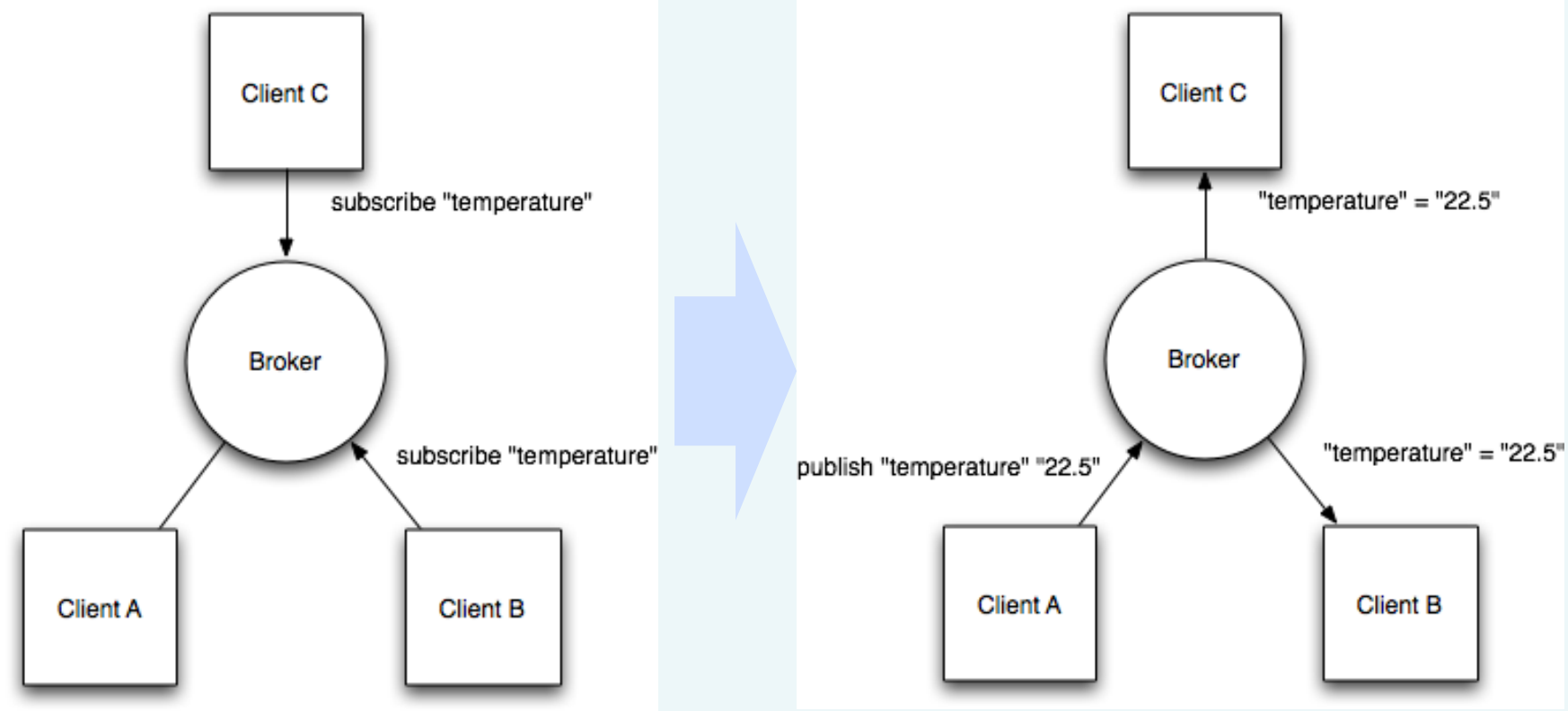
MQTT

- Connection Flow
  - The MQTT connection itself is always between one client and the broker
    - No client is connected to another client directly
  - The connection is initiated through a client sending a ***CONNECT*** message to the broker
  - The broker response with a ***CONNACK*** and a status code



MQTT

- Scenario
  - Simple network with 3-clients and 1-broker



MQTT

- Scenario
  - Simple network with 3-clients and 1-broker
    - All 3-clients open TCP connections with the broker
    - Clients B and C subscribe to the topic "temperature"
    - Client A published a value of '22.5' for topic "temperature"
    - The broker forwards the message to all subscribed clients
    - The publisher subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one

MQTT

- MQTT Features
  - Topic matching
  - Application level QoS
  - Last will and Testament
  - Persistence
  - Security
  - MQTT-SN

MQTT

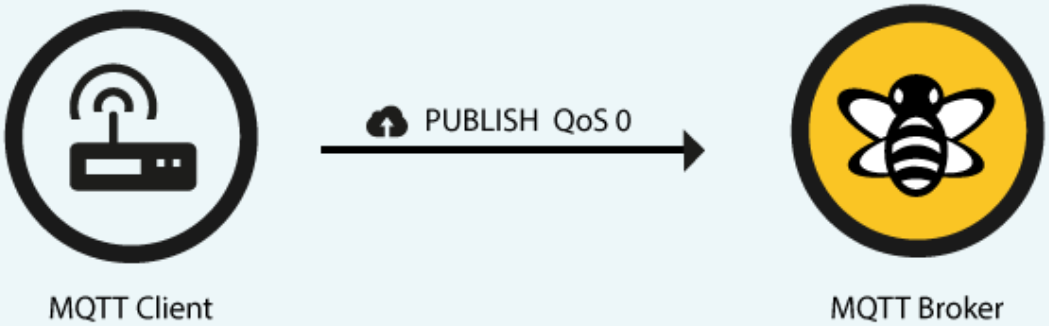
- Topic Matching
  - Hierarchical Topics
    - Wildcards('+', '#') are allowed when registering a subscription (but not when publishing)
      - » allowing whole hierarchies to be observed by clients
      - » '+' matches any single directory name
        - kitchen/+/temperature ➔ kitchen/foo/temperature
      - » '#' matches any number of directories of any name
        - kitchen/# ➔ kitchen/fridge/compressor/valve1/temperature

MQTT

- MQTT QoS
  - Application level QoS
    - MQTT supports three quality of service levels
      - » "At most once" – QoS(0)
      - » "At least once" – QoS(1)
      - » "Exactly once" – QoS(2)

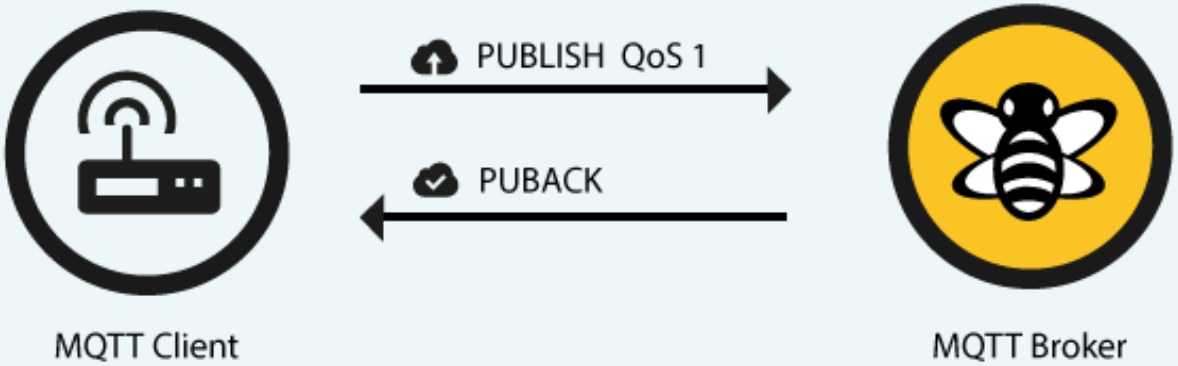
MQTT

- MQTT QoS
  - Application level QoS
    - "At most once" → "Fire and Forget"
      - » Where messages are delivered according to the best efforts of the underlying TCP/IP network
      - » Message loss can occur
      - » This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after



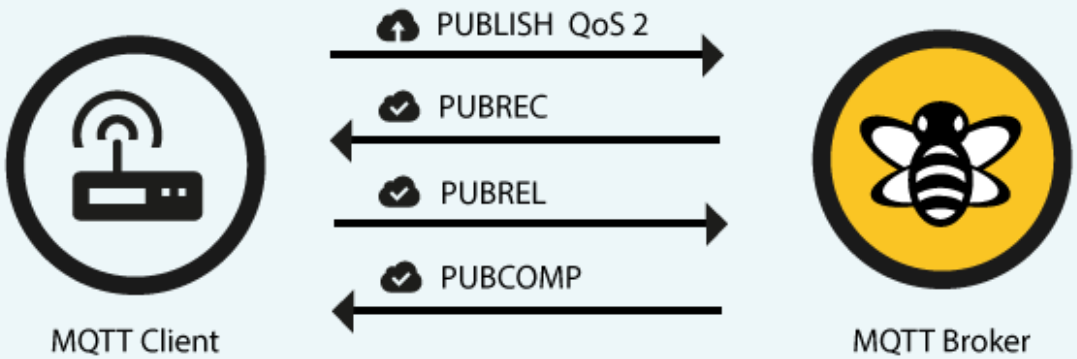
MQTT

- MQTT QoS
  - Application level QoS
    - "At least once" → Acknowledged Delivery
      - » Guaranteed that a message will be delivered ***at least once*** to the receiver
      - » But the message can also be delivered more than once
      - » The sender will store the message until it gets an acknowledgement in form of a ***PUBACK*** command message from the receiver



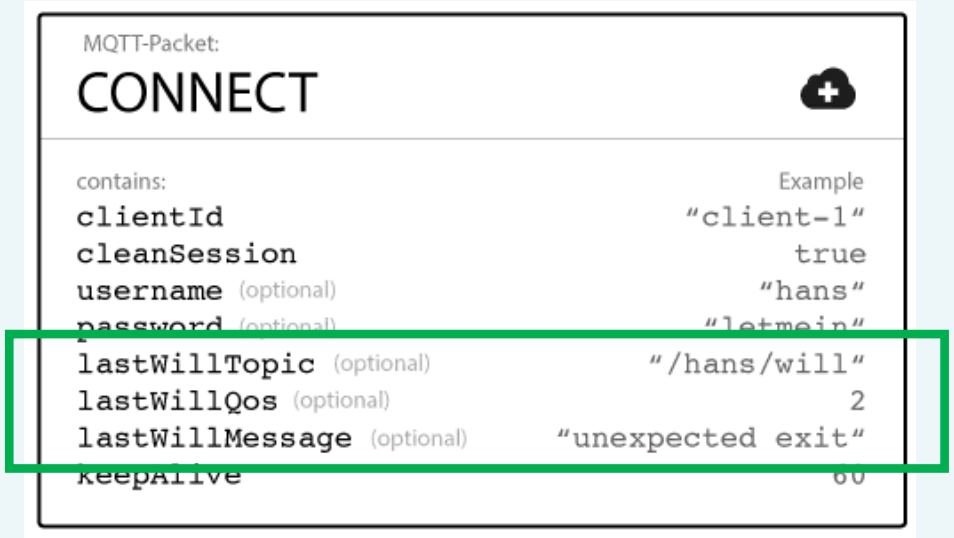
MQTT

- MQTT QoS
  - Application level QoS
    - "Exactly once" → Assured Delivery
      - » Where message are assured to arrive ***exactly once***
      - » It is the safest and also the slowest quality of service level
      - » The guarantee is provided by two flows there and back between sender and receiver
      - » This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied



MQTT

- MQTT Last will and Testament
  - Notify other clients about an ungracefully disconnected client
  - MQTT clients can register a custom "last will and testament" message to be sent by the broker if they disconnect
  - These messages can be used to signal to subscribers when a device disconnects.



MQTT

- Persistence
  - MQTT has support for persistent messages stored on the broker
  - When publishing messages, clients may request that the broker persists the message
  - Only the most recent persistent message is stored
  - When a client subscribes to a topic, any persisted message will be sent to the client
  - Unlike a message queue, MQTT brokers do not allow persisted messages to back up inside the server

MQTT

- Security
  - MQTT brokers may require username and password authentication from clients to connect
  - To ensure privacy, the TCP connection may be encrypted with SSL/TLS

MQTT

- MQTT-SN
  - MQTT: Two drawbacks for very constrained devices
    - Every MQTT client must support TCP and will typically hold a connection open to the broker at all times
      - » For some environments where packet loss is high or computing resources are scarce
    - MQTT topic names are often long strings which make them impractical for 802.15.4
    - Both of these shortcomings are addressed by the MQTT-SN protocol
      - » Defines a UDP mapping of MQTT and adds broker support for indexing topic names

MQTT

- MQTT Message Format
  - The message header for each MQTT command message contains a fixed header
  - Some messages also require a variable header and a payload
  - Fixed Header

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

  - Variable Header

bit	7	6	5	4	3	2	1	0
	Protocol Version							
	0	0	0	0	0	0	1	1

MQTT

- MQTT Message Type

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

MQTT

- MQTT CONNECT
  - CONNECT Packet Example
    - Remaining Length is the length of the variable header (10 bytes) plus the length of the Payload

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (1)				Reserved			
	0	0	0	1	0	0	0	0
byte 2…	Remaining Length							

MQTT

- MQTT CONNECT
  - Variable Header for CONNECT Packet
    - Consists of four fields
      - » Protocol Name
      - » Protocol Level
      - » Connect Flags
      - » Keep Alive

	Description	7	6	5	4	3	2	1	0
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (4)	0	0	0	0	0	1	0	0
byte 3	'M'	0	1	0	0	1	1	0	1
byte 4	'Q'	0	1	0	1	0	0	0	1
byte 5	'T'	0	1	0	1	0	1	0	0
byte 6	'T'	0	1	0	1	0	1	0	0
Protocol Level									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 7	Level (4)	0	0	0	0	0	1	0	0
Connect Flags									
byte 8	User Name Flag (1) Password Flag (1) Will Retain (0) Will QoS (01) Will Flag (1) Clean Session (1) *Reserved* (0)	1	1	0	0	1	1	1	0
Keep Alive									
byte 9	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 10	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

MQTT

- MQTT Message Format
  - CONNACT Packet

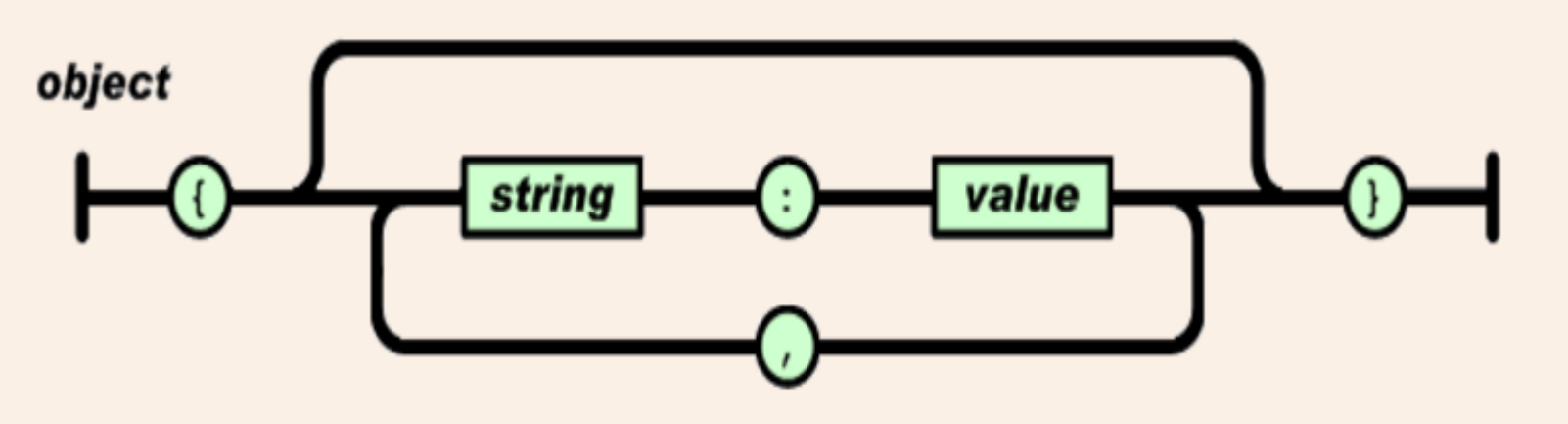
Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet Type (2)				Reserved			
	0	0	1	0	0	0	0	0
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

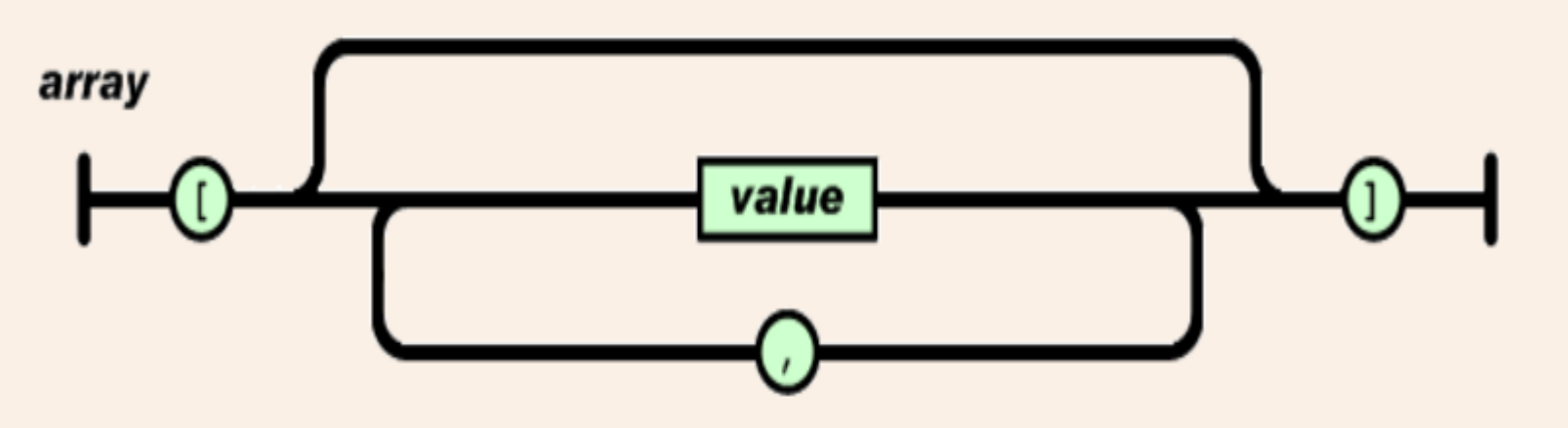
	Description	7	6	5	4	3	2	1	0
Connect Acknowledge Flags		Reserved							SP[1]
byte 1		0	0	0	0	0	0	0	X
Connect Return code									
byte 2		X	X	X	X	X	X	X	X

Value	Return Code Response	Description
0	0x00 Connection Accepted	Connection accepted
1	0x01 Connection Refused, unacceptable protocol version	The Server does not support the level of the MQTT protocol requested by the Client
2	0x02 Connection Refused, identifier rejected	The Client identifier is correct UTF-8 but not allowed by the Server
3	0x03 Connection Refused, Server unavailable	The Network Connection has been made but the MQTT service is unavailable
4	0x04 Connection Refused, bad user name or password	The data in the user name or password is malformed
5	0x05 Connection Refused, not authorized	The Client is not authorized to connect
6-255		Reserved for future use

JSON

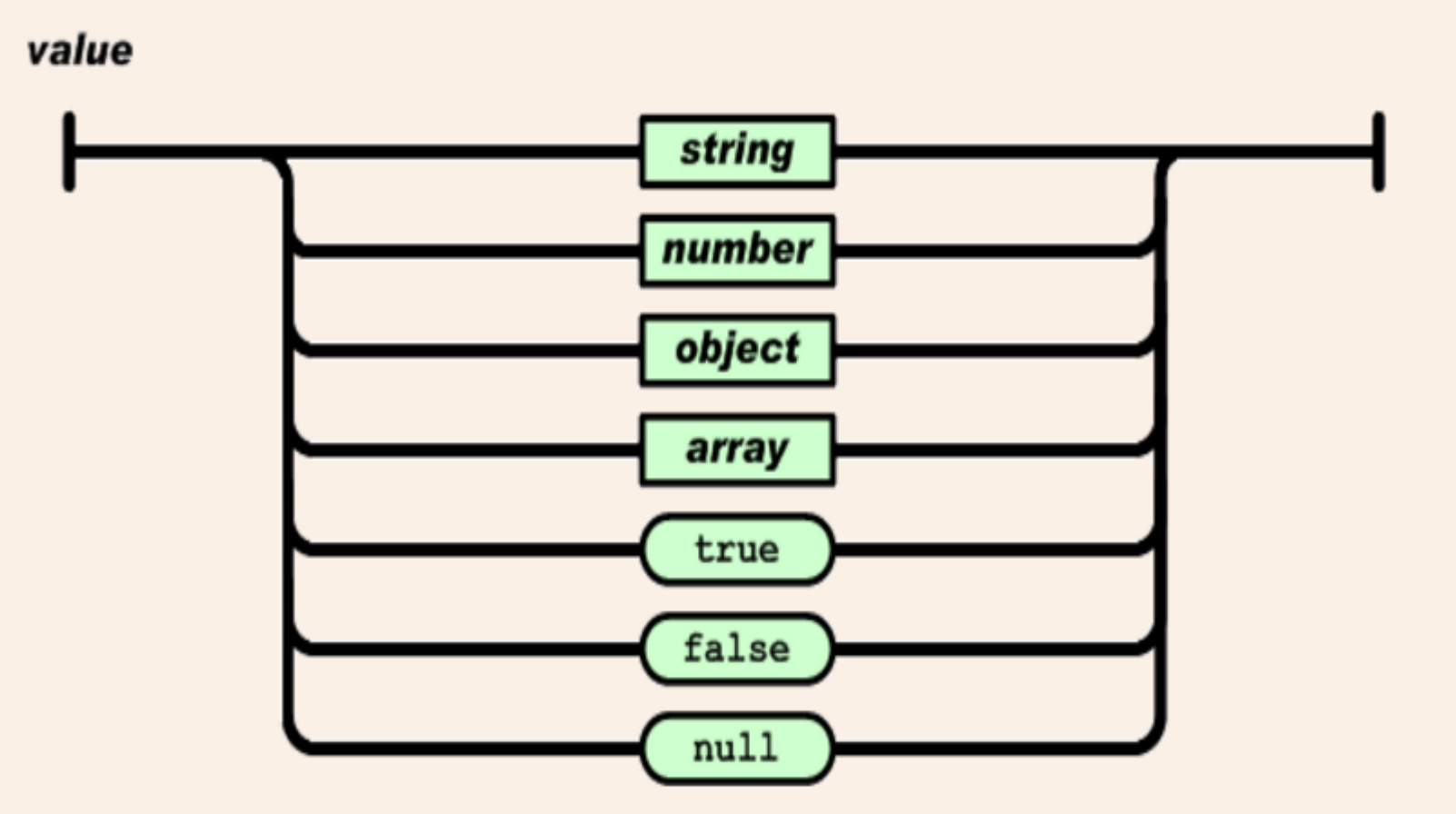
- JSON: JavaScript Object Notation





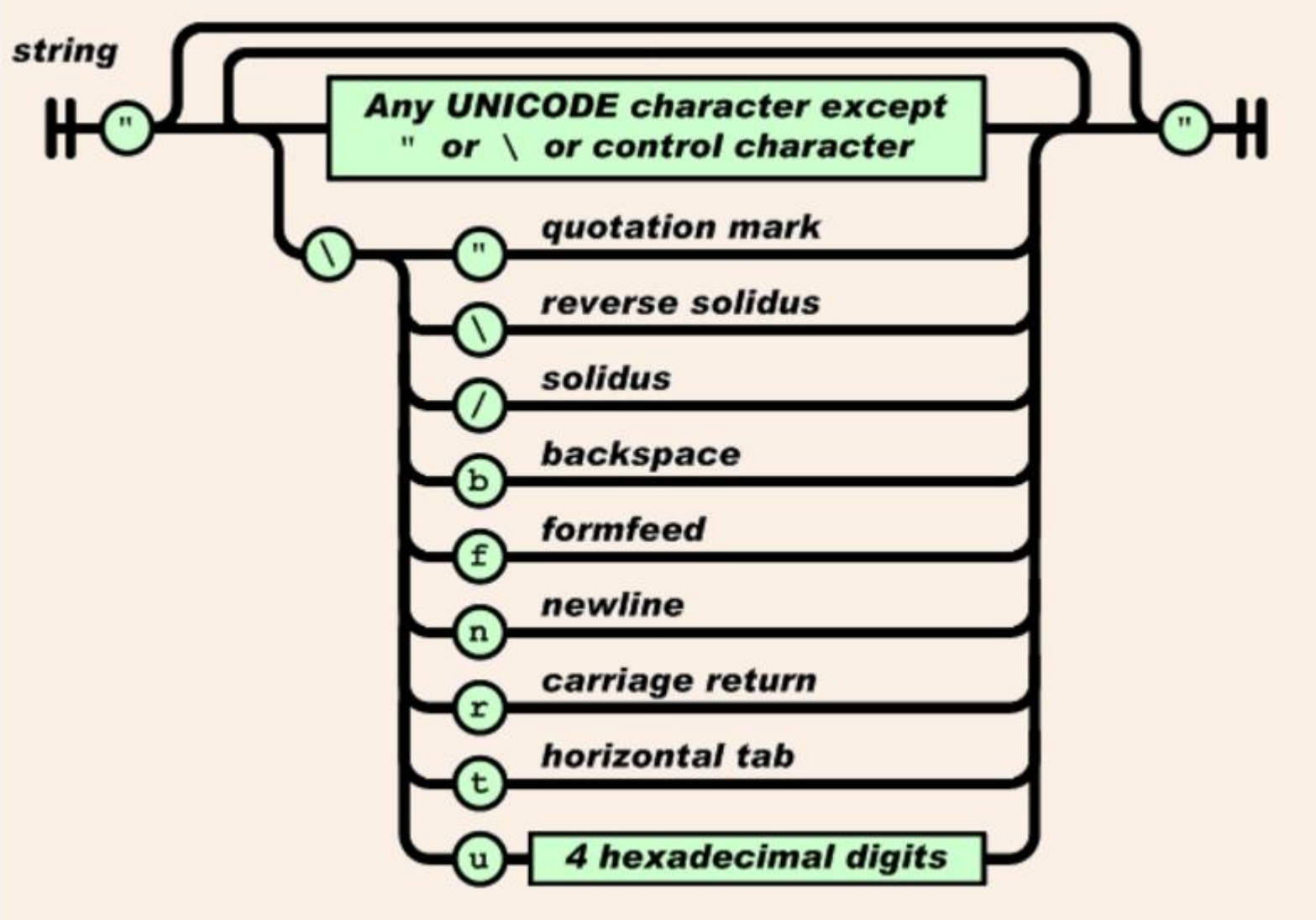
JSON

- JSON: JavaScript Object Notation



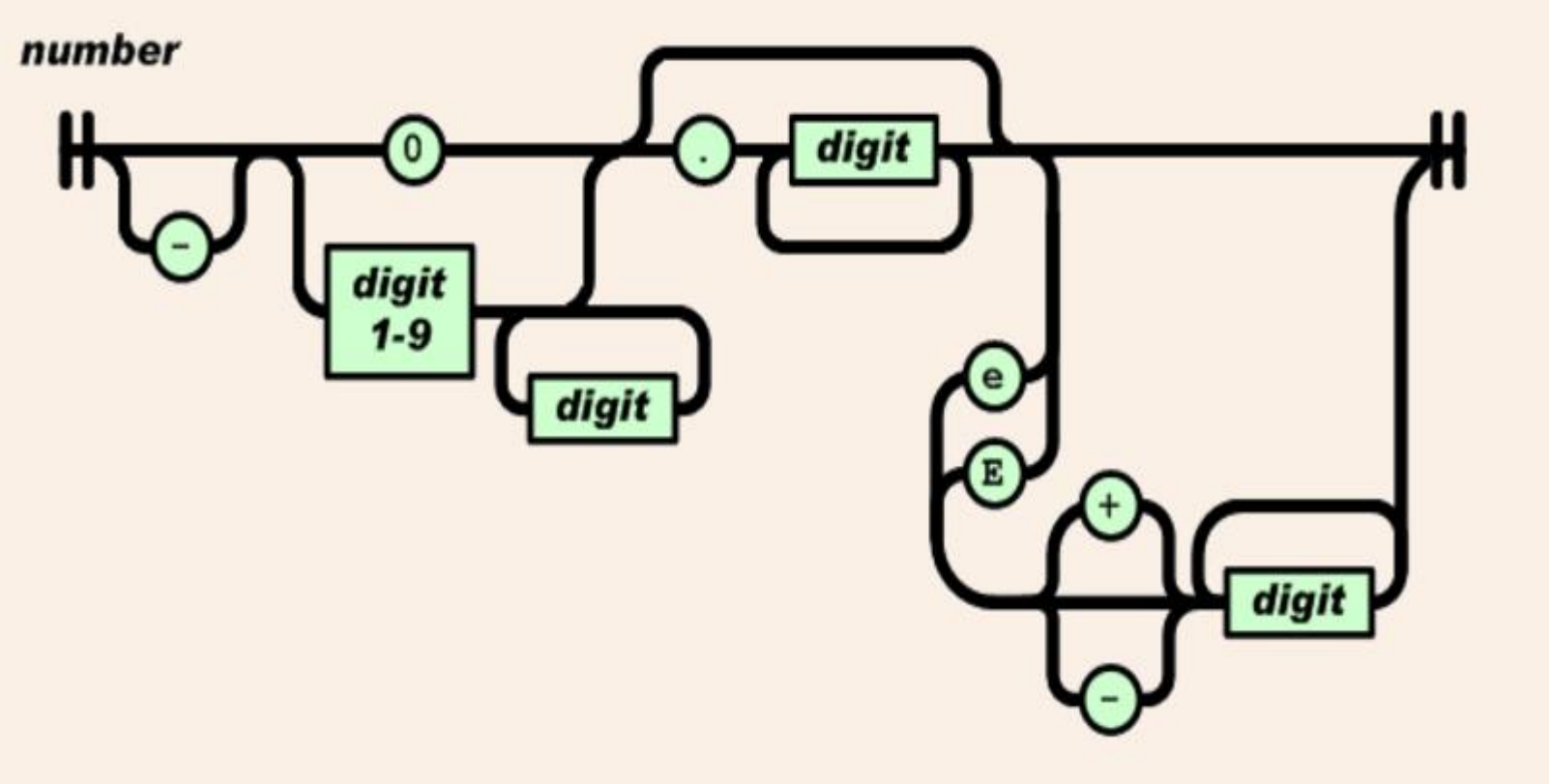
JSON

- JSON: JavaScript Object Notation



JSON

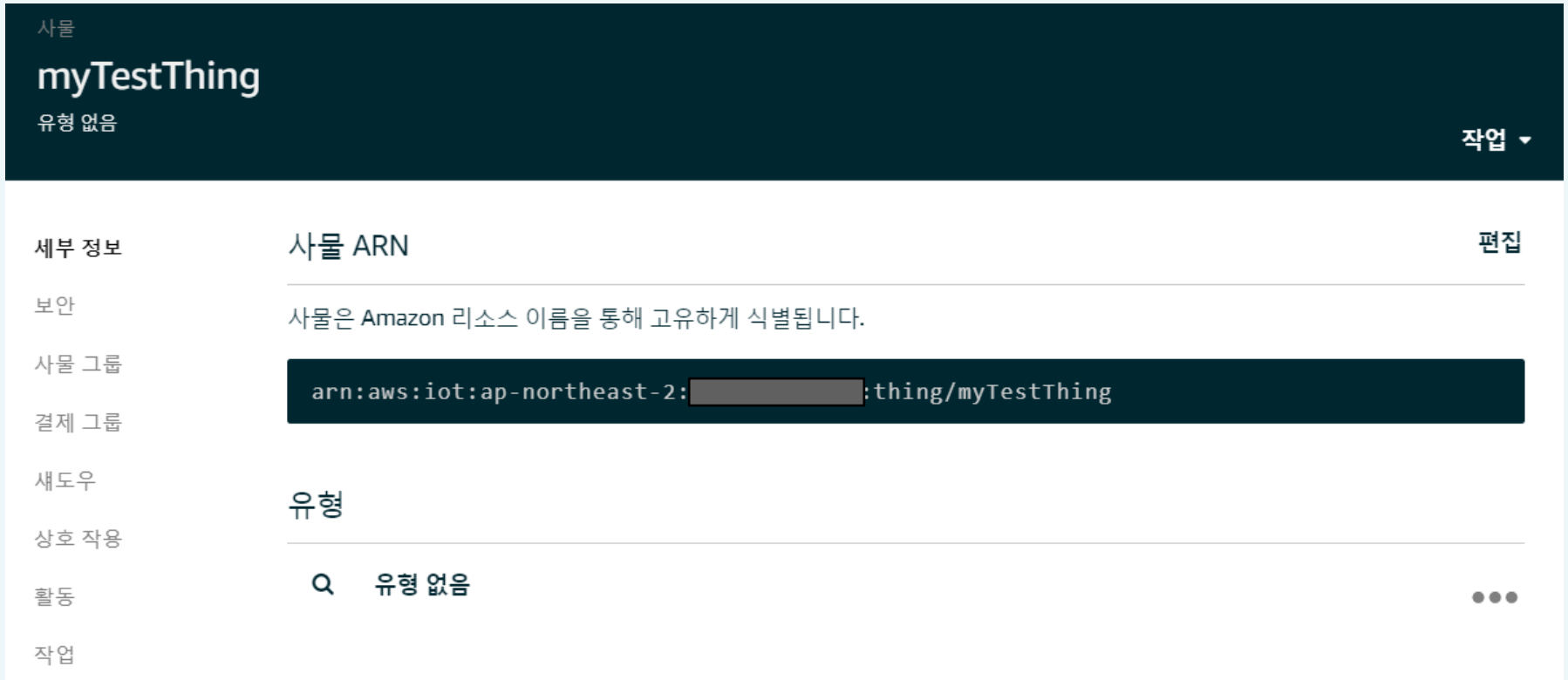
- JSON: JavaScript Object Notation



AWS IoT

1. AWS Account (Free-tier or Educate Program)
2. Working Region 선택: Seoul
3. Create Things.. (Shadow)
4. Using MQTT.. Publish, Subscribe Topics
5. Rule-Engine
6. Testing..
7. Using MQTT-fx
8. Implement and Connect Devices

Register Thing, 보안



Register Thing, 보안

인증서 생성 완료!

이들 파일을 다운로드하여 안전한 장소에 저장하십시오. 인증서는 언제든지 검색할 수 있지만 프라이빗 키와 퍼블릭 키는 이 페이지를 닫은 후에 검색할 수 없습니다.

디바이스에 연결하려면 다음을 다운로드해야 합니다.

이 사물에 대한 인증서	65b2889b7e.cert.pem	다운로드
퍼블릭 키	65b2889b7e.public.key	다운로드
프라이빗 키	65b2889b7e.private.key	다운로드

AWS IoT의 루트 CA도 다운로드해야 합니다.
AWS IoT의 루트 CA 다운로드

활성화

사물
myTestThing
유형 없음

세부 정보
보안
사물 그룹
결제 그룹
섀도우
상호 작용

인증서

인증서 생성 | 기타 옵션 보기

0af70056543849ca8f...

Register Thing, 보안

CA certificates for server authentication

Depending on which type of data endpoint you are using and which cipher suite you have negotiated, AWS IoT Core server authentication certificates are signed by one of the following root CA certificates:

VeriSign Endpoints (legacy)

- RSA 2048 bit key: VeriSign Class 3 Public Primary G5 root CA certificate

Amazon Trust Services Endpoints (preferred)

> **Note**
>
> You might need to right click these links and select **Save link as...** to save these certificates as files.

- RSA 2048 bit key: Amazon Root CA 1.
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.
- ECC 256 bit key: Amazon Root CA 3.

Register Thing - Policy

```
arn:aws:iot:ap-northeast-2:[redacted]:policy/myTestThing_Policy
```

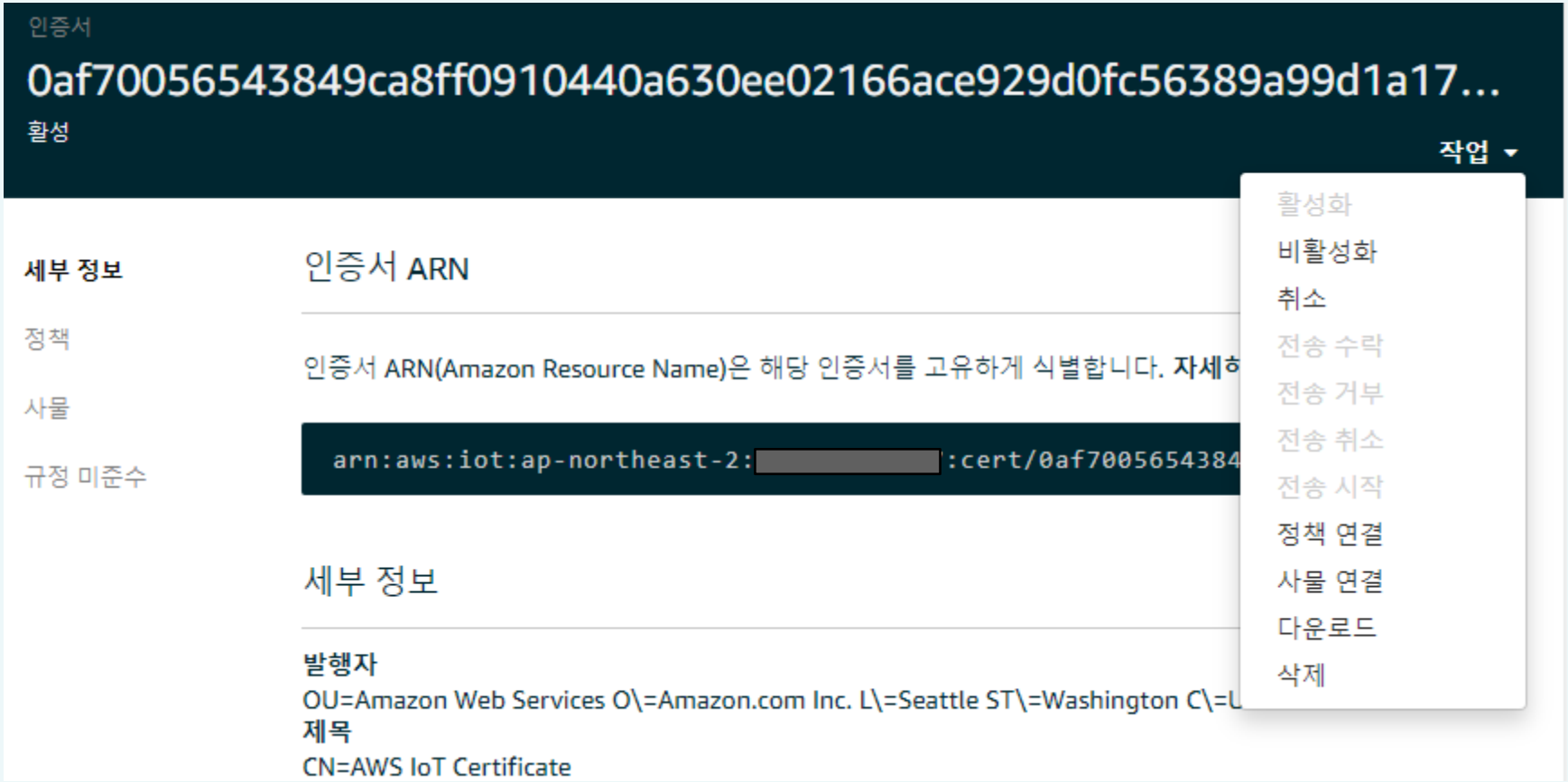
정책 문서

정책 문서에서는 요청의 권한을 정의합니다. **자세히 알아보기**

버전 1 2019. 8. 12. 오후 4:54:33 업데이트됨 **정책 문서 편집**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

Register Thing – 정책/사물연결



Design Topics / Payloads

사물

myTestThing

유형 없음

작업

세부 정보

보안

사물 그룹

결제 그룹

섀도우

상호 작용

이 사물은 이미 연결되어 있는 것 같습니다.

디바이스 연결

HTTPS

이 Rest API 엔드포인트를 사용하여 사물 섀도우를 업데이트합니다. **자세히 알아보기**

```
[redacted]-ats.iot.ap-northeast-2.amazonaws.com
```

MQTT

주제를 사용하여 애플리케이션 및 사물이 사물(사물 섀도우)의 상태 정보를 가져오거나 업데이트하거나 삭제할 수 있도록 합니다. **자세히 알아보기**

사물 섀도우 업데이트

```
$aws/things/myTestThing/shadow/update
```

Design Topics / Payloads

섀도우 ARN

섀도우 ARN은 이 사물의 섀도우를 고유하게 식별합니다. **자세히 알아보기**

```
arn:aws:iot:ap-northeast-2:[redacted]:thing/myTestThing
```

섀도우 문서

최근 업데이트: 2019. 11. 16. 오후 8:23:35

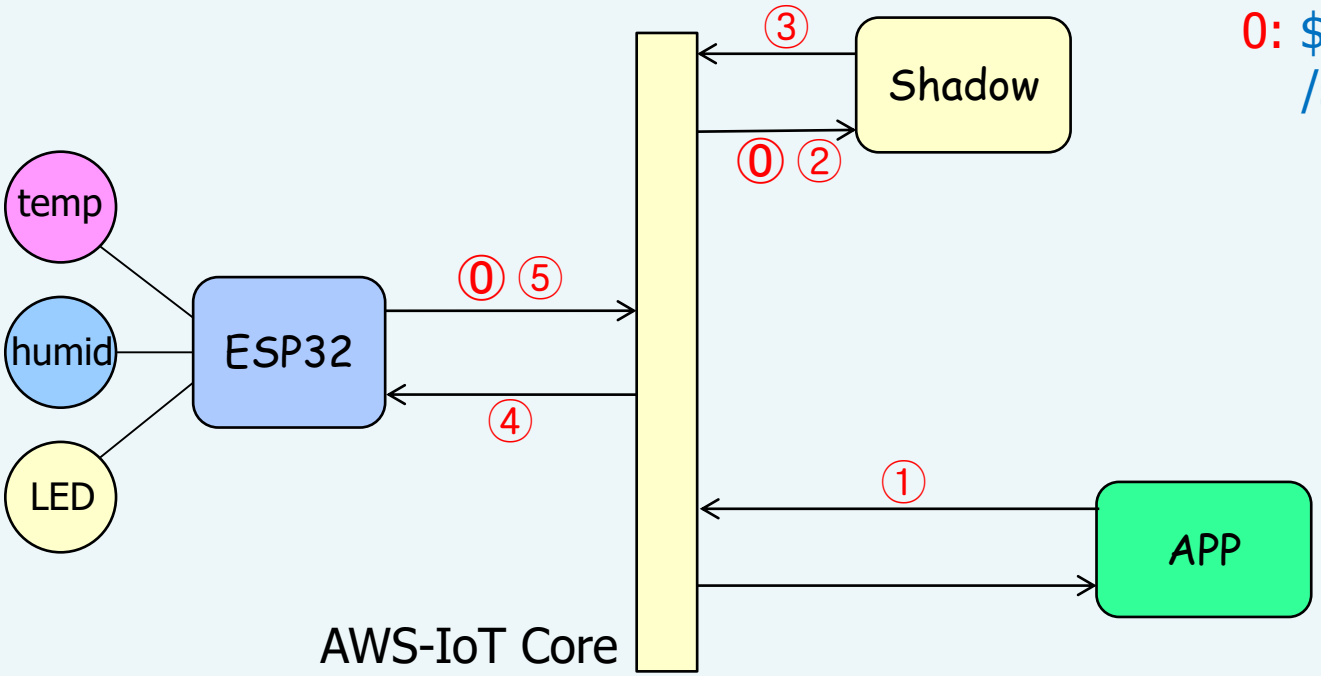
섀도우 상태:

```
{
  "desired": {
    "led": "OFF"
  },
  "reported": {
    "temp": 22.9,
    "humid": 72,
    "led": "OFF"
  }
}
```

```
{ "state" : {
      "reported" : {
          "temp" : nn,
          "humid" : nn,
          "led" : "ON" | "OFF"
      }
  }
}
```

```
{ "state" : {
      "desired" : {
          "led" : "ON" | "OFF"
      }
  }
}
```

Design Payloads / Payloads



0: $aws/things/myTestThing /shadow/update

```
{"state": {
    "reported": {
        "temp": 22,
        "humid": 64,
        "led": "OFF"
    }
  }
}
```

Current state
5초 마다

1: .../update {"state": {"desired": {"led": "ON"}}}
2: .../update {"state": {"desired": {"led": "ON"}}}
3: .../update/delta {"state": {"led": "ON"}}
4: .../update/delta {"state": {"led": "ON"}}
5: .../update {"state": {"reported": {"temp": 22, "humid": 64, "led": "ON"}}}

Create Rules

규칙

Control_led_shadow

활성

작업

개요

Tags

설명

편집

control led according to temp republish topic /update "desired" led: "ON" (temp<20)

규칙 쿼리 설명문

편집

이 규칙을 사용하여 처리하고자 하는 메시지의 소스입니다.

```
SELECT {'desired':{'led':"ON"}} as state FROM '$aws/things/myTestThing/shadow/update'
WHERE state.reported.temp < 20
```

SQL 버전 사용 2016-03-23

작업

작업은 규칙이 트리거되면 이루어지는 것입니다. **자세히 알아보기**

메시지를 AWS IoT 주제에 재게시

$$aws/things/myTestThing/shadow/update

제거 편집

Create Rules

규칙

control_ledOFF_shadow

활성

작업

개요

Tags

설명

편집

control led according to temp republish topic /update "desired" led: "OFF" (temp>25)

규칙 쿼리 설명문

편집

이 규칙을 사용하여 처리하고자 하는 메시지의 소스입니다.

```
SELECT {'desired':{'led':"OFF"}} as state FROM '$aws/things/myTestThing/shadow/update'
WHERE state.reported.temp > 25
```

SQL 버전 사용 2016-03-23

작업

작업은 규칙이 트리거되면 이루어지는 것입니다. **자세히 알아보기**

메시지를 AWS IoT 주제에 재게시

$$aws/things/myTestThing/shadow/update

제거 편집

Create Rules

작업 구성

메시지를 AWS IoT 주제에 재게시
AWS IOT 재게시

이 작업은 메시지를 다른 AWS IoT 주제에 재게시합니다.

*주제

$$aws/things/myTestThing/shadow/update

Reserved Topic 을 republish 할 경우 '$' 대신 '$$' 을 사용해야 함

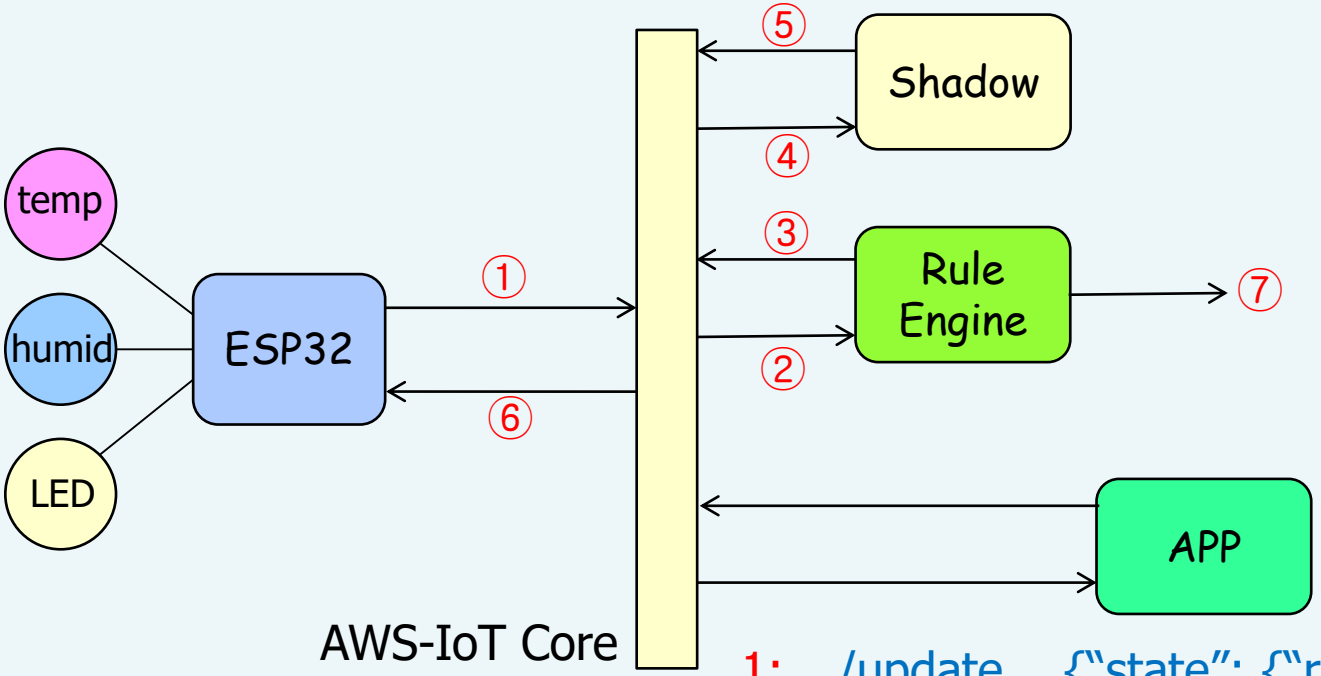
서비스 품질

- (●) 0 – 메시지가 0번 이상 전달됩니다.
- ( ) 1 – 메시지가 1번 이상 전달됩니다.

이 작업을 수행할 권한을 AWS IoT에 부여하는 역할을 선택하거나 생성합니다.

republish_role	역할 생성	선택

Create Rules



** 3: republish Topic
** 7: ex) SNS

1: …/update {"state": {"reported": {"temp": 19, "humid": 64, "led": "OFF"}}}

2: = 1

3: …/update {"state": {"desired": {"led": "ON"}}}

4: = 3

5: …/update/delta {"state": {"led": "ON"}}

Create Rules

- Control_led_shadow

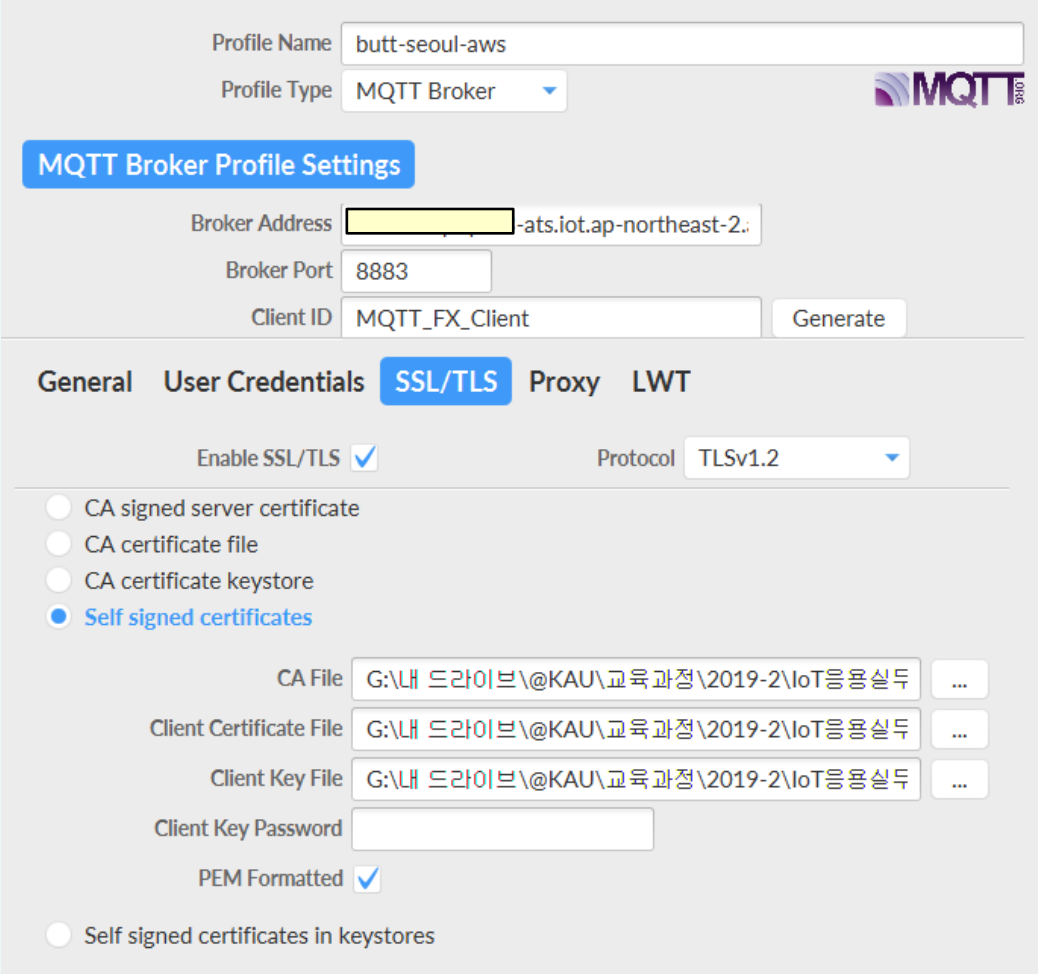
```
SELECT {'desired':{'led':"ON"}} as state
FROM '$aws/things/myTestThing/shadow/update'
WHERE state.reported.temp < 20
```

- Control_ledOFF_shadow

```
SELECT {'desired':{'led':"OFF"}} as state
FROM '$aws/things/myTestThing/shadow/update'
WHERE state.reported.temp > 25
```

Test Shadow by MQTT-fx

- Thing을 만들기전에 Topic/Payload 동작 확인
  - Broker addr:
    Thing Rest-API endpoint
  - Broker Port: 8883
  - CA, Certificate,
    Private-key file 지정



Test Shadow by MQTT-fx

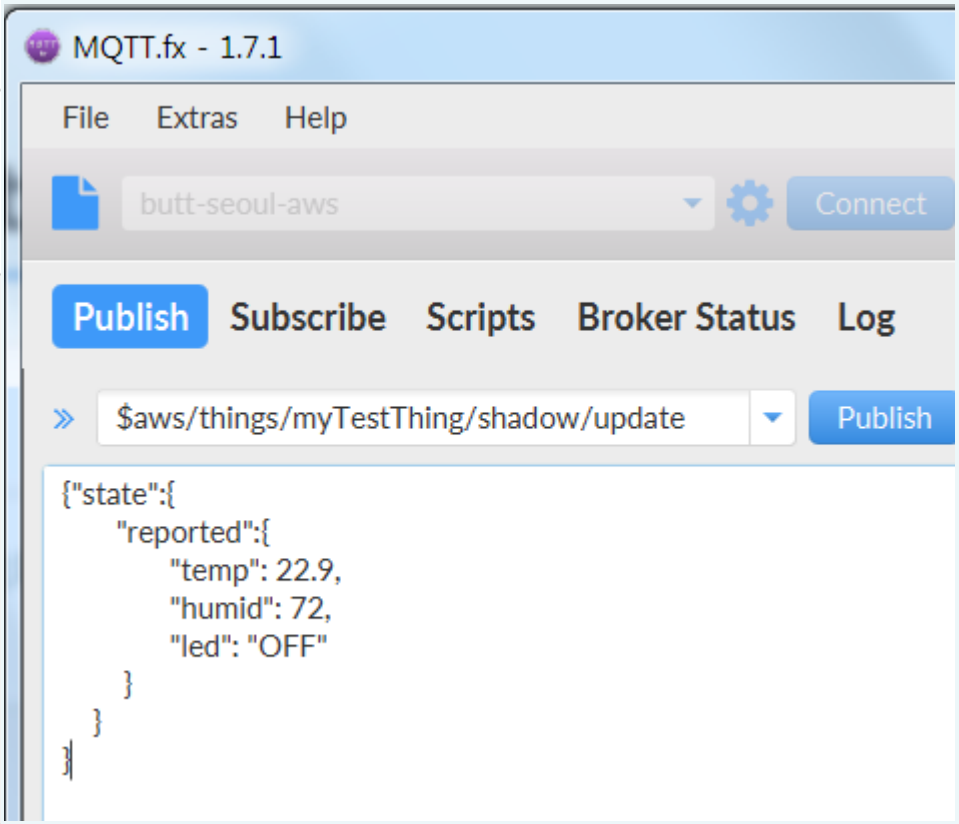
- Shadow 상태를 보면서 pub / sub 동작을 시험함

섀도우 문서

최근 업데이트: 2019. 11. 16. 오후 8:23:35

섀도우 상태:


```
{
  "desired": {
    "led": "OFF"
  },
  "reported": {
    "temp": 22.9,
    "humid": 72,
    "led": "OFF"
  }
}
```

Test Shadow by MQTT-fx

- Things와 Shadow 와의 Topic Flow

<Thing> 온습도값 Publish → <shadow> 상태변경
<Thing> Subscribe: shadow으로부터의 명령 (../update/delta)
<App> led 변경명령 Publish →
<shadow> 현재값과 차이: delta 생성
<shadow> led 변경명령 Publish → <Thing> led 변경

Test Shadow by MQTT-fx