ThingPlug

ThingPlug® API Document

For LoRa-Based & IoT Application Developement

Version 1.7

SK telecom

Contents

1 개요

본 문서는 ThingPlug를 기반으로 IoT 어플리케이션을 개발하는 개발자들을 위해 SK 텔레콤이 제공하는 API에 대한 문서로써, 본 문서에서는 ThingPlug API 중에서도 LoRa 기반으로 IoT 어플리케이션을 개발하고자 하는 개발자들의 이해를 돕고자 다음 내용을 포함합니다.

- ThingPlug 구조 소개
- ThingPlug 접속 방법 소개
- LoRa 기반 IoT 어플리케이션 개발을 위해 API를 제공하는 ThingPlug 플랫폼 소개
- LoRa 기반 IoT 어플리케이션 개발을 위해 필요한 API 설명
- LoRa 네트워크 구조
- ThingPlug API가 준수하는 oneM2M 표준에 대한 설명

2 참고문서

본 문서는 아래의 문서들을 참고한다. 해당 문서는 ThingPlug 홈페이지를 통해 다운 받으실 수 있습니다.

2.1 oneM2M 표준 규격

- oneM2M Functional Architecture Specification (TS-0001) V1.12
- oneM2M Protocol Specification (TS-0004) V1.4
- oneM2M MQTT Binding Specification (TS-0010) V1.5
- oneM2M HTTP Binding Specification (TS-0009) V1.5

2.2 LoRa 표준 규격

- LoRaWAN Specification V1.0.1

3 LoRa 네트워크 구조

SK 텔레콤에서 구축한 LoRa 네트워크는 아래 그림 1 과 같이 ThingPlug, LoRa 네트워크 서버, LoRa 기지국, LoRa 디바이스로 구성되어 있습니다.

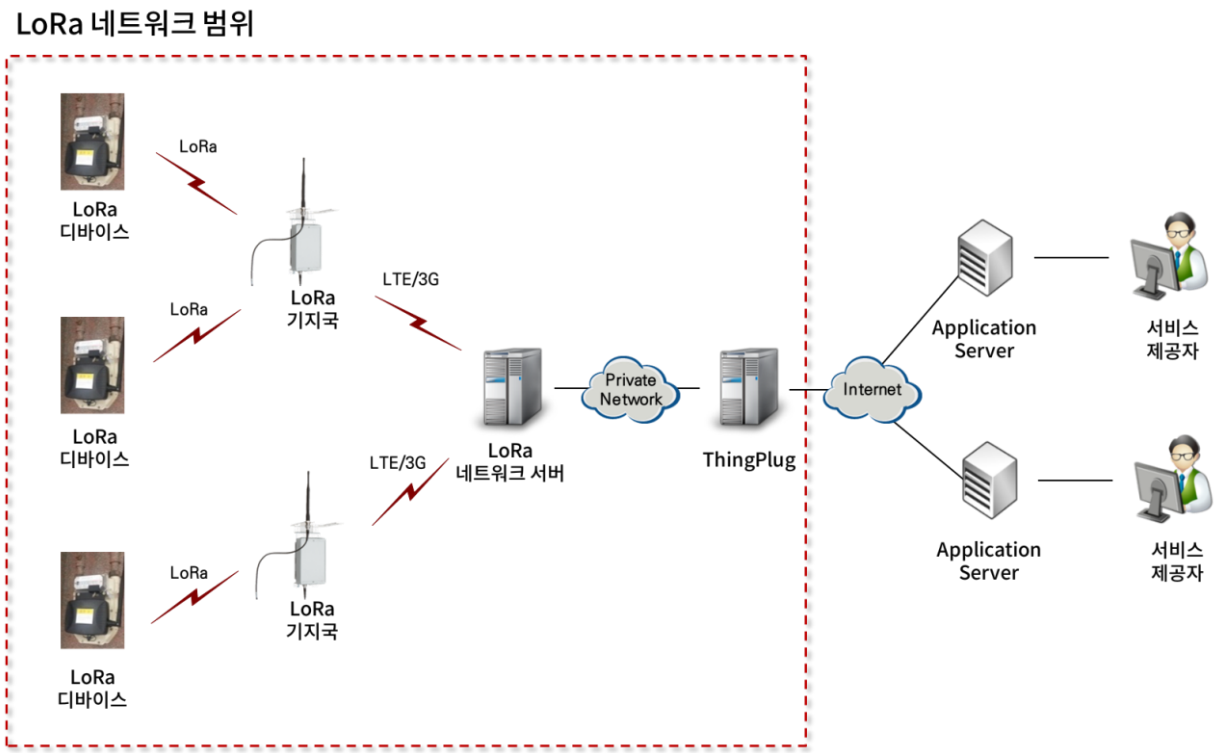


그림 1 LoRa 네트워크 구조

3.1 LoRa 네트워크 구성 요소

구분	설명

ThingPlug	• ThingPlug 는 LoRa 네트워크 서버와 연결되어 LoRa 디바이스로부터 생성된 데이터를 관리하는 기능을 하며, 외부 엔티티의 LoRa 네트워크 서비스 접근을 위한 기능을 제공합니다.
	• ThingPlug 는 LoRa 기반으로 IoT 서비스를 개발하는 개발자에게 LoRa 기술을 모르더라도 어플리케이션을 개발할 수 있도록 oneM2M 표준 기반의 API(Application Programming Interface)를 제공합니다.
	  • LoRa 디바이스 및 서비스 등록
	  • LoRa 디바이스의 주기 보고 정보 저장
	  • LoRa 디바이스 제어

	• LoRa 네트워크 정보 접근 제어 및 보안
LoRa 네트워크 서버	• LoRaWAN 표준 규격에서 언급하는 Network Server Entity 를 의미합니다. • LoRa Network Server 는 LoRa 디바이스를 인증하고, 데이터를 전달하는 역할을 하며, LoRa 디바이스 리스트 관리 기능, LoRa 기지국 연동 기능, ThingPlug 연동 기능, Application Data 전달 기능, Downlink 데이터 전송을 위한 LoRa Gateway 선택 기능 등을 수행합니다. • LoRa Network Controller 기능: 무선망 관련 값들을 설정하고, 무선 정책 설정 기능, LoRa 디바이스 및 게이트웨이 등록 기능 등을 수행합니다. • LoRa Application 관리 기능: Application 별 고유 서비스를 관리/제어하는 기능으로서, Application 별 데이터 전송/관리 기능, 데이터 암호화/복호화 기능 등을 수행합니다.
LoRa 기지국	• LoRa 기술이 적용된 기지국으로서, LoRa 디바이스와 LoRa 네트워크 서버 간 데이터를 전달하는 역할을 하고, LoRa 통신용 기지국 프로토콜 및 LoRa Network Server 연동 기능 등을 수행하는 장치입니다.
LoRa 디바이스	• LoRa 기술을 적용한 디바이스로서 LoRa Class 별 데이터 전송 동작 및 패킷 암호화/복호화 기능 등을 수행하는 장치입니다.

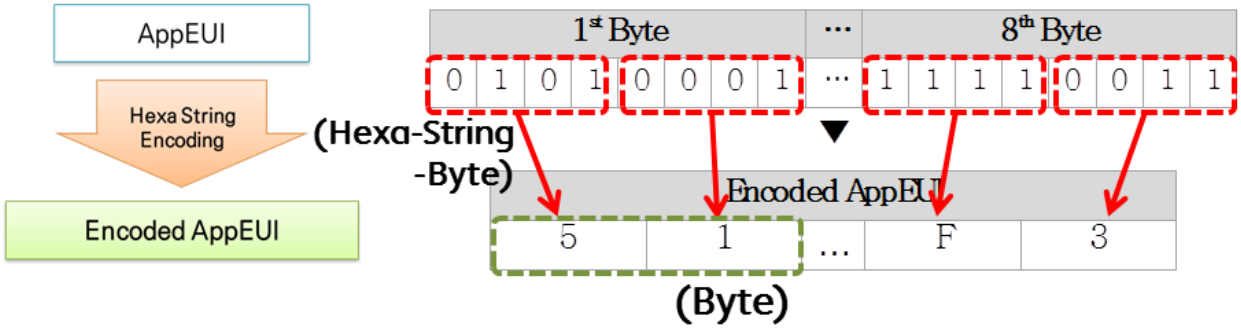
4 LoRa 식별 체계

4.1 AppEUI (Application EUI Extended Unique Identifier)

AppEUI는 LoRa의 서비스를 식별하는 식별자입니다. LoRa 네트워크 시스템에서 Application은 한 개 이상의 LoRa디바이스로 구성된 동일한 서비스를 구성하는 그룹을 의미합니다. 실제로는 수도 검침 서비스가 하나의 Application이 될 수 있습니다.

LoRa 표준에서 Application 식별자는 IEEE Institute of Electrical and Electronics Engineers 에서 정의한 64-bit의 EUI로 정의됩니다. LoRa 서비스 식별자는 별도의 과정을 통해 SK텔레콤으로부터 할당 받습니다.

ThingPlug 에서 상기 AppEUI 를 사용할 때는 Hexa String 으로 사용합니다. 주의할 점은 알파벳은 소문자로 표기한다는 것입니다.



4.2 LTID (LoRa & ThingPlug ID)

NW서버는 ThingPlug와 연동 시에, 단말을 구분하기 위해 LTID(LoRa & ThingPlug ID)를 사용하여야 합니다. LTID는 당사 규격에서 신규 정의한 값이고, AppEUI, DevEUI를 조합 한 값으로서 Globally Unique 한 값입니다.

LTID 는 AppEUI, DevEUI 를 조합한 이후, AppEUI 앞부분의 4byte 를 제외하고, Hexa string 으로 나타낸 24byte 값입니다. 알파벳은 소문자로 표기하여야 합니다.

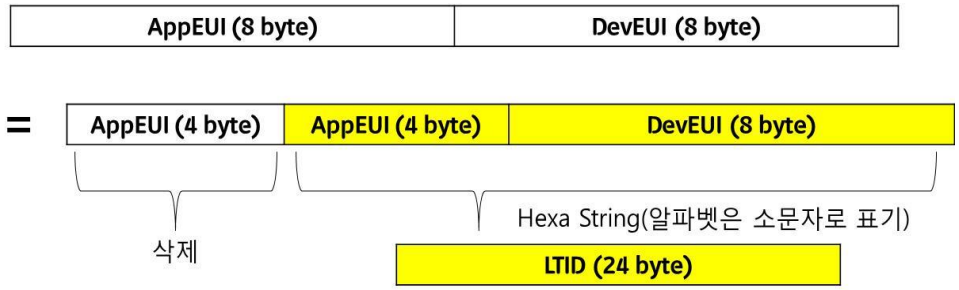


그림 2 LTID 구조

4.3 Application Server ID (AppSVRID)

Application Server에서 제공하는 IoT 어플리케이션의 식별자로써 Application Server 개발자가 등록한 ThingPlug ID를 사용합니다. (Case Sensitive) ThingPlug에 가입하여 디바이스를 등록하고, API 접근을 위해 사용하는 Key 정보 등에 대한 내용은 ThingPlug 홈페이지에 [LoRa 개통 프로세스 매뉴얼]을 참고 부탁드립니다.

5 ThingPlug 접속 방법

LoRa 기반 IoT 어플리케이션 개발자가 LoRa 디바이스로부터 ThingPlug 까지 송신된 주기보고, 등록 정보 또는 LoRa 디바이스를 제어하기 위해서는 ThingPlug 에 접속한 후 ThingPlug 에서 제공하는 API 에 접근하여 어플리케이션에서 필요로 하는 기능을 호출하면 됩니다. 그러기 위해서는 가장 먼저 ThingPlug 에 접속하여야 하는데, 이 장에서는 oneM2M 기반의 ThingPlug 에 접속하는 방법을 소개 하겠습니다.

본 장에서는 REST 에 대한 개념과 MQTT 및 HTTP 프로토콜에 대한 사전지식을 필요로 하며, 간단하게 'ThingPlug Tip' 섹션를 참고 부탁 드립니다.

5.1 ThingPlug 구조

oneM2M 기반 ThingPlug 는 그림 3 과 같이 IoT 어플리케이션 및 IoT 디바이스 개발자들이 공통적으로 필요로 하는 기능을 포함하고, 이를 API 형태로 제공합니다.

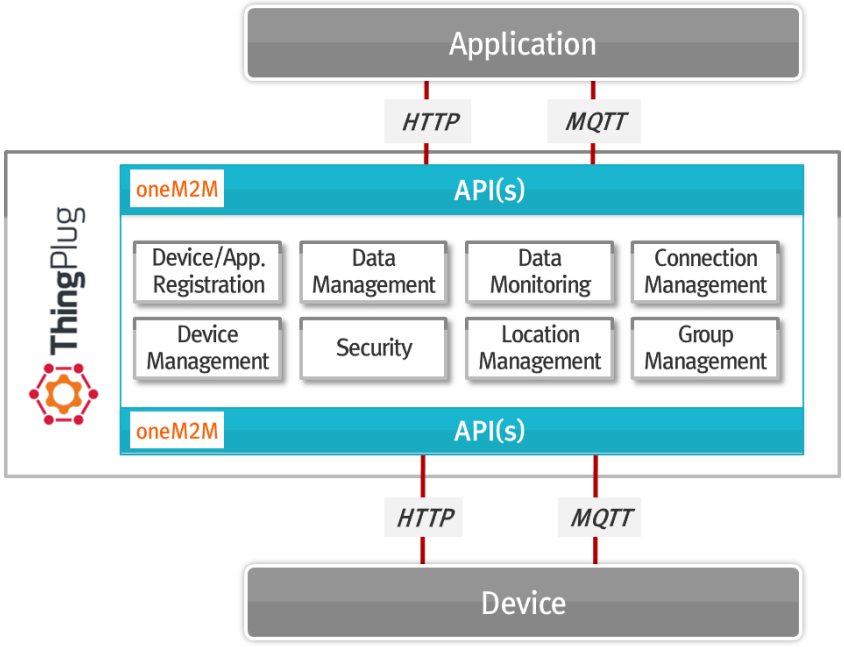


그림 3 oneM2M 표준 기반 ThingPlug 구조

ThingPlug 외부에서 oneM2M 표준 기반 RESTful API 에 접근하기 위해서는 HTTP 와 MQTT 를 사용합니다. oneM2M 표준에는 CoAP 프로토콜 기반으로 API 를 호출하는 방식을 정의하였지만, 저사양 장치 및 근거리 통신에 적합한 CoAP 프로토콜 방식은 제공하지 않습니다. ThingPlug 에서는 저사양 장치를 위해 SK 텔레콤 독자적으로 개발한 TCP 기반 API 를 제공합니다. 이에 대한 자세한 설명은 ThingPlug 홈페이지를 참고 부탁 드립니다.

5.2 ThingPlug 환경 구성

LoRa용 ThingPlug는 최초 단말을 연동하여 테스트하여 보는 개발자용 오픈 테스트베드와 상용 서비스를 위한 SKT LoRa 상용 ThingPlug로 구성 되어있습니다.

오픈 테스트 베드
- 개발 지원
- 상용망 도입 전 필수 검증

SKT LoRa 상용 ThingPlug
- 시험 번호 기반 필드 테스트
- 상용 서비스
- 오픈 테스트 베드 검증 후 진입 (필수)

항목	오픈테스트베드	상용망 시험번호	상용망 서비스
용도	개발	시험번호 기반 필드 테스트	상용 서비스
이용 자격	테스트 베드 이용 신청 완료	시험번호 사용 허가 및 신청 완료 오픈 테스트베드 검증 완료	SKT LoRa 상용 서비스도입 협의 완료 오픈 테스트베드 검증 완료
이용 요금	무료	무료	담당AM과 협의
이용 기간	무제한	3개월	계약 기간
사용 신청	http://lora.sktiot.com	담당 매니저에게 문의	담당 매니저에게 문의
포털 주소	http://thingplug.sktiot.com	담당 매니저에게 문의	담당 매니저에게 문의
플랫폼 호스트 주소	http://onem2m.sktiot.com	담당 매니저에게 문의	담당 매니저에게 문의
버전	1.0 (loratestbed/v1_0)	1.0 ({AppEUI}/v1_0)	1.0 ({AppEUI}/v1_0)
단말 등록 주체	파트너사	대리점	대리점
서비스ID	공동 사용	공동 사용 (서비스 유형/지역 별)	개통 시스템을 통한 발급
ThingPlug ID	직접 생성 사용	개통 시스템을 통한 발급 (20자 이하)	개통 시스템을 통한 발급 (20자 이하)

※ 유의사항
- 반드시 오픈 테스트베드를 통해 기본 가이드 및 검증을 거친 후 상용망에 등록 합니다.
- 상용망 등록 과정은 CCBS 를 통해 자동으로 이루어집니다 (상용 포털에 대한 직접 접근은 금지합니다.)
- 시험번호로 필드 테스트 시 시스템 과부하를 유발하는 동작을 삼갑니다.

5.3 HTTP를 통한 ThingPlug 접속 방법

Web 을 위해 많이 사용되는 HTTP 기반으로 ThingPlug 에 접속하기 위해서는 다음 경로를 통해서 접속합니다.

Host	http://onem2m.sktiot.com

Port	9000(HTTP), 9443 (HTTPS)
Service Name	loratestbed
Version	1.0 (URL에서는 v1_0, case sensitive)

상기 정보 중에서 Service Name 은 REST 구조에서 Host 와 Port 를 작성한 후 가장 먼저 나오는 Resource 로써 서비스를 구분하는 구분자로 인식하시면 되며, Version 은 각 서비스별로 Version 을 구분할 수 있는 Version 정보로 인식하시면 됩니다. 만약, ThingPlug 를 통해 smarthealth 1.0 라는 서비스가 동작되고 있다고 하면 이는 https://onem2m.sktiot.com:9443/smarthealth/v1_0 으로 표기될 수 있을 것입니다.

아래 예제는 HTTP Client 테스트 프로그램으로 잘 알려진 'Postman' (http://www.getpostman.com)으로 ThingPlug 에서 접속한 결과를 보여줍니다. 위에 표에서 언급한 경로와 ThingPlug 에서 제공하는 가이드 라인을 기반으로 Postman 을 구동하면 다음 그림의 하단과 같이 API 호출했을 때의 결과값을 확인하실 수 있습니다. 접속 후에 다양한 API 를 어떻게 활용해야 하는 것인지는 다음 6 장을 참고하시면 됩니다.

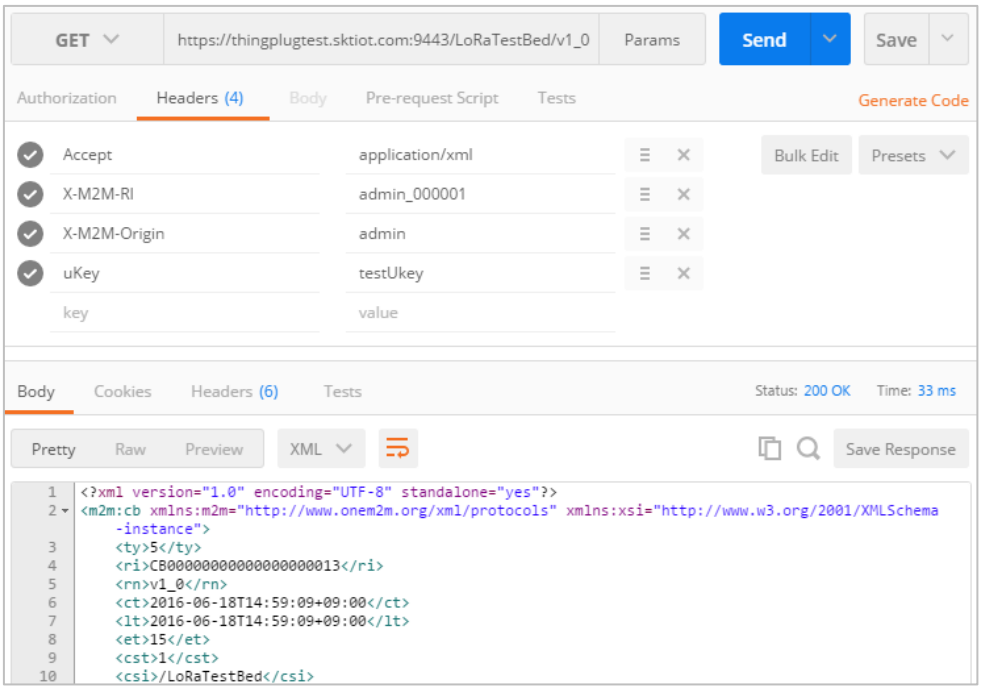


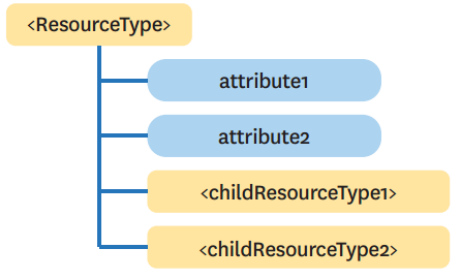
그림 4 HTTP 기반으로 ThingPlug 를 호출한 결과

ThingPlug Tip! RESTful Architecture and Resource-based API

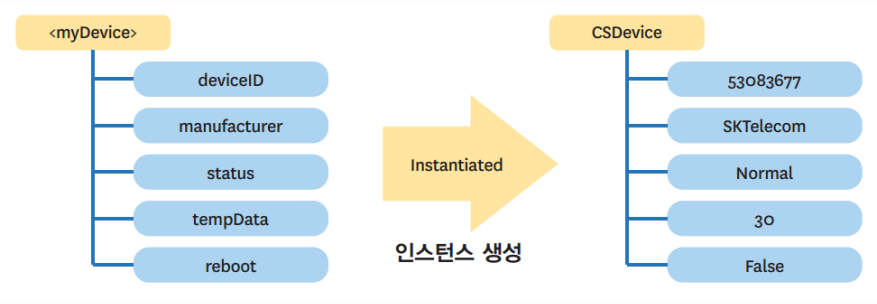
oneM2M 표준에 정의된 API 를 호출하기 위해서는 약속된 형태의 통신을 해야 합니다. 이때 통신 방식은 패턴이 있는데 요청 Request 과 응답 Response 이 쌍을 이룹니다. 즉, 발신자 Originator 가 해당 기능을 제공하는 수신자 Receiver 에 요청을 하는데, 명시된 REST 기반 자원 API 에 대해 'Request'를 하는 식입니다. 그 API 는 자원을 생성 CREATE, 조회 RETRIEVE, 갱신 UPDATE 및 삭제 DELETE 를 할 수 있는 곳을 나타냅니다. 이렇게 호출 받은 수신자는 해당 'Request'의 동작 결과를 수행한 후 'Response' 형태로 전달합니다.

oneM2M 에서 정의한 기능 호출을 위한 REST 기반 API 의 구조는 다음과 같이 자원 API 또는 자원

API Resource Type 또는 Resource API 으로 표현되며, 각 자원 API 은 하나 이상의 속성 Attribute 과 추가적인 기능을 명시하기 위해서 자식 자원 API 을 포함합니다. 여기서 속성은 일반적으로 함수 호출할 때 포함하는 인자 Argument 와 같다고 생각하면 됩니다.



또한 각 자원이 실제로 생성되면 별도의 자원의 이름 Resource Name 을 부여합니다. 이는 마치 객체 지향 언어에서의 클래스 Class 와 인스턴스 Instance 의 차이와 같다고 생각하면 됩니다.
예를 들어, 장치의 기본 정보를 표기하고 제어하는 자원 API 가 존재한다고 합시다. 장치의 식별자, 상태 표기 및 재부팅 제어를 할 수 있는 속성을 가진 〈myDevice〉 자원 API(그림 왼쪽)이 있다고 가정하겠습니다. 다음 그림과 같이 HTTP 프로토콜을 통해서 IoT 장치 정보를 저장하기 위해서는 아래의 메시지를 ThingPlug 서버에 전송합니다. 전송되면 그림의 오른쪽과 같이 ThingPlug 에 해당 장치의 기본 정보가 저장됩니다.




```
[Header]
POST /CSDevice
Host: https://sandbox.sktiot.com
Accept: application/xml
ResourceType : myDevice
[Content]
<?xml>
<myDevice>
   <deviceID> 53083677 </deviceID>
   <manufacturer> SKTelecom </manufacturer>
   <status> Normal </status>
   <tempData> 30 </tempData>
   <reboot> False </reboot>
</myDevice>
```

장치가 오작동해서 재 부팅해야 하는 상황이 올 수도 있습니다. 이 경우 추가로 〈myDevice〉 자원 API 정의 중 'reboot' 속성을 'True'로 설정하면 인스턴스와 연결되어있는 실제 디바이스를 재부팅하는 것이라고 가정해 볼 수 있습니다. 이때는 기존의 인스턴스화 되어 있는 자원을 갱신하는 PUT 요청에 따라 'reboot' 속성을 갱신함으로써 해당 기능을 호출할 수 있게 됩니다.

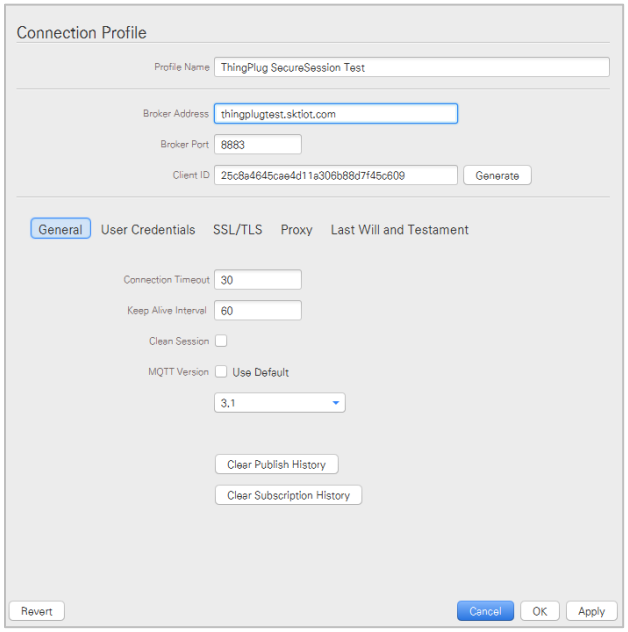
5.4 MQTT를 통한 ThingPlug 접속 방법

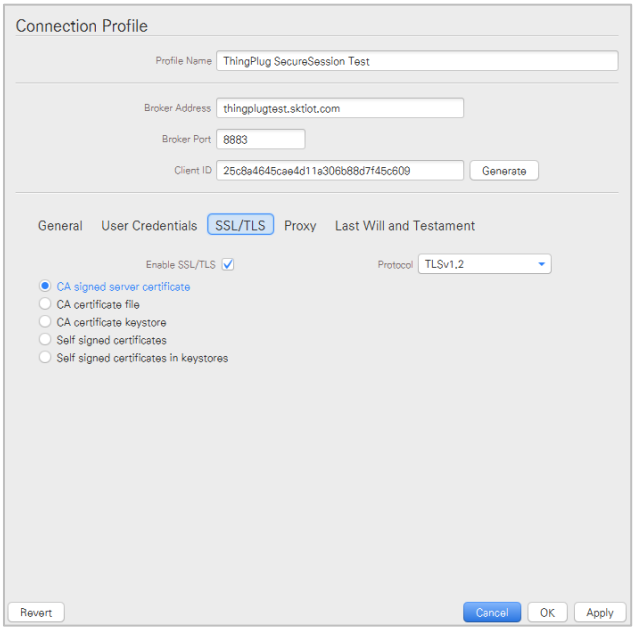
저사양 및 Push 메시지 전달에 용이한 MQTT 기반으로 ThingPlug 에 접속하기 위해서는 다음 경로를 통해서 접속합니다.

Host	onem2m.sktiot.com
Port	1883(MQTT), 8883 (MQTTS)
Client ID	[ThingPlug 홈페이지 ID]_[사용자 정의 Post-Fix]
UserName	ThingPlug 홈페이지 ID
Password	상기 해당 ID의 uKey
TLS Version	TLS 1.2 (8883(MQTTS) 접속 시)
Clean Session	True

MQTT 프로토콜은 접속 시 (CONNECT 메시지 전달 시) 위의 표에 있는 Client ID, Username, Password 를 포함한 후, 접속을 우선 수행하여 세션을 연결한 후에 자신이 관심이 있는 Topic 구독(Subscription)을 요청하고, ThingPlug 의 API 호출을 요청(Publish) 합니다.

아래 예제는 MQTT Client 테스트 프로그램으로 잘 알려진 'MQTT.fx'(http://mqttfx.jfx4ee.org)로 ThingPlug 에서 접속한 결과를 보여줍니다. 위에 표에서 언급한 세팅 값과 ThingPlug 에서 제공하는 가이드 라인을 기반으로 MQTT.fx 를 구동하면 다음 그림의 하단과 같이 API 호출했을 때의 결과값을 확인하실 수 있습니다. 접속 후에 다양한 API 를 어떻게 활용해야 하는 것인지는 다음 6 장을 참고하시면 됩니다.





ThingPlug Tip! MQTT로 ThingPlug와 통신하기

MQTT(Messaging Queuing Telemetry Transport)는 최근 IoT 시대에 와서 각광 받는 프로토콜 기술

중 하나입니다. 특징은 실시간 및 경량 메시징에 적합한 Publish/Subscribe 구조로 구성된 프로토콜로 Web 에서 사용하는 Request/Response 구조의 HTTP 프로토콜 대비 가볍게 메시지를 주고 받을 수 있다는 점과 Always on 형태로 연결을 Broker 에서 관리하여 푸시가 많은 IoT 에 적합하는 점입니다.

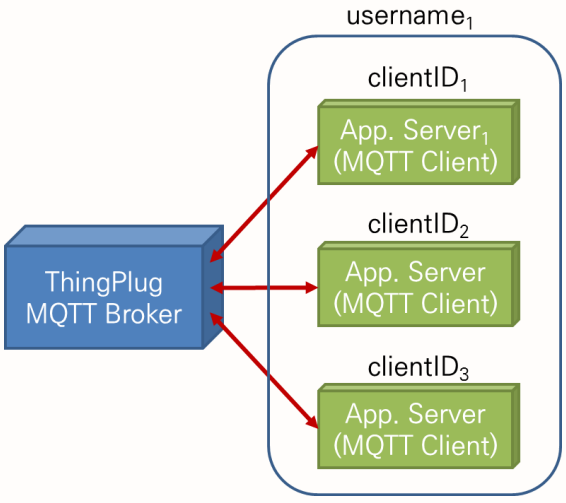
ThingPlug 에서도 MQTT 프로토콜을 지원합니다. 본 팁에서는 ThingPlug 에서 보다 안전하게 MQTT 프로토콜을 사용하실 수 있도록 제공하는 정책에 대해 설명을 드리고자 합니다. MQTT 프로토콜은 아래 주소로 접속하시면, 표준 문서를 보실 수 있습니다.
(http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf)

[ThingPlug MQTT Broker에 접속]

ThingPlug MQTT Broker에 정상적으로 접속하기 위해서는 Broker 입장에서 접속하는 접속자가 정상적인 접속자 인지 파악하는 인증 단계(Authentication)를 거칩니다. 이를 위해 필요한 값이 MQTT 프로토콜에서 브로커에 접속할 때, 표기하는 내용 중에 하나인 username과 password 입니다. 5.3절의 표의 내용과 같이 ThingPlug MQTT Broker에 접속할 때, username과 password를 각각 'ThingPlug 홈페이지 ID'와 'ID의 uKey'로 할당 해주시면 됩니다. 만약, 정식 인증 받지 않는 접속자인 경우에는 'Not authorized to connect'라는 에러를 리턴하게 됩니다.

정상적인 접속을 위해 추가적으로 필요한 정보는 clientID 입니다. clientID는 말 그대로 Client의 식별자를 의미합니다. 그렇다면 username과 clientID는 어떤 차이가 있을까요? 아래 그림과 같이 username은 Broker 자체 접속하기 위한 권한을 의미합니다. ThingPlug ID를 하나 할당 받으신 개발자분들께서는 하나의 username을 할당 받으시고, 해당 username에 다양한 목적의 복수 개의 MQTT Client를 할당하여 시스템을 구성할 수 있고, 각 MQTT Client는 기본적으로 uniqueness가 보장되도록 할당하여 합니다. ThingPlug MQTT Broker에서 clientID를 할당하는 방법은 다음 방법을 따릅니다. (이미 username은 ThingPlug 홈페이지에서 가입 시 uniqueness를 확인합니다)

- clientID = **[ThingPlug 홈페이지 ID]_[별도 Post-fix]**
  예) ThingPlug 홈페이지 ID 가 thingplugMqtt 인 경우에는, clientID 를 다음과 같이 할당합니다. thingplugMqtt_0001, thingplugMqtt_0002 와 같이 할당합니다. Post-fix 에 대한 다른 Rule 은 없으나, ID 중복이 되는 경우에는 기존에 사용하는 ID 의 접속이 Disconnect 됩니다.



[Subscribe 정책]

ThingPlug로부터 Push 메시지를 수신하기 위해서, 또는 Application Server가 ThingPlug로부터 요청에 대한 응답을 수신하기 위해서는 아래 Topic에 Subscribe를 해야합니다. 아래의 정책을 따르지 않으면, Subscribe를 정상적으로 실행할 수 없습니다.

- **푸시를 받기 위한 Topic**
  /oneM2M/req_msg/+/[clientID]
- **요청에 대한 응답을 수신하기 위한 Topic**
  /oneM2M/resp/[clientID]/+

[Publish 정책]

메시징에서 중요한 부분 중에 하나가 발신자를 명확히 해야 한다는 점입니다. 만약 발신자 정보를 위조해서 보내면 중대한 보안 사고가 발생할 수도 있습니다. 그리하여 ThingPlug MQTT 프로토콜에서는 발신자가 실제 메시징을 발행(Publish)하는 객체가 되도록 제한됩니다. oneM2M의 표준을 일부 차용하는 ThingPlug에서는 다음과 같은 Topic 구조를 가집니다.

- **ThingPlug 에 요청하는 API 호출인 경우 (App. Server to ThingPlug)**
  From *To*
  /oneM2M/req/[요청 메시지 발신자]/[요청을 수신하는 서비스]

- **ThingPlug 로부터 요청의 응답을 수신하는 API 인 경우 (ThingPlug to App. Server)**
  From *To*
  /oneM2M/resp/[요청을 수신하는 서비스]/[요청 메시지 발신자]

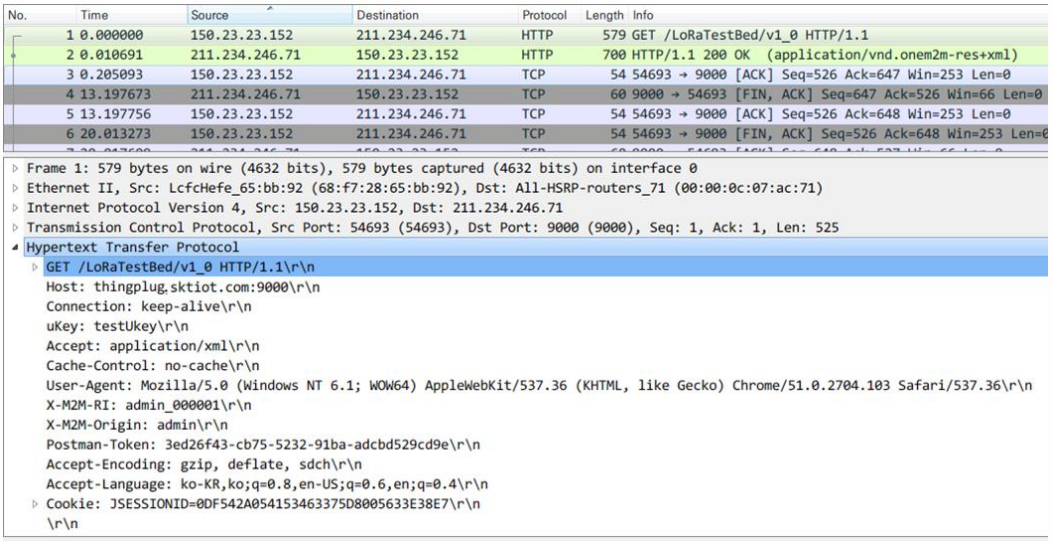
상기 2가지 경우 중에서 ThingPlug에 요청하는 API 호출이 Application Server가 Publish 하는 경우입니다. 이때, [요청 메시지 발신자]가 ThingPlug MQTT Broker에 접속한 clientID와 상이한 경우에 ThingPlug MQTT Broker는 보안의 이유로 해당 Client의 접속을 차단합니다.

ThingPlug Tip! MQTT vs. MQTTS | HTTP vs. HTTPS?

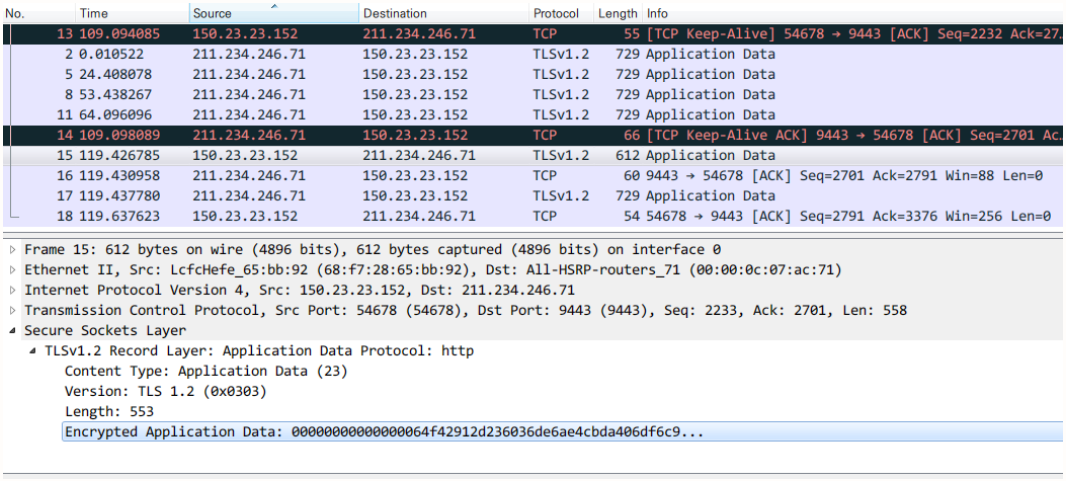
IoT 시대가 도래함에 따라 다양한 장치들이 인터넷에 연결되기 시작하고 있고, 이에 따라 다양한 보안 이슈들이 발생하기 시작했습니다. IoT 플랫폼으로써 ThingPlug 는 다양한 디바이스에 대한 정보 및 어플리케이션에 대한 정보를 API 를 통해 저장 및 호출할 때, 해당 정보들이 암호화 될 수 있도록 HTTP, MQTT 프로토콜 정보들이 TLS (Transport Layer Security) 위에서 전달될 수 있도록 각각 HTTPS, MQTTS 를 지원합니다.

아래 그림은 ThingPlug 에 접속할 때, HTTP 와 HTTPS 프로토콜로 전달되는 패킷을 패킷 감시 프로그램인 Wireshark 로 캡쳐하여 보여주고 있습니다.

[HTTP 인 경우]



[HTTPS 인 경우]

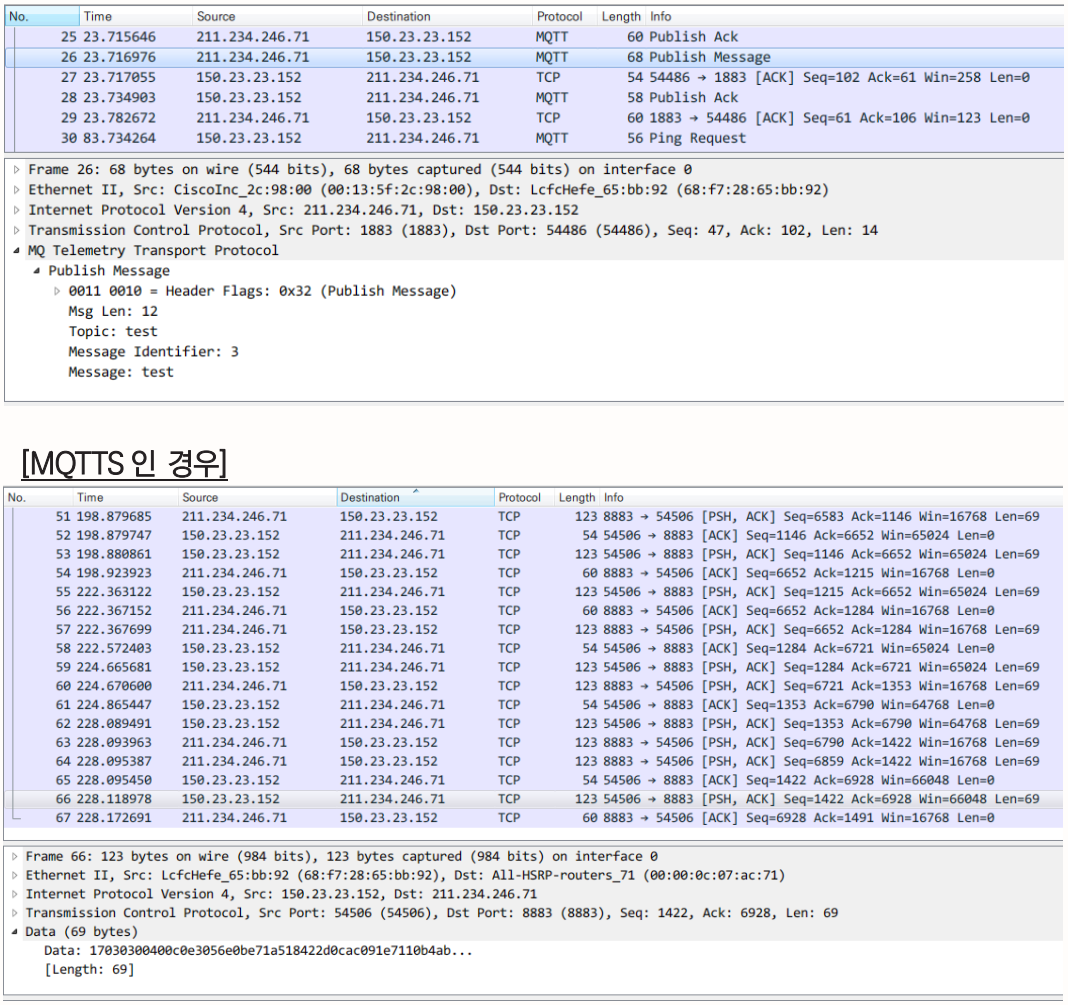


위의 캡쳐 화면에서 보실 수 있는 것과 같이 HTTP 프로토콜인 경우, 중간 패킷 캡쳐 프로그램에서 전달되는 메시지를 다 확인해볼 수 있지만, HTTPS 프토토콜로 접속하는 경우에는 패킷 캡쳐 프로그램에서도 Encrypted Application Data 라고 나와서 내부의 메시지가 어떤 메시지인지 유추해볼 수 없도록 구성되어 있습니다.

ThingPlug 에서는 최신 암호화 방식을 따르는 TLS 1.2 버전을 지원하여 ThingPlug 을 통해 IoT 어플리케이션이 서비스 될 때, 발생하는 다양한 데이터를 보다 안전하게 지킬 수 있도록 지원하고 있습니다.

단, 보안 채널 통신을 사용하지 않아도 되는 보안에 민감하지 않는 어플리케이션에서는 굳이 보안 채널 통신을 통해 접속하지 않아도 되기 때문에 일반 HTTP, MQTT 프로토콜도 ThingPlug 에서는 지원합니다. 하지만, HTTP, MQTT 를 사용할 경우 전달되는 채널에서 보안 이슈가 발생할 수 있음을 꼭 확인해 주세요!

[MQTT 인 경우]



No.	Time	Source	Destination	Protocol	Length	Info
25	23.715646	211.234.246.71	150.23.23.152	MQTT	60	Publish Ack
26	23.716976	211.234.246.71	150.23.23.152	MQTT	68	Publish Message
27	23.717055	150.23.23.152	211.234.246.71	TCP	54	54486 → 1883 [ACK] Seq=102 Ack=61 Win=258 Len=0
28	23.734903	150.23.23.152	211.234.246.71	MQTT	58	Publish Ack
29	23.782672	211.234.246.71	150.23.23.152	TCP	60	1883 → 54486 [ACK] Seq=61 Ack=106 Win=123 Len=0
30	83.734264	150.23.23.152	211.234.246.71	MQTT	56	Ping Request

```
▷ Frame 26: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▷ Ethernet II, Src: CiscoInc_2c:98:00 (00:13:5f:2c:98:00), Dst: LcfcHefe_65:bb:92 (68:f7:28:65:bb:92)
▷ Internet Protocol Version 4, Src: 211.234.246.71, Dst: 150.23.23.152
▷ Transmission Control Protocol, Src Port: 1883 (1883), Dst Port: 54486 (54486), Seq: 47, Ack: 102, Len: 14
◢ MQ Telemetry Transport Protocol
  ◢ Publish Message
    ▷ 0011 0010 = Header Flags: 0x32 (Publish Message)
      Msg Len: 12
      Topic: test
      Message Identifier: 3
      Message: test
```

[MQTTS 인 경우]

No.	Time	Source	Destination	Protocol	Length	Info
51	198.879685	211.234.246.71	150.23.23.152	TCP	123	8883 → 54506 [PSH, ACK] Seq=6583 Ack=1146 Win=16768 Len=69
52	198.879747	150.23.23.152	211.234.246.71	TCP	54	54506 → 8883 [ACK] Seq=1146 Ack=6652 Win=65024 Len=0
53	198.880861	150.23.23.152	211.234.246.71	TCP	123	54506 → 8883 [PSH, ACK] Seq=1146 Ack=6652 Win=65024 Len=69
54	198.923923	211.234.246.71	150.23.23.152	TCP	60	8883 → 54506 [ACK] Seq=6652 Ack=1215 Win=16768 Len=0
55	222.363122	150.23.23.152	211.234.246.71	TCP	123	54506 → 8883 [PSH, ACK] Seq=1215 Ack=6652 Win=65024 Len=69
56	222.367152	211.234.246.71	150.23.23.152	TCP	60	8883 → 54506 [ACK] Seq=6652 Ack=1284 Win=16768 Len=0
57	222.367699	211.234.246.71	150.23.23.152	TCP	123	8883 → 54506 [PSH, ACK] Seq=6652 Ack=1284 Win=16768 Len=69
58	222.572403	150.23.23.152	211.234.246.71	TCP	54	54506 → 8883 [ACK] Seq=1284 Ack=6721 Win=65024 Len=0
59	224.665681	150.23.23.152	211.234.246.71	TCP	123	54506 → 8883 [PSH, ACK] Seq=1284 Ack=6721 Win=65024 Len=69
60	224.670600	211.234.246.71	150.23.23.152	TCP	123	8883 → 54506 [PSH, ACK] Seq=6721 Ack=1353 Win=16768 Len=69
61	224.865447	150.23.23.152	211.234.246.71	TCP	54	54506 → 8883 [ACK] Seq=1353 Ack=6790 Win=64768 Len=0
62	228.089491	150.23.23.152	211.234.246.71	TCP	123	54506 → 8883 [PSH, ACK] Seq=1353 Ack=6790 Win=64768 Len=69
63	228.093963	211.234.246.71	150.23.23.152	TCP	123	8883 → 54506 [PSH, ACK] Seq=6790 Ack=1422 Win=16768 Len=69
64	228.095387	211.234.246.71	150.23.23.152	TCP	123	8883 → 54506 [PSH, ACK] Seq=6859 Ack=1422 Win=16768 Len=69
65	228.095450	150.23.23.152	211.234.246.71	TCP	54	54506 → 8883 [ACK] Seq=1422 Ack=6928 Win=66048 Len=0
66	228.118978	150.23.23.152	211.234.246.71	TCP	123	54506 → 8883 [PSH, ACK] Seq=1422 Ack=6928 Win=66048 Len=69
67	228.172691	211.234.246.71	150.23.23.152	TCP	60	8883 → 54506 [ACK] Seq=6928 Ack=1491 Win=16768 Len=0

```
▷ Frame 66: 123 bytes on wire (984 bits), 123 bytes captured (984 bits) on interface 0
▷ Ethernet II, Src: LcfcHefe_65:bb:92 (68:f7:28:65:bb:92), Dst: All-HSRP-routers_71 (00:00:0c:07:ac:71)
▷ Internet Protocol Version 4, Src: 150.23.23.152, Dst: 211.234.246.71
▷ Transmission Control Protocol, Src Port: 54506 (54506), Dst Port: 8883 (8883), Seq: 1422, Ack: 6928, Len: 69
◢ Data (69 bytes)
    Data: 17030300400c0e3056e0be71a518422d0cac091e7110b4ab...
    [Length: 69]
```

6 LoRa 기반 IoT 애플리케이션 개발을 위한 ThingPlug API

본 장에서는 LoRa 서비스 개발을 위해 제공되는 ThingPlug API 를 정의하며, 각 API 들을 접근하기 위해 정의된 MQTT 및 HTTP 프로토콜 메시지의 예를 제시합니다. 좀더 상세하게는 LoRa 디바이스가 네트워크에 등록(Join)된 후, 주기보고 메시지 저장과 제어를 위한 메시지 Request/Response 메시지의 예를 명시합니다.

6.1 LoRa 기반 IoT 애플리케이션을 위한 Resource API 구성

일반적으로 RESTful API 의 아키텍쳐를 따르는 ThingPlug API 각 Resource API 가 부모/자식 관계를 가지는 Hierarchical 구조로 구성됩니다.

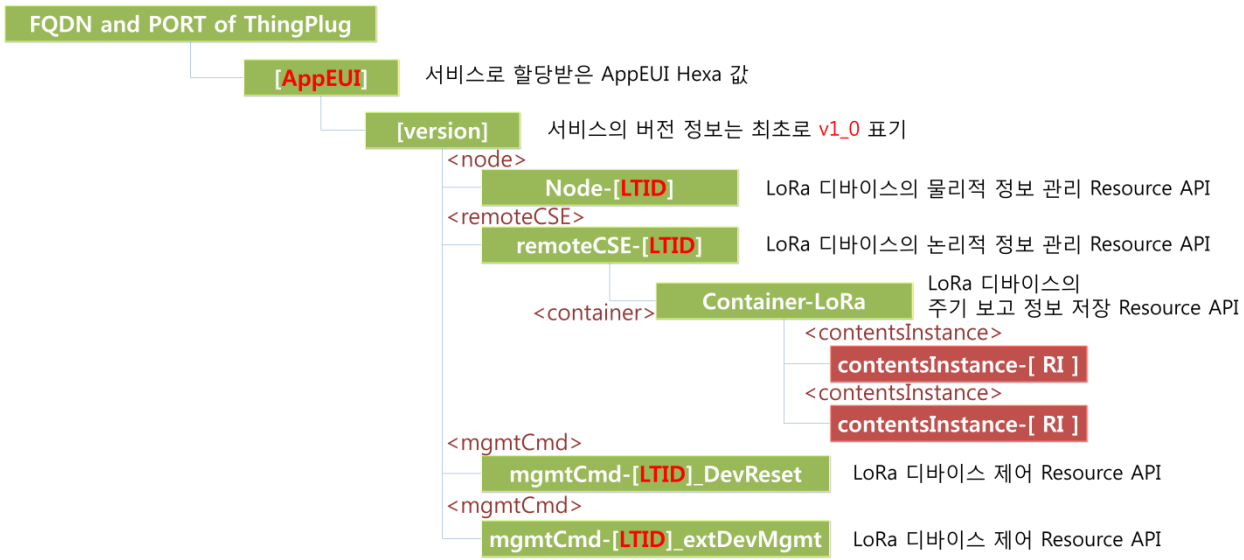


그림 5 하나의 LoRa 디바이스가 정상적으로 개통되었을 때, 생성되는 Resource API 구조

상기 구성도는 LoRa 네트워크에 존재하는 특정 LoRa 디바이스가 정상적으로 개통되었을 때, ThingPlug 서버에 저장되는 Resource API 의 구조를 나타냅니다. 상기 그림에서 보면, 녹색으로 표현되는 부분은 개통된 후에 생성되는 Resource API 를 적색으로 표현된 부분은 개통 후, LoRa 디바이스가 주기보고를 생성하여 네트워크에 전달하면 생성되는 Resource API 를 표현합니다. 만약에 LoRa 디바이스가 10 번의 주기보고가 네트워크로 전달되면 적색 Resource API 가 10 개 생성되며, Application 은 해당 API 에 접근하여 생성된 주기보고 정보를 파악할 수 있습니다.

상기 그림에서 박스 위에 표현된 〈 〉 표시는 Resource API Type 을 명칭 합니다. 좀더 상세한 내용은 아래 박스 내용을 참고 부탁 드립니다.

LoRa 어플리케이션 개발을 위해서 SK 텔레콤에서 제공하는 Resource API Type 은 다음과 같습니다.

순번	Resource API Type	설명
1	〈node〉	LoRa 디바이스의 물리적 정보를 저장하는 Resource API
2	〈remoteCSE〉	LoRa 디바이스의 논리적 정보(주기 정보 등)를 저장하는 Resource API
3	〈container〉	LoRa 디바이스의 주기 보고가 저장되는 저장소 Resource API

4	〈contentInstance〉	실제 주기 보고 정보가 저장되는 Resource API
5	〈mgmtCmd〉	LoRa 디바이스 제어 Resource API
6	〈execInstance〉	LoRa 디바이스 제어 결과 저장되는 Resource API
7	〈subscription〉	LoRa 디바이스 정보의 변화를 Notification 받을 수 있게 설정하는 Resource API

6.2 Resource API 의 공통 속성 (Attribute)

6.2.1 Resource API 공통 속성

ThingPlug 에 정의된 모든 Resource API 에서 사용되는 공통 속성(Attribute) 값이 존재하며, 해당 속성에 대한 설명은 아래 표와 같으며 좀 더 상세한 내용은 [OFAS1.12], [TADLN0.5]를 참고합니다. 표준에 명시된 속성 이름은 Short 과 Long Name 타입이 있으며, ThingPlug 에서는 Short Name 타입을 사용합니다.

속성 이름		의미	사용 메시지	
Short	Long		Request	Response
op	Operation	CRUD Operation를 Enum으로 표기 1: CREATE 2: RETRIEVE 3: UPDATE 4: DELETE 5: NOTIFY	M	-
to	To	Request 메시지를 적용하고자 하는 타겟 주소	M	-
fr	From	Request 메시지의 발신자 식별자 * HTTP Binding : X-M2M-Origin	M	-
ty	Type	Resource Type을 Enum으로 표기 3: container 4: contentInstance 7: execInstance 12: mgmtCmd 14: node 16: remoteCSE	O (Create 시에만)	M
ri (Req Header)	Request Identifier	발신한 Request 메시지의 고유 식별자로 사용된다. 생성 방법은 {fr}_{sequential number} 형태로 구성 (개발사에서 자유롭게 설정 가능) [예] fr이 00001111 이며, 해당 발신자로부터 1004번째 메시지 인 경우 → 00001111_1004) * HTTP Binding : X-M2M-RI	M	M
nm	Name	생성되는 Resource의 이름 입력한 nm 값은 함께 입력된 ty 값과 매핑되어 최종 생성된 Resource은 ty의 Resource 이름과 입력한 nm의 조합으로 구성 [예] ty=3, nm=test인 경우 → container-test가	O (Create 시에만)	-

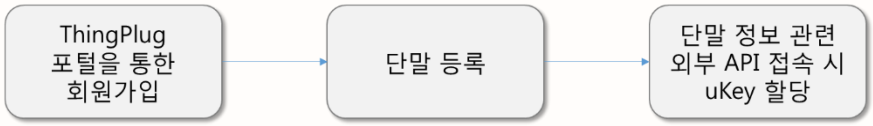
		최종 결과		
cty	Content Type	Content-Type의 약자로 Payload의 내용의 표현 (Representation 타입을 명시) XML : application/vnd.onem2m-prsp+xml JSON : application/vnd.onem2m-prsp+json	M	M
uKey	uKey	ThingPlug의 Northbound (ThingPlug API to Application Server) API 접근 권한을 위한 Access Token으로 해당 uKeys ThingPlug 홈페이지의 회원가입 시 발급 ※ 다음 페이지 Tip 에서 uKey 할당 방법 소개	O (특정 API에만)	O (특정 API에만)
rsc	Response Status Code	요청된 호출에 대한 ThingPlug 응답 코드를 의미한다. 상세한 내용은 A.2 (Annex 2)를 참고	-	M
ri (Resp Payload)	Resource Identifier	Resource Identifier의 약자로 생성된 Resource의 고유 식별자	-	M
rn	Resource Name	생성된 Resource의 이름	-	M
ct	Created Time	생성된 Resource의 Created Time	-	M
lt	Last Modified Time	생성된 Resource의 Last Modified Time	-	M
et	Expiration Time	생성된 Resource의 유효기간 (Expiration Time)	-	O (데이터만 유효)
lbl	Label	생성된 Resource의 태그 정보	-	M

※ 필수 여부 : M(Mandatory), O(Optional)

ThingPlug Tip! uKey 할당 및 확인 방법

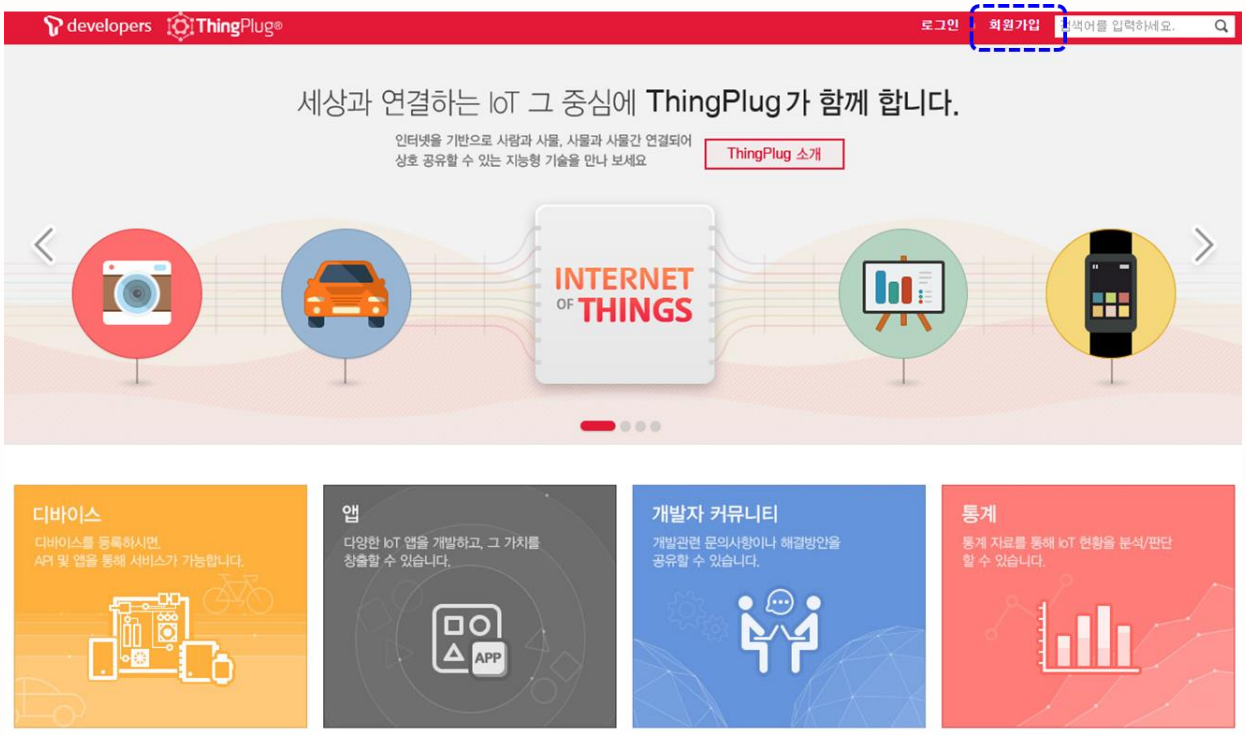
uKey의 개념은 간단하게 외부에서 ThingPlug REST API 접근 시 정상적인 접근인지 아닌지 아닌지를 판단하는 Access Token Key의 개념과 동일합니다.

예를 들어, A라는 서비스에 B,C 단말이 등록이 되어있다고 가정하면, 해당 단말은 기본적으로 A 서비스 Application의 접근 시에만 B,C 단말에 대한 API 접근을 허하여야 합니다. 그렇지 않는다면, 보안 이슈가 발생될 수 있습니다. 이를 방지 하기 위해서는 ThingPlug에서는 아래 기술된 절차를 기반으로 가입자에게 uKey를 할당하고, uKey를 통해 접속할 수 있는 디바이스를 특정하기 위해서 포털을 통한 단말 등록 절차를 진행하게 됩니다.

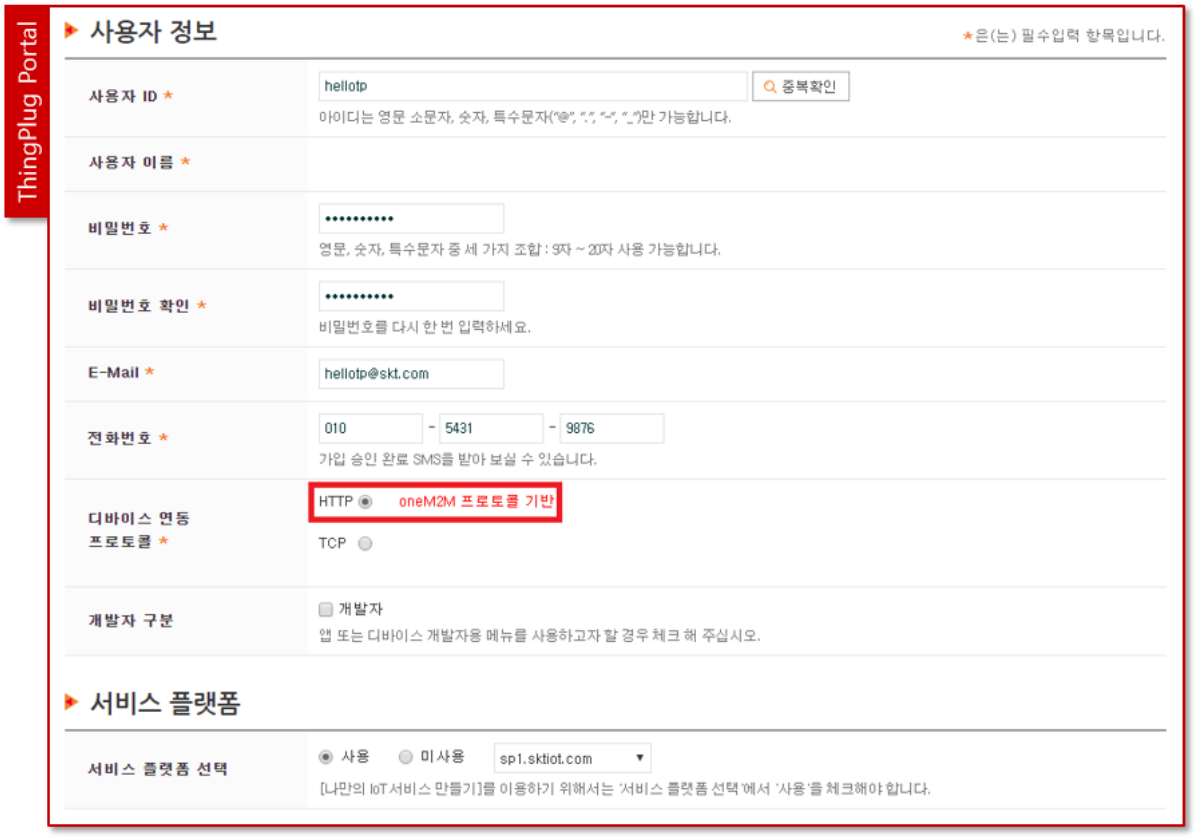


[ThingPlug 포털을 통한 회원가입]

1. https://thingplug.sktiot.com 접속



2. 포털 회원 가입 및 프로토콜 선택



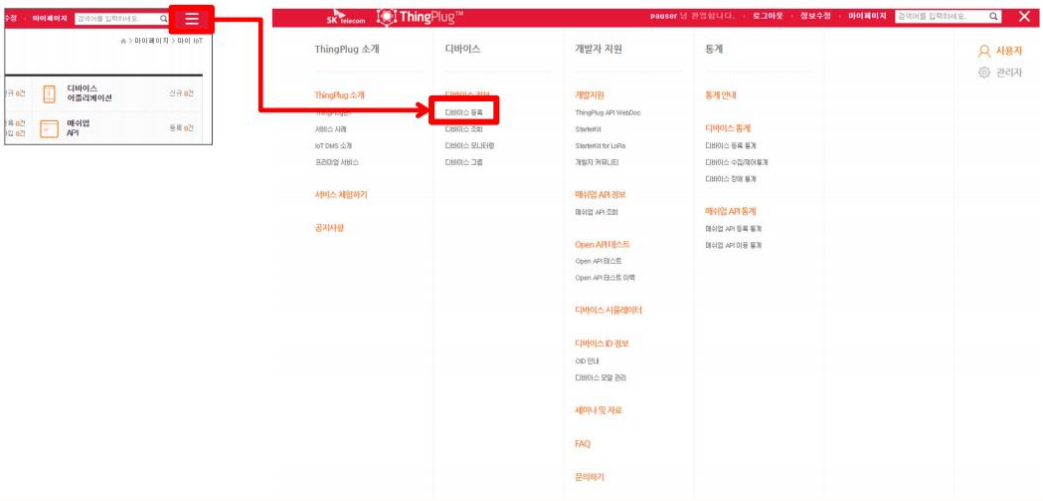
3. 우측 상단 마이페이지 – uKey(사용자 인증키) 확인

[단말 등록]

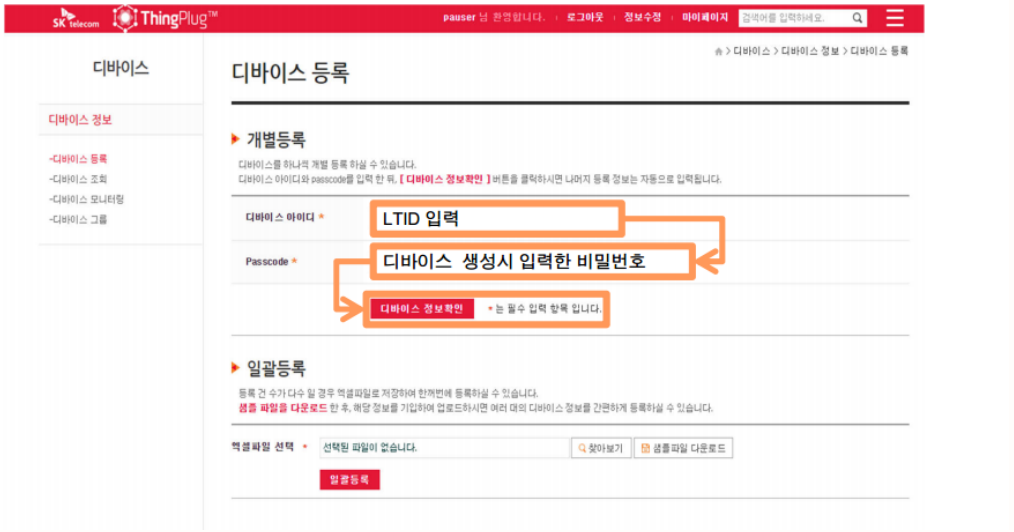
단말 등록 시에는 이전에 단말이 정상적으로 ThingPlug 포털에 〈remoteCSE〉 자원의 등록이 정상적으로 완료되어야 합니다. 포털에서 단말을 식별하는 방법은 단말의 아이디 및 PassCode 속성입니다. 만약 2가지 정보를 정확하게 모르고 있다면, 정상적으로 단말을 등록 및 해당 단말의 uKey 연동을 수행할 수 없습니다.

1. 단말 등록

- 로그인 – 우측상단 메뉴 – 디바이스 정보 – 디바이스 등록



- 디바이스 아이디(LTID), Passcode(remoteCSE 생성때 입력한 값) 입력 – 디바이스 정보확인



- 디바이스 등록 – 위치정보, 디바이스 이미지 입력 – 저장 버튼 클릭

[단말 정보 관련 외부 API 접속 시 uKey 할당]

정상적으로 단말이 등록된 경우에는 아래 그림의 하단부와 같이 등록된 단말 리스트가 보이고, 해당 단말에 한하여 그림 중간에 있는 uKey를 통한 API 접근이 가능합니다.

6.3 LoRa 디바이스 물리적 정보를 파악하는 Resource

6.3.1 〈node〉 Resource – Retrieve

〈node〉 Resource는 LoRa 디바이스의 물리적 정보를 저장하는 Resource API입니다. 해당 Resource를 Retrieve 함으로써 LoRa 디바이스의 정보(식별자 정보)를 파악할 수 있습니다.

속성 이름		의미	필수 여부	비고
Short	Long			
ni	Node Identifier	LoRa 디바이스의 식별자를 의미하는 값으로 본 연동에서는 각 디바이스의 LTID를 사용	M	
mga	Mgmt. Address	LoRa 디바이스 제어를 위해 호출해야 하는 디바이스 제어 어플리케이션의 주소	M	
ppt	Property	추가 별도 속성 정보 (Reserved)	O	

6.3.1.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request

```
GET /0000000000000001/v1_0/node-000000010000000020160431 HTTP/1.1
Host: 211.115.15.160:9000
Accept: application/xml
X-M2M-RI: 0000000000000001_000001
X-M2M-Origin: 0000000000000001
uKey:
MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCtGK2RDVDhCUA==
```

Response

```
200 OK
Content-Length →383
Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8
Date →Sun, 22 May 2016 13:31:51 GMT
Server →Apache-Coyote/1.1
X-M2M-RI →0000000000000001_000001
X-M2M-RSC →2000
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:nod xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ty>14</ty>
    <ri>ND00000000000000000359</ri>
    <rn>000000010000000020160431</rn>
    <pi>CB00000000000000000031</pi>
    <ct>2016-05-22T22:23:01+09:00</ct>
    <lt>2016-05-22T22:23:01+09:00</lt>
    <ni>000000010000000020160431</ni>
    <mga>HTTP|http://my.address.com:xxxx</mga>
</m2m:nod>
```

6.3.1.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)
-

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">
    <op>2</op>
    <to>/0000000000000001/v1_0</to>
    <fr>000000010000000020160431</fr>
    <ri>0000000000000001_000001</ri>
    <cty>application/vnd.onem2m-prsp+xml</cty>
    <uKey>MlBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRTh
    oa3FSaCtGK2RDVDhCUA==</uKey>
</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<ri>0000000000000001_000001</ri>
<rsc>2000</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>
```

```
<pc>
    <nod>
        <ty>14</ty>
        <ri>ND000000000000000000359</ri>
        <rn>000000010000000020160431</rn>
        <pi>CB00000000000000000031</pi>
        <ct>2016-05-22T22:23:01+09:00</ct>
        <lt>2016-05-22T22:23:01+09:00</lt>
        <ni>000000010000000020160431</ni>
        <mga>MQTT|thingplugtest_0001</mga>
</nod>
</pc>
</m2m:rsp>
```

6.4 LoRa 디바이스 주기 정보 저장하는 Resource APIs

6.4.1 <remoteCSE> - Retrieve

<remoteCSE> Resource 는 LoRa 디바이스의 주기 보고 등과 같은 논리적 정보를 저장하는 Root Resource 로 하기 해당 Resource 의 정보를 획득한다.

속성 이름		의미	필수 여부	비고
Short	Long			
cst	CSE Type	CSE Type으로 enum으로 표현된다. 1: Server 2: Gateway 3: Device	M	
csi	CSE-ID	CSE 식별자로 LoRa 디바이스의 LTID를 지정	M	
poa	Point of Access	호출이 가능한 LoRa 디바이스 어플리케이션 주소	M	
rr	Request Reachability	MQTT 프로토콜에서는 'True' 설정한다. 또는 HTTP 프로토콜인데, 수신 받는 객체의 IP가 고정 아이피 인 경우에도 'True'로 설정하고, 나머지 경우는 'False'로 표현	M	
nl	Node Link	논리적 정보를 포함하는 실제 물리적 LoRa 디바이스 Resource인 <node> Resource의 Resource 식별자	M	
ppt	Property	추가 별도 속성 정보 (Reserved)	O	

ThingPlug Tip! dKey 소개

dKey는 단말에서 ThingPlug REST API를 접근 하기 위해 사용되는 Key입니다.
단말의 논리 정보를 등록하는 remoteCSE가 생성된 경우 response로 dKey가 전달이 되며, 해당 값을 이용해 container, contentInstance 등의 자원 생성을 할 수 있게 됩니다.

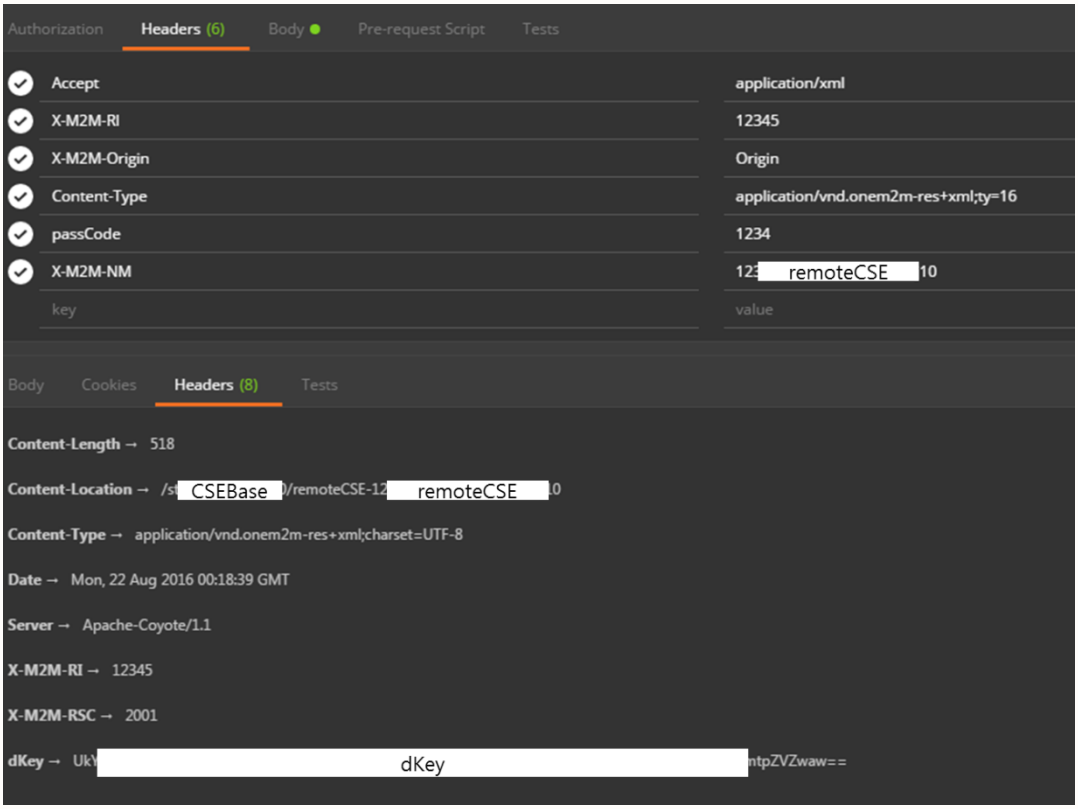


그림 6 remoteCSE 자원 생성 후 반환된 dKey (POSTMAN)

[dKey vs uKey]

dKey는 단말이 **새로운 자원의 생성(Create)** 또는 **기존 자원의 변경(Update)**을 위해 사용하게 됩니다. 예를 들자면, 센서디바이스가 센서값을 ThingPlug에 올리고자 하는 경우 contentInstance를 생성할때 dKey를 속성으로 넣어주어야합니다.

반대로 **uKey**는 어플리케이션에서 ThingPlug에 생성된 자원의 정보를 확인 또는 단말 제어를 하고자 할 때 사용하게 됩니다. 예를 들어, 올라온 최신 센서값을 확인 **([GET] latest)** 하고자 하는 경우, 또는 제어명령을 전달 **([PUT] mgmtCmd)** 하기 위해서는 uKey를 속성값에 넣어주어야 합니다.

[dKey vs passcode]

앞서 말씀드렸듯, **dKey**는 API call을 위한 속성으로 사용이 됩니다.

passCode는 포탈에서 단말 등록 절차에서 필요한 Key이며, 해당 단말의 자원이 올바로 생성되었는지 확인하기 위한 인증키로 사용됩니다. 앞서 나온 'uKey 및 단말등록 절차'에 소개되어 있습니다.

[dKey는 LoRa에서 언제 사용되나요?]

dKey는 ThingPlug의 Southbound API 호출을 위해 사용되는 Key입니다. LoRa에서는 N/W Server에서 ThingPlug API를 호출할 때 사용되며, 단말에서 직접 dKey를 Payload에 추가하여 사용하지는 않습니다.

그림 7 포탈에서 단말 등록시 사용되는 passCode

6.4.1.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 00000001000000020160431

Request
GET /0000000000000001/v1_0/remoteCSE-00000001000000020160431 HTTP/1.1 Host: 211.115.15.160:9000 Accept: application/xml X-M2M-RI: 0000000000000001_000001 X-M2M-Origin: 0000000000000001 uKey: MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCt GK2RDVDhCUA==

Response
200 OK Content-Length →517 Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8 Date →Mon, 23 May 2016 10:43:34 GMT Server →Apache-Coyote/1.1 X-M2M-RI →0000000000000001_000001 X-M2M-RSC →2000 〈?xml version="1.0" encoding="UTF-8" standalone="yes"?〉 〈m2m:csr xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"〉 〈ty〉16〈/ty〉 〈ri〉RC0000000000000000350〈/ri〉 〈rn〉00000001000000020160431〈/rn〉

```
    <pi>CB00000000000000000031</pi>
    <ct>2016-05-22T22:48:13+09:00</ct>
    <lt>2016-05-22T22:48:13+09:00</lt>
    <acpi>AP00000000000000000497</acpi>
    <cst>1</cst>
    <csi>00000001000000020160431 </csi>
    <rr>true</rr>
    <nl>ND000000000000000000359</nl>
</m2m:csr>
```

6.4.1.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 00000001000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">
    <op>2</op>
    <to>/0000000000000001/v1_0/remoteCSE-00000001000000020160431</to>
    <fr>00000001000000020160431</fr>
    <ri>0000000000000001_000001</ri>
    <cty>application/vnd.onem2m-prsp+xml</cty>
    <uKey>MlBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRTh
    oa3FSaCtGK2RDVDhCUA==</uKey>
</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<ri>0000000000000001_000001</ri>
<rsc>2000</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>

<pc>
  <csr>
    <ty>16</ty>
    <ri>RC000000000000000000350</ri>
```

```
    <rn>00000001000000000020160431</rn>
    <pi>CB00000000000000000031</pi>
    <ct>2016-05-22T22:48:13+09:00</ct>
    <lt>2016-05-22T22:48:13+09:00</lt>
    <acpi>AP00000000000000000497</acpi>
    <cst>1</cst>
    <csi>00000001000000000020160431 </csi>
    <rr>true</rr>
    <nl>ND00000000000000000359</nl>
  </csr>
</pc>
</m2m:rsp>
```

6.4.2 <container> - Retrieve

<container> Resource 는 LoRa 디바이스의 주기보고를 저장하는 저장소 Resource 입니다. 표준 관련된 내용은 아래 표를 참고 부탁 드립니다.

속성 이름		의미	필수 여부	비고
Short	Long			
st	Statetag	해당 저장소에 저장된 Data가 있거나, 수정된 경우에 누적으로 1씩 증가한다.	M	
cr	Creator	해당 저장소에 Data를 저장하는 발신자의 식별자	M	
cni	Current Number of Instances	해당 Resource에 하부에 추가되는 실제 Data 인스턴스의 개수	M	
cbs	Current Byte Size	해당 Resource에 하부에 저장된 데이터의 사이즈	M	
ppt	Property	추가 별도 속성 정보	O	

6.4.2.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 00000001000000000020160431

Request

```
GET /0000000000000001/v1_0/remoteCSE-00000001000000000020160431/container-LoRa
HTTP/1.1
Host: 211.115.15.160:9000
Accept: application/xml
X-M2M-RI: 0000000000000001_000001
X-M2M-Origin: 0000000000000001
uKey:
MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCt
GK2RDVDhCUA==
```

Response

```
200 OK
Content-Length →434
Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8
Date →Sun, 22 May 2016 14:11:41 GMT
Server →Apache-Coyote/1.1
X-M2M-RI →0000000000000001_000001
X-M2M-RSC →2000

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ty>3</ty>
    <ri>CT0000000000000000533</ri>
    <rn>LoRa</rn>
    <pi>RC0000000000000000350</pi>
    <ct>2016-05-22T22:59:32+09:00</ct>
    <lt>2016-05-22T22:59:32+09:00</lt>
    <st>0</st>
    <cr>RC0000000000000000350</cr>
    <cni>0</cni>
    <cbs>0</cbs>
</m2m:cnt>
```

6.4.2.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)
-

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">
    <op>2</op>
    <to>/0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-
        LoRa</to>
    <fr>000000010000000020160431</fr>
    <ri>0000000000000001_000001</ri>
    <cty>application/vnd.onem2m-prsp+xml</cty>
    <uKey>MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRTh
    oa3FSaCtGK2RDVDhCUA==</uKey>
</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<ri>0000000000000001_000001</ri>
<rsc>2000</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>

<pc>
  <cnt>
    <ty>3</ty>
    <ri>CT00000000000000000533</ri>
    <rn>LoRa</rn>
    <pi>RC00000000000000000350</pi>
    <ct>2016-05-22T22:59:32+09:00</ct>
    <lt>2016-05-22T22:59:32+09:00</lt>
    <st>0</st>
    <cr>RC00000000000000000350</cr>
    <cni>0</cni>
    <cbs>0</cbs>
  </cnt>
</pc>
</m2m:rsp>
```

oneM2M Tip! 〈container〉 Resource API

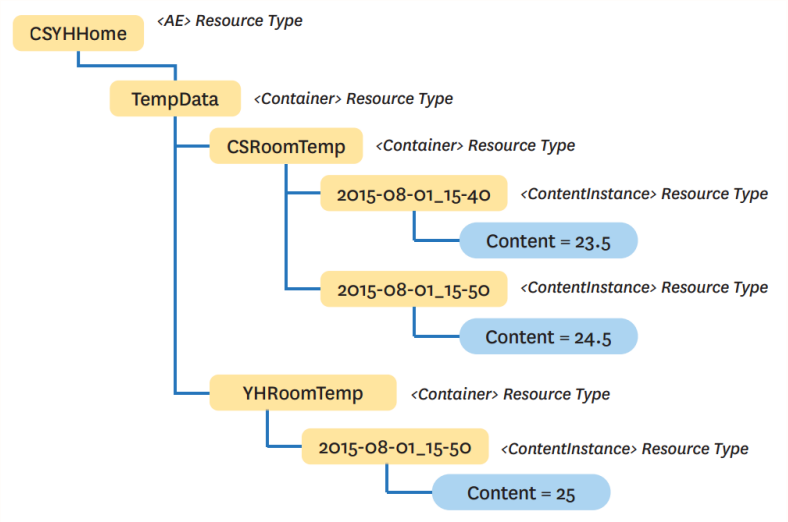
ThingPlug가 제공해야 할 가장 중요한 기능 중 하나는 IoT 장치의 센서가 획득한 정보 또는 액츄에이터의 동작 상황을 데이터로 변환한 후 이를 저장하고 해당 데이터를 앱이 조회 또는 수정할 수 있도록 하는 것입니다.

예를 들어, 스마트 홈 서비스에서 세탁기는 '현재 세탁 중'이라는 자신의 상태 정보를 서버 플랫폼에 저장하고 공유하여 자신의 상태를 알려주거나 외부 애플리케이션에서 상태를 변경할 수 있도록 구성될 수 있습니다. 이는 자원의 상태를 전달하면 특정 동작을 유도하고자 하는 REST 시스템 동작의 특징과도 일맥상통합니다.

oneM2M에서는 발생한 데이터를 저장하고 공유하는 데 있어서 다음 표에 나오는 바와 같이 〈container〉와 〈contentInstance〉 자원 타입을 호출하여 사용합니다. 두 자원은 우리가 사용하는 OS에서 폴더와 파일의 관계와도 같습니다. 우리가 문서를 작업한 다음에 저장할 때는 우선 적합한 폴더를 만든 후에 문서를 폴더에 넣습니다. 마찬가지로 oneM2M에서도 〈container〉 자원 타입을 통해서 정보의 구조적인 의미를 표현하고, 실제 애플리케이션AE 는 상태 정보는 〈contentInstance〉 자원 타입을 통해서 저장하도록 설계되어 있습니다.

예를 들어, 철수의 방과 영희의 방에 각각 온도 센서가 있다고 가정해보겠습니다. 각 센서 AE가 각 정보를 ThingPlug 서버에 저장한다고 하면, 서버에 CSYHHome 애플리케이션을 〈AE〉 자원으로 등록하고, 해당 자원을 Root로 다음 디렉토리 구조와 같이 철수의 방의 온도(CSRoomTemp)와 영희의 방 온도(YHRoomTemp)를 생성합니다. 그런 다음에 실제 데이터를 〈contentInstance〉 자원의 'content' 속성에 저장하면, 상기 〈container〉 자원 안에 실제 데이터들이 저장됩니다.

예를 들어, 어떤 사람이 외부에서 철수 방의 온도를 알고 싶다고 하면, https://sandbox.sktiot.com/CSYHHome/TempData/CSRoomTemp/2015-08-01_15-40을 웹 브라우저를 통해서 호출하면 (접근 권한이 있는 경우) 다음과 같은 응답을 받게 될 것입니다. 그런 후에 사용자는 'content' 속성 안에 있는 값만을 획득한 후에 이를 다양한 애플리케이션에 활용하면 됩니다.



6.4.3 〈latest〉 – Retrieve

〈latest〉 Resource 는 Virtual Resource 로 LoRa 디바이스의 주기보고로 저장된 값 중에서 가장 최신에 저장된 값을 호출하는 API 입니다.

속성 이름		의미	필수 여부	비고
Short	Long			
cnf	Content Information	해당 데이터의 타입을 명시한다. 본 연동에서는 'LoRa/Sensor'로 명시한다.	M	
con	content	실제 주기 보고 데이터 * LoRa 단말인 경우에는 hexa string 형태로 encoding 되어서 전달됩니다.	M	
et	Expiration Time	데이터를 보관하는 시간 (정책에 따라 일괄 반영됨)	M	
cr	Creator	해당 저장소에 Data를 저장하는 발신자의 식별자	M	
cs	Content Size	Content Size	M	

6.4.3.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request
GET /0000000000000001/v1_0/remoteCSE-0000000100000000020160431/container-LoRa/latest HTTP/1.1 Host: 211.115.15.160:9000 Accept: application/xml X-M2M-RI: 0000000000000001_000001 X-M2M-Origin: 0000000000000001 uKey: MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCtGK2RDVDhCUA==

Response
200 OK Content-Length →536 Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8 Date →Mon, 23 May 2016 10:02:36 GMT Server →Apache-Coyote/1.1 X-M2M-RI →0000000000000001_000001 X-M2M-RSC →2000 〈?xml version="1.0" encoding="UTF-8" standalone="yes"?〉 〈m2m:cin xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"〉 〈ty〉4〈/ty〉 〈ri〉CI00000000000002984968〈/ri〉 〈rn〉CI00000000000002984968〈/rn〉 〈pi〉CT00000000000000000533〈/pi〉 〈ct〉2016-05-23T19:02:32+09:00〈/ct〉 〈lt〉2016-05-23T19:02:32+09:00〈/lt〉 〈et〉9999-12-31T00:00:00+00:00〈/et〉 〈st〉1〈/st〉 〈cr〉RC00000000000000000350〈/cr〉 〈cnf〉 LoRa/Sensor〈/cnf〉 〈cs〉70〈/cs〉 〈con〉012391392391239123912932932931929312931929319239129319 2391293923123123〈/con〉 〈/m2m:cin〉

6.4.3.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 0000000100000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request에 대한 Response)
-

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">
    <op>2</op>
    <to>/0000000000000001/v1_0/remoteCSE-0000000100000000020160431
        /container-LoRa/latest</to>
    <fr>0000000100000000020160431</fr>
    <ri>0000000000000001_000001</ri>
    <cty>application/vnd.onem2m-prsp+xml</cty>
     <uKey>MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRT
     hoa
     3FSaCtGK2RDVDhCUA==</uKey>
</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<ri>0000000000000001_000001</ri>
<rsc>2000</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>

<pc>
  <cin>
    <ty>4</ty>
    <ri>CI000000000000002984968</ri>
    <rn>CI000000000000002984968</rn>
    <pi>CT00000000000000000533</pi>
    <ct>2016-05-23T19:02:32+09:00</ct>
    <lt>2016-05-23T19:02:32+09:00</lt>
    <et>9999-12-31T00:00:00+00:00</et>
    <st>1</st>
    <cr>RC00000000000000000350</cr>
    <cnf> LoRa/Sensor </cnf>
    <cs>70</cs>
     <con>012391392391239123912932932931929312931929319239129319239123
     93923123123</con>
  </cin>
</pc>
</m2m:rsp>
```

6.5 LoRa 디바이스 제어를 위한 Resource

6.5.1 〈mgmtCmd〉 Resource Update (제어 실행)

그림 8 에 명시된 것과 같이 하나의 LoRa Device 가 네트워크에 정상적으로 등록된 경우에 장치 제어를 위한 기본 API 는 서버에 자동으로 등록됩니다.

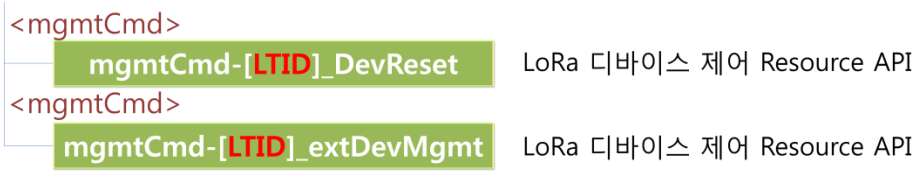


그림 8 LoRa 디바이스 제어를 위한 Resource API 구조

상기 그림 8 에 명시된 Resource 에 Update 동작을 수행함으로써 LoRa 디바이스 제어 명령어가 LoRa NW 서버를 통해 LoRa 디바이스로 전달됩니다. LoRa 서비스에서 사용하는 Resource API 는 아래 표와 같습니다.

Resource API Type	제어 설명
mgmtCmd-[LTID]_DevReset	디바이스 리셋
mgmtCmd-[LTID]_extDevMgmt	서비스에 따라 Customer Server와 디바이스 제조사에서 정의하여 사용 (Application Specific Device Management)

<table>
<tr><th colspan="2">속성 이름</th><th rowspan="2">의미</th><th rowspan="2">필수 여부</th><th rowspan="2">비고</th></tr>
<tr><th>Short</th><th>Long</th></tr>
<tr><td>cmt</td><td>Command Type</td><td>디바이스 제어 타입
(예, 디바이스 리셋, 주기 변경 등)</td><td>O</td><td></td></tr>
<tr><td>exe</td><td>Execute Enable</td><td>장치 제어 실행 상태
기본 값은 False에서 실행을 요청했을 때, True로 세팅합니다.</td><td>M</td><td></td></tr>
<tr><td>ext</td><td>Execute Target</td><td>해당 장치 제어가 실행되는 대상 장치을 식별하기 위해서 물리적 장치 정보를 포함하는 〈node〉 자원의 Resource 식별자를 입력합니다.</td><td>O</td><td></td></tr>
<tr><td>exra</td><td>Execute Request Argument</td><td>장치 제어를 위해 필요한 Argument를 포함합니다. 아래 값은 hexa string(소문자)로 기입합니다.

아래 DevReset인 경우에 〈exra〉 속성에 값이 들어가는 경우에는 에러를 리턴합니다.
<table>
<tr><th>Resource API</th><th>Value</th></tr>
<tr><td>DevReset</td><td>N/A</td></tr>
<tr><td>extDevMgmt</td><td>서비스 제공자가 별도로 정의한 제어 명령을 추가합니다.</td></tr>
</table>
</td><td>O</td><td></td></tr>
<tr><td>ppt</td><td>Property</td><td>추가 별도 속성 정보 (Reserved)</td><td>O</td><td></td></tr>
</table>

호출하고자 하는 장치제어 API 가 Application Specific Device Management(API 명이 mgmtCmd-[LTID]-extDevMgmt)인 경우에는 API 내 exra 속성 값에 Application 에 특성에 맞는 값을 지정해주시면 됩니다. 포맷이나 형태는 제조사 또는 Application 개발사에 정의에 따라 정의하시면 됩니다.

예를 들어, LED 를 켜는 동작을 정의한다고 했을 때, 첫번째 Byte 를 제어 타입으로 명시하고, 두번째 Byte 를 제어 값으로 업체가 포맷팅 했다고 가정하고, LED 제어 타입을 01, 동작의 켬을 01, 끔을 00 으로 표현한다고 했을 때 〈exra〉 속성에 기입해야 할 값은 '0101'입니다. LED 를 끄는 동작은 '0100' 입니다.

6.5.1.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request
PUT /0000000000000001/v1_0/mgmtCmd-000000010000000020160431_DevReset HTTP/1.1 Host: 211.115.15.160:9000 Accept: application/xml X-M2M-Origin: 0000000000000001 X-M2M-RI: 0000000000000001_00001 Content-Type: application/vnd.onem2m-res+xml uKey: MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCtGK2RDVDhCUA== 〈?xml version="1.0" encoding="UTF-8"?〉 〈m2m:mgc xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"〉 〈exe〉true〈/exe〉 〈/m2m:mgc〉

Response
200 OK Content-Length →682 Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8 Date →Mon, 23 May 2016 10:41:17 GMT Server →Apache-Coyote/1.1 X-M2M-RI →0000000000000001_00001 X-M2M-RSC →2004 〈?xml version="1.0" encoding="UTF-8" standalone="yes"?〉 〈m2m:mgc xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"〉 〈ty〉12〈/ty〉

```
    <ri>MC00000000000000000332</ri>
    <rn>00000001000000020160431_DevReset</rn>
    <pi>CB00000000000000000031</pi>
    <ct>2016-05-23T19:40:31+09:00</ct>
    <lt>2016-05-23T19:41:17+09:00</lt>
    <cmt>DevReset</cmt>
    <ext>ND00000000000000000359</ext>
    <m2m:exin>
        <ty>8</ty>
        <ri>EI00000000000001608975</ri>
        <rn>EI00000000000001608975</rn>
        <pi>MC00000000000000000332</pi>
        <ct>2016-05-23T19:41:17+09:00</ct>
        <lt>2016-05-23T19:41:17+09:00</lt>
        <et>9999-12-31T00:00:00+00:00</et>
        <exs>2</exs>
        <ext>ND00000000000000000359</ext>
    </m2m:exin>
</m2m:mgc>
```

6.5.1.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 00000001000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request에 대한 Response)

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<op>3</op>
<to>/0000000000000001/v1_0/mgmtCmd-00000001000000020160431_DevReset</to>
<fr>00000001000000020160431</fr>
<ri>0000000000000001_000001</ri>
<cty>application/vnd.onem2m-prsp+xml</cty>
<uKey>MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3
FSaCtGK2RDVDhCUA==</uKey>

<pc>
  <mgc>
     <cmt>DevReset</cmt>
     <exe>true</exe>
  </mgc>
</pc>
```

```
</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<ri>0000000000000001_000001</ri>
<rsc>2004</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>

<pc>
<mgc>
<ty>12</ty>
<ri>MC00000000000000000332</ri>
<rn>00000001000000020160431_DevReset</rn>
<pi>CB00000000000000000031</pi>
<ct>2016-05-23T19:40:31+09:00</ct>
<lt>2016-05-23T20:04:29+09:00</lt>
<cmt>DevReset</cmt>
<ext>ND00000000000000000359</ext>

<exin>
    <ty>8</ty>
    <ri>EI00000000000001608982</ri>
    <rn>EI00000000000001608982</rn>
    <pi>MC00000000000000000332</pi>
    <ct>2016-05-23T20:04:29+09:00</ct>
    <lt>2016-05-23T20:04:29+09:00</lt>
    <et>9999-12-31T00:00:00+00:00</et>
    <exs>2</exs>
    <ext>ND00000000000000000359</ext>
</exin>
</mgc>
</pc>
</m2m:rsp>
```

6.5.2 Client 프로그램을 활용한 단말 제어(mgmtCmd Push Message) 예제

단말로 제어명령을 보내기 위해서는 어플리케이션에서 기존에 생성한 mgmtCmd 자원을 Update하게 됩니다

그림 9 MQTT Client(mqtt.fx)를 이용한 mgmtCmd Update 요청

자원의 Update가 정상적으로 진행이 된 경우, ThingPlug에서 단말로 제어명령이 전달되며, command Type의 exra(argument)와 ThingPlug에서 추가한 node자원 식별자, 장치제어 실행 상태 등을 포함합니다. 명령이 전달되는 단말의 주소는 node를 생성할 추가한 mga(mgmtCmd Address)로, 해당 주소로 메시지가 전달됩니다.

명령어 수신은 mqtt client에서 토픽 "/oneM2M/req_msg/{AppEUI}/{LTID}" 을 구독하거나, mga로 등록 HTTP Response를 받을 수 있는 서버가 있다면 mgmtCmd Update시 해당 메시지를 전달받을 수 있습니다. 또한 명령이 제대로 전달이 되었는지를 확인하고자 하는 경우, 다음 세션에서 설명될 execInstance-{exec RI} 자원(Update시 해당 mgmtCmd 아래 생성)을 확인하면 됩니다.

6.5.2.1 MQTT Binding Example(mgmtCmd Push Message)

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/Application Server ID (AppSVRID)/0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/[LTID] (디바이스가 명령을 전달받고자 할 때)
  - /oneM2M/resp/Application Server ID (AppSVRID)/+ (해당 Request 에 대한 Response)

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<op>3</op>
<to>/0000000000000001/v1_0/mgmtCmd-
000000010000000020160431_DevReset</to>
<fr>000000010000000020160431</fr>
<ri>000000010000000020160431</ri>
<cty>application/vnd.onem2m-prsp+xml</cty>
<uKey>MlBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFt
RThoa3FSaCtGK2RDVDhCUA==</uKey>
```

```
<pc>
<mgc>
<exra>0</exra>
<exe>true</exe>
<cmt>DevReset</cmt>
</mgc>
</pc>
</m2m:req>
```

단말로 전달된 Push Message

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<op>1</op>
<fr>00000001000000020160431</fr>
<to>00000001000000020160431</to>
<rqi>931374e1-eccc-4e14-b0a3-875a8c649d7f</rqi>
<pc>
<exin>
<ty>8</ty>
<ri>EI000000000000001165</ri>
<rn>EI000000000000001165</rn>
<pi>MC000000000000000709</pi>
<ct>2016-08-17T14:03:11+09:00</ct>
<lt>2016-08-17T14:03:11+09:00</lt>
<et>2016-09-01T14:03:11+09:00</et>
<exs>1</exs>
<ext>ND000000000000000000359</ext>
<exra>0</exra>
<cmt>DevReset</cmt>
<nm>00000001000000020160431_DevReset</nm>
</exin>
</pc></m2m:req>
```

6.5.3 〈execInstance〉 Resource Retrieve (제어 실행 결과 확인)

디바이스 제어 메시지가 전달된 후에 LoRa 디바이스가 실행된 결과를 NW 에 전달하고 제어 메시지가 설정 되어 있는 〈mgmtCmd〉 Resource API 의 자식 자원으로 등록된 〈execInstance〉 Resource API 에 제어 실행 결과를 갱신합니다. Application Server 는 저장된 제어 실행 결과를 Retrieve 하여 제어가 제대로 되었는지 또는 실패하였는지 실패하였다면, 실패의 원인이 무엇인지 표기할 수 있습니다.

그림 10 에서 [exic_RI]는 6.5.1 절의 동작으로 결과 중에서 〈exin〉 type 내부에 작성된 〈ri〉 (Resource Identifier) 값을 나타냅니다. 아래 박스에 하이라이트 된 값을 의미합니다.

```
〈exin〉
    〈ty〉8〈/ty〉
    〈ri〉EI000000000000001608982〈/ri〉
    〈rn〉EI000000000000001608982〈/rn〉
    〈pi〉MC000000000000000000332〈/pi〉
    〈ct〉2016-05-23T20:04:29+09:00〈/ct〉
    〈lt〉2016-05-23T20:04:29+09:00〈/lt〉
```

```
    〈et〉9999-12-31T00:00:00+00:00〈/et〉
    〈exs〉2〈/exs〉
    〈ext〉ND00000000000000000359〈/ext〉
〈/exin〉
```

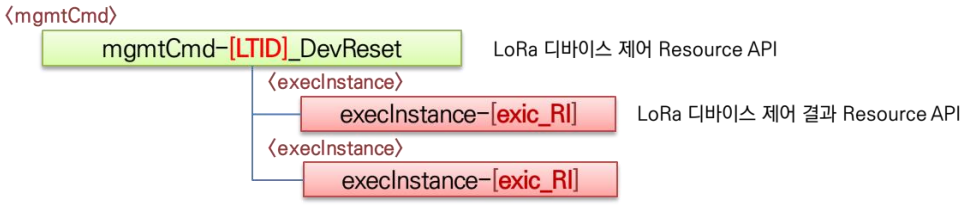


그림 10 디바이스 제어 후 생성되는 제어 결과 Resource API 구조

<table>
<tr><th colspan="2">속성 이름</th><th rowspan="2">의미</th><th rowspan="2">필수 여부</th><th rowspan="2">비고</th></tr>
<tr><th>Short</th><th>Long</th></tr>
<tr><td>cmt</td><td>Command Type</td><td>장치 제어 타입
(예, 디바이스 리셋, 펌웨어 업그레이드 등)</td><td>M</td><td></td></tr>
<tr><td>exe</td><td>Execute Enable</td><td>장치 제어 실행 상태
기본 값은 False에서 실행을 요청했을 때, True로 세팅된다.</td><td>M</td><td></td></tr>
<tr><td>ext</td><td>Execute Target</td><td>해당 장치 제어가 실행되는 대상 장치을 식별하기 위해서 물리적 장치 정보를 포함하는 〈node〉 자원의 Resource 식별자를 입력한다.</td><td>M</td><td></td></tr>
<tr><td>exra</td><td>Execute Request Argument</td><td>장치 제어를 위해 필요한 Argument를 포함한다.</td><td>O</td><td></td></tr>
<tr><td>exr</td><td>Execute Result</td><td>장치 제어 결과
<table>
<tr><th>Value</th><th>Interpretation</th></tr>
<tr><td>0</td><td>STATUS_REQUEST_SUCCEED</td></tr>
<tr><td>1</td><td>STATUS_REQUEST_UNSUPPORTED</td></tr>
<tr><td>2</td><td>STATUS_REQUEST DENIED</td></tr>
<tr><td>3</td><td>STATUS_CANCELLATION_DENIED</td></tr>
<tr><td>4</td><td>STATUS_INTERNAL_ERROR</td></tr>
<tr><td>5</td><td>STATUS_INVALID_ARGUMENTS</td></tr>
<tr><td>6</td><td>STATUS_RESOURCES_EXCEEDED</td></tr>
<tr><td>7</td><td>STATUS_FILE_TRANSFER_FAILED</td></tr>
<tr><td>8</td><td>STATUS_FILE_TRANSFER_SERVER_AUTHENTICATION_FAILURE</td></tr>
<tr><td>9</td><td>STATUS_UNSUPPORTED_PROTOCOL</td></tr>
<tr><td>10</td><td>STATUS_UPLOAD_FAILED</td></tr>
<tr><td>11</td><td>STATUS_FILE_TRANSFER_FAILED_MULTICAST_GROUP_UNABLE_JOIN</td></tr>
<tr><td>12</td><td>STATUS_FILE_TRANSFER_FAILED_SERVER_CONTACT_FAILED</td></tr>
<tr><td>13</td><td>STATUS_FILE_TRANSFER_FAILED_FILE_ACCESS_FAILED</td></tr>
<tr><td>14</td><td>STATUS_FILE_TRANSFER_FAILED_DOWNLOAD_INCOMPLETE</td></tr>
<tr><td>15</td><td>STATUS_FILE_TRANSFER_FAILED_FILE_CORRUPTED</td></tr>
<tr><td>16</td><td>STATUS_FILE_TRANSFER_FILE_AUTHENTICATION_FAILURE</td></tr>
<tr><td>17</td><td>STATUS_FILE_TRANSFER_FAILED</td></tr>
<tr><td>18</td><td>STATUS_FILE_TRANSFER_SERVER_AUTHENTICATION_FAILURE</td></tr>
<tr><td>19</td><td>STATUS_FILE_TRANSFER_WINDOW_EXCEEDED</td></tr>
<tr><td>20</td><td>STATUS_INVALID_UUID_FORMAT</td></tr>
<tr><td>21</td><td>STATUS_UNKNOWN_EXECUTION_ENVIRONMENT</td></tr>
<tr><td>22</td><td>STATUS_DISABLED_EXECUTION_ENVIRONMENT</td></tr>
<tr><td>23</td><td>STATUS_EXECUTION_ENVIRONMENT_MISMATCH</td></tr>
</table></td><td>M</td><td></td></tr>
</table>

		24 STATUS_DUPLICATE_DEPLOYMENT_UNIT 25 STATUS_SYSTEM_RESOURCES_EXCEEDED 26 STATUS_UNKNOWN_DEPLOYMENT_UNIT 27 STATUS_INVALID_DEPLOYMENT_UNIT_STATE 28 STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_DOWNGRADE_DISALLOWED 29 STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_UPGRADE_DISALLOWED 30 STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_VERSION_EXISTS 31 STATUS_DEVICE_MANAGEMENT_TIME_OUT 32 STATUS_NW_SVR_TRANSFER_FAILED		
exs	Execute Status	장치 제어 상태 정보 Value / Interpretation 1 INITIATED 2 PENDING 3 FINISHED 4 CANCELLING 5 CANCELLED 6 STATUS_NON_CANCELLABLE	M		
ppt	Property	추가 별도 속성 정보 (Reserved)	O		

6.5.3.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request

```
GET /0000000000000001/v1_0/mgmtCmd-
000000010000000020160431_DevReset/execInstance-EI00000000000001608983 HTTP/1.1
Host: 211.115.15.160:9000
Accept: application/xml
X-M2M-Origin: 0000000000000001
X-M2M-RI: 0000000000000001_00001
Content-Type: application/vnd.onem2m-res+xml
uKey:
QmNPUEhuQ2cyVEo1U21kbWpMcWZOOFhJM0tpMEpYK0VpTHk3bkNaZ29KYUJMZDQxbm
NmdzhySHpuRVZEQS94Nw==
```

Response

```
200 OK
Content-Length →428
Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8
Date →Mon, 23 May 2016 11:58:27 GMT
Server →Apache-Coyote/1.1
X-M2M-RI →0000000000000001_00001
X-M2M-RSC →2000

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:exin xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ty>8</ty>
```

```
    <ri>EI0000000000000001608983</ri>
    <rn>EI0000000000000001608983</rn>
    <pi>MC00000000000000000332</pi>
    <ct>2016-05-23T20:49:54+09:00</ct>
    <lt>2016-05-23T20:49:54+09:00</lt>
    <et>9999-12-31T00:00:00+00:00</et>
    <exs>2</exs>
    <ext>ND00000000000000000359</ext>
</m2m:exin>
```

6.5.3.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">
    <op>2</op>
    <to>/0000000000000001/v1_0/mgmtCmd-
    000000010000000020160431_DevReset/execInstance-EI0000000000000001608983</to>
    <fr>000000010000000020160431</fr>
    <ri>0000000000000001_000001</ri>
    <cty>application/vnd.onem2m-prsp+xml</cty>
     <uKey>MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRT
     hoa3FSaCtGK2RDVDhCUA==</uKey>
</m2m:req>
```

Response

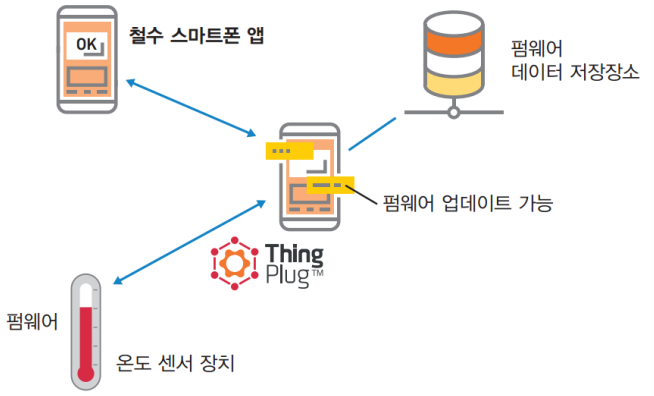
```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<ri>0000000000000001_000001</ri>
<rsc>2000</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>

<pc>
    <exin>
        <ty>8</ty>
```

```
        〈ri〉EI0000000000000001608983〈/ri〉
        〈rn〉EI0000000000000001608983〈/rn〉
        〈pi〉MC00000000000000000332〈/pi〉
        〈ct〉2016-05-23T20:49:54+09:00〈/ct〉
        〈lt〉2016-05-23T20:49:54+09:00〈/lt〉
        〈et〉9999-12-31T00:00:00+00:00〈/et〉
        〈exs〉2〈/exs〉
        〈ext〉ND00000000000000000359〈/ext〉
    〈/exin〉
〈/pc〉
〈/m2m:rsp〉
```

oneM2M Tip! 디바이스 제어 관련 API

철수의 집에 설치된 온도 센서 장치 펌웨어Firmware 업데이트가 제조사에서 신규 개발 완료되었다고 합시다. 이를 어떻게 업데이트할 수 있을까요?

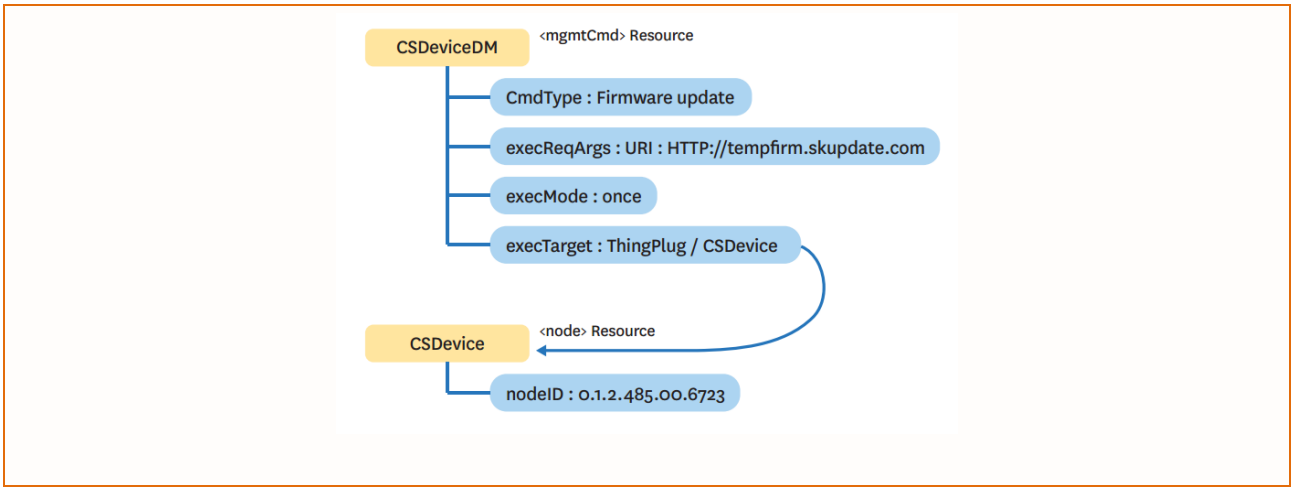
다음과 같은 구조로 시스템이 구성되어 있다고 가정하겠습니다



철수의 스마트 홈 앱에 신규 펌웨어 업데이트 알림이 전달되면, 스마트 홈 앱은 〈mgmtCmd〉 자원 타입을 생성하고 'execTarget' 속성에 물리적 타겟 장치(온도 센서)의 식별자가 명시된 〈node〉 자원을 명시하게 됩니다.

중요한 부분은 〈mgmtCmd〉 자원의 'cmdType' 속성과 'execReqArgs' 속성입니다. 지금의 예를 기준으로 한다면 다음 그림과 같이 자원을ThingPlug 서버에 생성하면 됩니다. 다음 그림에서 execReqArgs에 포함된 웹 주소는 업데이트 할 펌웨어가 저장되어 있는 장소 위치입니다. 이제 ThingPlug 서버가 철수의 스마트 홈 앱을 대신하여 장치에 탑재된 펌웨어 업데이트 기능을 동작시켜주는 메시지를 보낼 것입니다.

요청한 장치 제어 명령이 타겟 장치에서 성공적으로 진행되면, 그 결과를 장치가 전달해야 하겠죠? 이때는 ThingPlug 서버가 별도의 동작 제어 메시지를 실행한 결과를 장치로부터 수신하여 철수의 스마트폰 앱으로 보내기 위하여 〈mgmtCmd〉 자원의 자식 자원으로 〈execInstance〉를 생성하여 해당 자원 속성에 제어 결과를 작성하게 됩니다. 이를 통해 철수의 스마트 홈 앱은 요청한 펌웨어 업데이트 요청이 동작이 되었는지 아닌지를 조회하여 앱 사용자에게 알려줄 수 있습니다.



6.6 Application Server에서 Push 메시지를 전달 받기 위한 Resource API

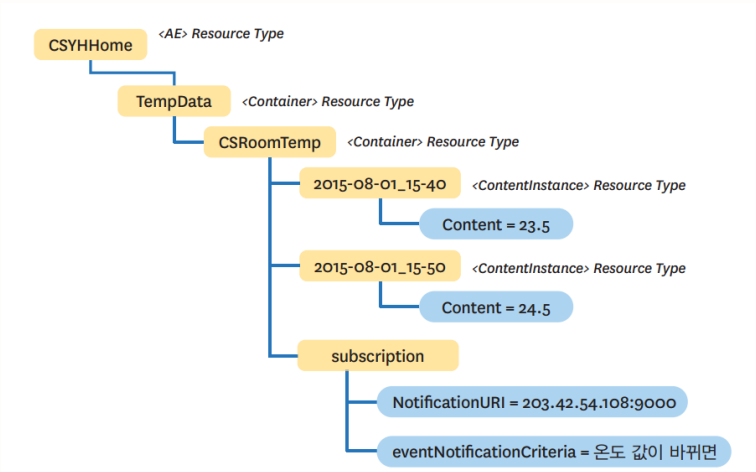
oneM2M Tip! 〈subscription〉 Resource API

IoT 장치를 관리하는 IoT 앱은 대개 다수의 장치를 모니터링하게 됩니다. 이 모니터링을 통해 문제를 미리 발견하는 것이 앱의 존재 의미이기도 합니다. 문제를 발견하는 방법은 크게 두 가지가 있습니다.

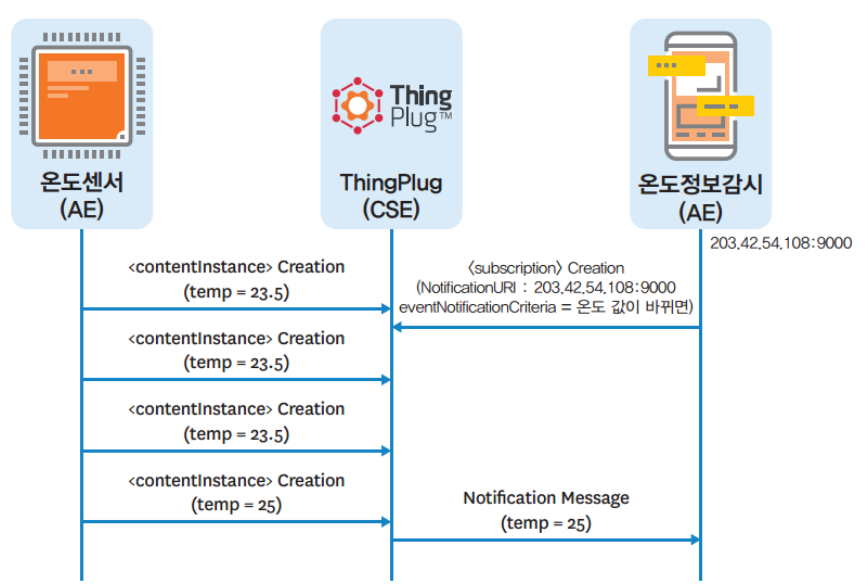
한 가지는 CCTV로 사람을 감시하듯 장치들이 올려주는 정보를 실시간으로 보는 방법이며, 좀 더 스마트한 방법은 올려주는 데이터에 특정 조건을 걸어서 해당 조건을 만족하면 시스템이 관리자 또는 관련 있는 사람들에게 전달하는 방법이 있습니다.

특히 후자를 표준에서는 구독Subscription 및 통지Notification이라고 합니다. 이는 마치 신문을 배달받는 것과 같습니다. 어떤 신문을 구독하려면 신문사에 연락해서 계약을 진행합니다. 이런 행위를 구독이라고 합니다. 이때 계약자는 배달 받을 주소와 배달 받는 신문의 종류 및 배달 주기 등을 전달하고, 계약에 따라 신문이 나오면 희망 배달지에 배달 즉, 통지되는 형태로 서비스가 진행됩니다.

oneM2M 표준 기준으로 특정 데이터를 구독하고 통지 메시지를 받는 방법은 비교적 간단합니다. 다음 표에 있는 〈subscription〉 자원 타입을 관심이 있는 자원의 자식 자원으로 생성하기만 하면 됩니다. 이때 신문 구독 시 계약 조건을 명시하는 것과 같이, 통지 메시지를 받을 주소와 조건을 명시하면 됩니다. 앞서 예로 들었던 자원 구조를 다시 예로 들어보겠습니다.



철수 방에 온도 값이 바뀌는 이벤트에 대한 통지 메시지를 받고 싶다면, 그림과 같이 철수 방의 온도를 저장하는 〈container〉 자원 타입 인스턴스인 CSRoomTemp의 자식 자원으로 〈subscription〉 자원을 생성하고, 구독에 대한 조건을 'eventNotificationCriteria' 속성에 명시합니다. 구독을 설정했던 자원(그림에서는 CSRoomTemp)이 조건에 맞으면 통지 메시지가 'NotficationURI' 속성에 명시한 주소로 전달됩니다.



ThingPlug도 〈subscription〉 기반의 데이터 변경 시 통지 기능을 제공합니다. 다양한 IoT 애플리케이션에 〈subscription〉 자원 기반의 구독 통지 기능을 부여하면 좀 더 스마트한 애플리케이션을 만들 수 있습니다.

6.6.1 〈subscription〉 Create

〈subscription〉은 특정 자원에 쌓이는 정보 변경이나 특정 조건이 발생할 때 통지(notification) 메시지 또는 푸시(Push) 메시지를 ThingPlug 서버가 보내도록 설정하는 Resource API 입니다.

속성 이름		의미	필수 여부	비고
Short	Long			

<table>
<tr><td>enc</td><td>Event Notification Criteria</td><td>통지가 실행(Trigger)가 되는 조건(Condition)
아래와 같은 자식 속성(child)을 추가합니다.
1) rss (Resource Status)
<table><tr><th>값</th><th>의미</th></tr><tr><td>1</td><td>childCreated</td></tr><tr><td>2</td><td>childDeleted</td></tr><tr><td>3</td><td>Updated</td></tr><tr><td>4</td><td>Deleted</td></tr><tr><td>5</td><td>childUpdated(제어결과 업데이트)</td></tr></table>[사용방법]
해당 Subscription이 통지를 보내는 조건이 해당 Subscription의 부모 자원의 Update 결과를 알리는 경우
〈enc〉
〈rss〉3〈/rss〉
〈/enc〉

해당 Subscription이 통지를 보내는 조건이 해당 Subscription의 부모 자원의 다른 자식 자원의 Create를 알리는 경우
〈enc〉
〈rss〉1〈/rss〉
〈/enc〉</td><td>M</td><td></td></tr>
<tr><td>nu</td><td>Notification URI</td><td>통지 메시지가 전달되어야 하는 객체의 주소
• HTTP : HTTP|http://[FQDN or IP Address]:[Port]
• HTTP|https://[FQDN or IP Address]:[Port]
• MQTT : MQTT|[clientID]</td><td>M</td><td></td></tr>
<tr><td>nct</td><td>Notification Content Type</td><td>통지 메시지 타입을 설정
Modified Attribute only (1번)
Whole Resource (2번) – 기본 설정값</td><td>M</td><td></td></tr>
<tr><td>cr</td><td>Creator</td><td>해당 〈subscription〉을 생성한 발신자의 식별자</td><td>M</td><td></td></tr>
</table>

6.6.1.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request
POST /0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-LoRa HTTP/1.1 Host: 211.115.15.160:9000 Accept: application/xml X-M2M-RI: 0000000000000001_000001 X-M2M-Origin: Origin X-M2M-NM: Subscription_1 Content-Type: application/vnd.onem2m-res+xml;ty=23 locale: en uKey: MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCtGK2RDVDhCUA==

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:sub
    xmlns:m2m="http://www.onem2m.org/xml/protocols"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <enc>
        <rss>1</rss>
    </enc>
    <nu>HTTP|http://127.0.0.1:1357</nu>
    <nct>2</nct>
</m2m:sub>
```

Response

```
201 Created
Content-Length →412
Content-Location →/0000000000000001/v1_0/remoteCSE-
00000001000000000020160431/container-LoRa/subscription-subscription_1
Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8
Date →Tue, 24 May 2016 14:34:34 GMT
Server →Apache-Coyote/1.1
X-M2M-RI →0000000000000001_000001
X-M2M-RSC →2001

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><m2m:sub
xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ty>23</ty>
<ri>SS0000000000000000292</ri>
<rn>subscription_1</rn>
<pi>MC0000000000000000574</pi>
<ct>2016-06-27T20:51:29+09:00</ct>
<lt>2016-06-27T20:51:29+09:00</lt>
<enc>
<rss>1</rss>
</enc>
<nu>HTTP|http://127.0.0.1:1357</nu>
<nct>2</nct>
</m2m:sub>
```

6.6.1.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 00000001000000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<op>1</op>
<to>/0000000000000001/v1_0/remoteCSE-00000001000000020160431/container-LoRa
</to>
<fr>00000001000000020160431</fr>
<ty>23</ty>
<ri>0000000000000001_000001</ri>
<nm>subscription_1</nm>
<cty>application/vnd.onem2m-prsp+xml</cty>
<uKey>MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3
FSaCtGK2RDVDhCUA==</uKey>

<pc>
  <sub>
    <enc>
      <rss>1</rss>
    </enc>
    <nu>MQTT|thingplugtest_0001</nu>
    <nct>2</nct>
  </sub>
</pc>
</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<ri>0000000000000001_000001</ri>
<rsc>2001</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>
<pc>
  <sub>
    <ty>23</ty>
    <ri>SS000000000000000000301</ri>
    <rn>subscription_1</rn>
    <pi>MC00000000000000000574</pi>
    <ct>2016-06-28T17:56:11+09:00</ct>
    <lt>2016-06-28T17:56:11+09:00</lt>
    <enc>
      <rss>1</rss>
    </enc>
    <nu>MQTT| thingplugtest_0001</nu>
    <nct>2</nct>
```

```
  〈/sub〉
〈/pc〉
〈/m2m:rsp〉
```

6.6.2 Client 프로그램을 활용한 Subscription 메시지 확인

관심이 있는 자원(예: container)을 구독하고자 하는 경우, 해당 자원의 자식 자원으로 〈subscription〉을 생성하게 됩니다. 생성시 정한 조건(enc)이 발생하면 nu(notification URI)로 notification 메시지가 전달되며, mqtt client에서 토픽 "/oneM2M/req_msg/{AppEUI}/{clientID}" 을 구독하거나, nu로 등록한 HTTP Response를 받을 수 있는 서버가 있다면 해당 메시지를 전달받을 수 있습니다.

6.6.2.1 MQTT Binding Example(Subscription Push Message)

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/Application Server ID (AppSVRID)/0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/Application Server ID (AppSVRID)(Subscription 을 등록 한 경우 필요)

Request (실제로 전달된 주기 보고 데이터로 LoRa 네트워크 에서 ThingPlug로 전달되는 내용입니다.)

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<op>1</op>
<to>/0000000000000001/v1_0/remoteCSE-
000000010000000020160431/container-LoRa</to>
<fr>000000010000000020160431</fr>
<ty>4</ty>
<ri>000000010000000020160431</ri>
<cty>application/vnd.onem2m-prsp+xml</cty>
<dKey>c1RnWnlNeW4wd1VYb0cwTUJ2KzhSYW5tc1NNSHQ0MXZtMVlKZjIvODlwKzZYNzc4
Sld4b3RJZE8zV1R4OXpqRg==</dKey>
<pc>
<cin>
<cnf>text</cnf>
<con>11,22,33</con>
</cin>
</pc>
</m2m:req>
```

전달된 Push Message

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<op>1</op>
<fr>000000010000000020160431</fr>
<to>000000010000000020160431</to>
<rqi>989fb3b9-9f66-47ba-8a15-e542b02b8611</rqi>
<pc>
```

```
<cin>
<ty>4</ty>
<ri>CI0000000000002301730</ri>
<rn>CI0000000000002301730</rn>
<pi>CT00000000000000000201</pi>
<ct>2016-08-17T14:32:57+09:00</ct>
<lt>2016-08-17T14:32:57+09:00</lt>
<et>2016-09-01T14:32:57+09:00</et>
<st>13124</st>
<cr>RC00000000000000002578</cr>
<cnf>text</cnf>
<cs>8</cs>
<con>11,22,33</con>
</cin>
</pc>
</m2m:req>
```

6.6.3 〈subscription〉 Retrieve

기 설정된 〈subscription〉 자원을 회수하는 Resource API 입니다.

6.6.3.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request

```
GET /0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-
LoRa/subscription-subscription_1 HTTP/1.1
Host: 211.115.15.160:9000
Accept: application/xml
X-M2M-RI: 12345
X-M2M-Origin: Origin
uKey:
MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCt
GK2RDVDhCUA==
```

Response

```
200 OK
Content-Length →421
Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8
Date →Tue, 28 Jun 2016 08:38:58 GMT
Server →Apache-Coyote/1.1
X-M2M-RI →12345
X-M2M-RSC →2000

〈?xml version="1.0" encoding="UTF-8" standalone="yes"?〉
〈m2m:sub xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"〉
```

```
    <ty>23</ty>
    <ri>SS0000000000000000299</ri>
    <rn>DevReset01</rn>
    <pi>MC00000000000000000574</pi>
    <ct>2016-06-27T20:51:29+09:00</ct>
    <lt>2016-06-27T20:51:29+09:00</lt>
    <enc>
        <rss>1</rss>
    </enc>
    <nu>HTTP|http://127.0.0.1:1357</nu>
    <nct>2</nct>
</m2m:sub>
```

6.6.3.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<op>2</op>
<to>/0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-
LoRa/subscription-subscription_1 </to>
<fr>000000010000000020160431</fr>
<ri>0000000000000001_000001</ri>
<cty>application/vnd.onem2m-prsp+xml</cty>
<uKey>MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3
FSaCtGK2RDVDhCUA==</uKey>

</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<ri>0000000000000001_000001</ri>
<rsc>2000</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>
```

```
<pc>
  <sub>
    <ty>23</ty>
    <ri>SS00000000000000000301</ri>
    <rn>subscription_1</rn>
    <pi>MC00000000000000000574</pi>
    <ct>2016-06-28T17:56:11+09:00</ct>
    <lt>2016-06-28T17:56:11+09:00</lt>
    <enc>
      <rss>1</rss>
    </enc>
<nu>MQTT|thingplugtest_0001</nu>
    <nct>2</nct>
  </sub>
</pc>
</m2m:rsp>
```

6.6.4 〈subscription〉 Update

기 설정된 〈subscription〉 자원을 갱신하는 Resource API 입니다.

6.6.4.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request

```
PUT /0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-
LoRa/subscription-subscription_1 HTTP/1.1
Host: 211.115.15.160:9000
Accept: application/xml
X-M2M-RI: 12345
X-M2M-Origin: Origin
Content-Type: application/vnd.onem2m-res+xml
uKey:
MlBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCt
GK2RDVDhCUA==

<?xml version="1.0" encoding="UTF-8"?>
<m2m:sub
    xmlns:m2m="http://www.onem2m.org/xml/protocols"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <enc>
        <rss>1</rss>
    </enc>
    <nu>HTTP|http://127.0.0.1:1357</nu>
    <nct>2</nct>
```

```
</m2m:sub>
```

Response

```
200 OK
Content-Length →401
Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8
Date →Tue, 28 Jun 2016 08:42:22 GMT
Server →Apache-Coyote/1.1
X-M2M-RI →12345
X-M2M-RSC →2004

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:sub xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ty>23</ty>
    <ri>SS000000000000000000299</ri>
    <rn>subscription_1</rn>
    <pi>MC00000000000000000574</pi>
    <ct>2016-06-28T17:32:44+09:00</ct>
    <lt>2016-06-28T17:42:22+09:00</lt>
    <enc/>
    <nu>HTTP|http://127.0.0.1:1357</nu>
    <nct>2</nct>
</m2m:sub>
```

6.6.4.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)

Request

```
<m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-
v1_0_0.xsd">

<op>3</op>
<to>/0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-
LoRa/subscription-subscription_1 </to>
<fr>000000010000000020160431</fr>
<ty>23</ty>
<ri>0000000000000001_000001</ri>
<cty>application/vnd.onem2m-prsp+xml</cty>
```

```
<uKey>MlBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3
FSaCtGK2RDVDhCUA==</uKey>

<pc>
  <sub>
    <enc>
      <rss>1</rss>
    </enc>
    <nu>MQTT|thingplugtest_0001</nu>
    <nct>2</nct>
  </sub>
</pc>
</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<ri>0000000000000001_000001</ri>
<rsc>2004</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>
<pc>
  <sub>
    <ty>23</ty>
    <ri>SS00000000000000000301</ri>
    <rn>subscription_1</rn>
    <pi>MC00000000000000000574</pi>
    <ct>2016-06-28T17:56:11+09:00</ct>
    <lt>2016-06-28T17:56:35+09:00</lt>
    <enc>
      <rss>1</rss>
    </enc>
<nu>MQTT|thingplugtest_0001</nu>
    <nct>2</nct>
  </sub>
</pc>
</m2m:rsp>
```

6.6.5 〈subscription〉 Delete

기 설정된 〈subscription〉 자원을 삭제하는 Resource API 입니다.

6.6.5.1 HTTP Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431

Request
DELETE /0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-LoRa/subscription-subscription_1 HTTP/1.1 Host: 211.115.15.160:9000 Accept: application/xml X-M2M-RI: 12345 X-M2M-Origin: Origin uKey: MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3FSaCtGK2RDVDhCUA==

Response
200 OK Content-Length →0 Content-Type →application/vnd.onem2m-res+xml;charset=UTF-8 Date →Tue, 28 Jun 2016 08:45:40 GMT Server →Apache-Coyote/1.1 X-M2M-RI →12345 X-M2M-RSC →2002

6.6.5.2 MQTT Binding Example

- AppEUI: 0000000000000001
- LTID: 000000010000000020160431
- Topic: /oneM2M/req/clientID /0000000000000001
- Subscribe : (선 지정 필수)
  - /oneM2M/req_msg/+/clientID (Subscription 을 등록 한 경우 필요)
  - /oneM2M/resp/clientID/+ (해당 Request 에 대한 Response)

Request
〈m2m:req xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-v1_0_0.xsd"〉 〈op〉4〈/op〉 〈to〉/0000000000000001/v1_0/remoteCSE-000000010000000020160431/container-

```
LoRa/subscription-subscription_1 </to>
<fr>00000001000000000020160431</fr>
<ty>23</ty>
<ri>0000000000000001_000001</ri>
<cty>application/vnd.onem2m-prsp+xml</cty>
<uKey>MIBhSmF5VzdpT0RoL1grKzFhTlJaVjVZSko3SFZhdzhCeGErbHY0RzhiTW5FdGFtRThoa3
FSaCtGK2RDVDhCUA==</uKey>

</m2m:req>
```

Response

```
<m2m:rsp xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-
requestPrimitive-v1_0_0.xsd">
<ri>0000000000000001_000001</ri>
<rsc>2002</rsc>
<cty>application/vnd.onem2m-res+xml;charset=UTF-8</cty>
<pc>
</pc>
</m2m:rsp>
```

A1. Change History

버 전	일 자	작성자	내 용
0.1	2016.03.12	안홍범 M	• 초안 작성
0.2	2016.03.14	안홍범 M	• MQTT 접속 절차 및 Connection Establishment 가이드
0.3	2016.03.16	안홍범 M	• 식별 체계 추가
0.4	2016.03.18	안홍범 M	• ThingPlug Resource API (〈node〉, 〈remoteCSE〉, 〈container〉, 〈contentInstance〉, 〈mgmtCmd〉)
0.5	2016.03.26	안홍범 M	• ThingPlug Resource API (〈subscripton〉) 추가 • Resource Tree 변경 • LoRa Data Profile 제공
0.6	2016.05.30	안홍범 M	• 용어 수정 • HTTP API 예제 추가 • oneM2M 표준 설명
0.7	2016.06.18	안홍범M	• ThingPlug 접속 방법 (HTTP, MQTT) • HTTP 및 MQTT over TLS 설명 추가
0.9	2016.08.01	안홍범M	• uKey 할당 방법 설명 추가
1.0	2016.08.05	안홍범M	• 단말 제어 방법 상세화 • Status Code 첨부 추가
1.1	2016.08.08	안홍범M	• Application Specific 단말 제어 방법 수정
1.2	2016.08.22	성지웅M	• mgmtCmd, Subscription Push Message MQTT 예제 추가 • dKey 소개 추가
1.3	2016.09.20	성지웅M	• mgmtCmd 중 RepPerchange, RepImmediate 삭제
1.4	2016.10.04	성지웅M	• MQTT clientID 정보 ApplicationSVRID 로 수정 • 자원 생성시점 Best Practice 추가
1.5	2016.11.15	안홍범M	• MQTT 브로커 정책 변동에 따른 MQTT 브로커 접속 방법 수정
1.6	2017.01.11	성지웅M	• LoRa 테스트베드, 시험번호 및 상용 서비스 소개 추가
1.7	2017.04.03	김규백M	• MQTT 토픽 정책 일부 변경, Clean Session 설정 True 로 가이드

A2. Response Status Code

[RSC framework overview]

The RSCs are categorised as one of 6 classes:

Definition of Response Status Code class

Status Class	Codeclass	Interpretation
Informational	1xxx	The request is successfully received, but the request is still on process.
Success	2xxx	The request is successfully received, understood, and accepted.
Redirection	3xxx	(Not used in present release)
Originator Error	4xxx	The request was malformed by the Originator and, is rejected.
Receiver Error	5xxx	The requested operation cannot be performed due to an error condition at the Receiver CSE.
Network Service Error	6xxx	The requested operation cannot be performed due to an error condition at the Network Serivce Entity.

[Definition of Response Status Codes]

[Overview]

The tables in the following clauses specify the RSCs for oneM2M releases. Each RSC includes: aresponse status represented as numeric code. The supplemental information may be returned when it is needed.

[Informational response class]

specifies the RSCs for acknowledgement responses for each release.

Informational Responses class

Numeric Code	Description
1000	ACCEPTED

[Successful response class]

Table below specifies the RSCs for Successful responses.

RSCs for Successful response class

Numeric Code	Description
2000	OK
2001	CREATED
2002	DELETED
2004	UPDATED
2100	CONTENT_EMPTY

[Redirection response class]

In this release, no values in this response class are defined.

RSCs for Redirection response class

Numeric Code	Description

undefined	undefined

[Originator Error response class]

Table below specify the RSCs for Originator Error responses.

RSCs for Originator Error response class

Numeric Code	Description
4000	BAD_REQUEST
4004	NOT_FOUND
4005	OPERATION_NOT_ALLOWED
4008	REQUEST_TIMEOUT
4101	SUBSCRIPTION_CREATOR_HAS_NO_PRIVILEGE
4102	CONTENTS_UNACCEPTABLE
4103	ACCESS_DENIED
4104	GROUP_REQUEST_IDENTIFIER_EXISTS
4105	CONFLICT

[Receiver Error response class]

Table below specifies the RSCs for Receiver Error responses.

RSCs for Receiver Error response class

Numeric Code	Description
5000	INTERNAL_SERVER_ERROR
5001	NOT_IMPLEMENTED
5103	TARGET_NOT_REACHABLE
5105	NO_PRIVILEGE
5106	ALREADY_EXISTS
5203	TARGET_NOT_SUBSCRIBABLE
5204	SUBSCRIPTION_VERIFICATION_INITIATION_FAILED
5205	SUBSCRIPTION_HOST_HAS_NO_PRIVILEGE
5206	NON_BLOCKING_REQUEST_NOT_SUPPORTED
5207	NOT_ACCEPTABLE

[Network System Error response class]

Table below specifies the RSCs for when the External System reported some errors.

RSCs for Network Service Error response class

Numeric Code	Description
6003	EXTERNAL_OBJECT_NOT_REACHABLE
6005	EXTERNAL_OBJECT_NOT_FOUND
6010	MAX_NUMBER_OF_MEMBER_EXCEEDED
6011	MEMBER_TYPE_INCONSISTENT

6020	MGMT_SESSION_CANNOT_BE_ESTABLISHED
6021	MGMT_SESSION_ESTABLISHMENT_TIMEOUT
6022	INVALID_CMDTYPE
6023	INVALID_ARGUMENTS
6024	INSUFFICIENT_ARGUMENTS
6025	MGMT_CONVERSION_ERROR
6026	MGMT_CANCELLATION_FAILED
6028	ALREADY_COMPLETE
6029	MGMT_COMMAND_NOT_CANCELLABLE