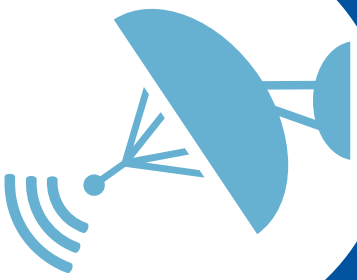
TTA-14092-SD
발행일 : 2014년 11월 30일



# oneM2M 서비스 플랫폼 표준 해설서

Telecommunications Technology Association

TTA 한국정보통신기술협회
Telecommunications Technology Association

| | |
|---|---|
| 발간 | 한국정보통신기술협회 (TTA) |
| 저자 | 송재승 (세종대학교) |
| | 이규명 (Liverpool University, 모다정보통신) |
| | 서정욱 (남서울대학교) |
| | 강남희 (덕성여자대학교) |
| | 김재호 (한국전자부품연구원) |
| | 최성찬 (한국전자부품연구원) |
| | 양현석 (LG U+) |
| | 최희동 (LG 전자) |
| | 정승명 (LG 전자) |
| | 김성윤 (LG 전자) |
| | 안홍범 (LG 전자) |
| 감수 | 류창호 (정보통신표준협회 프로젝트 메니저) |

TTA-14093-SD
발행일 : 2014년 11월 30일

# oneM2M 서비스 플랫폼 표준 해설서

한국정보통신기술협회
Telecommunications Technology Association

# CONTENTS

# 1. 사물 인터넷과 oneM2M 국제 표준

## 1.1. 사물인터넷의 개념

사물인터넷(Internet of Things)은 우리 주변의 모든 사물들이 인터넷에 연결되어 서로 대화하고 교감하며 정보를 주고받을 수 있게 해주는 지능형 기술 및 서비스를 의미한다. 사람의 간섭 없이도 모든 사물들이 스스로 대화하고 주변환경을 분석해서 서비스를 제공하는 시대를 열어주는 개념으로 디지털 혁명의 기술로 인식되고 있다.

1999년 MIT Auto-ID Center 설립자인 캐빈 아스톤 (Kevin Ashton)이 처음 사물인터넷 (Internet of Things)에 대한 개념 및 용어를 처음 제안해서 사용되기 시작한 이후 지속적인 발전을 거듭왔다. 최근의 사물인터넷은, 다양한 기기들과 서비스들의 개발로 이미 우리 생활 속에서 손쉽게 접해볼 수 있게 되었다.

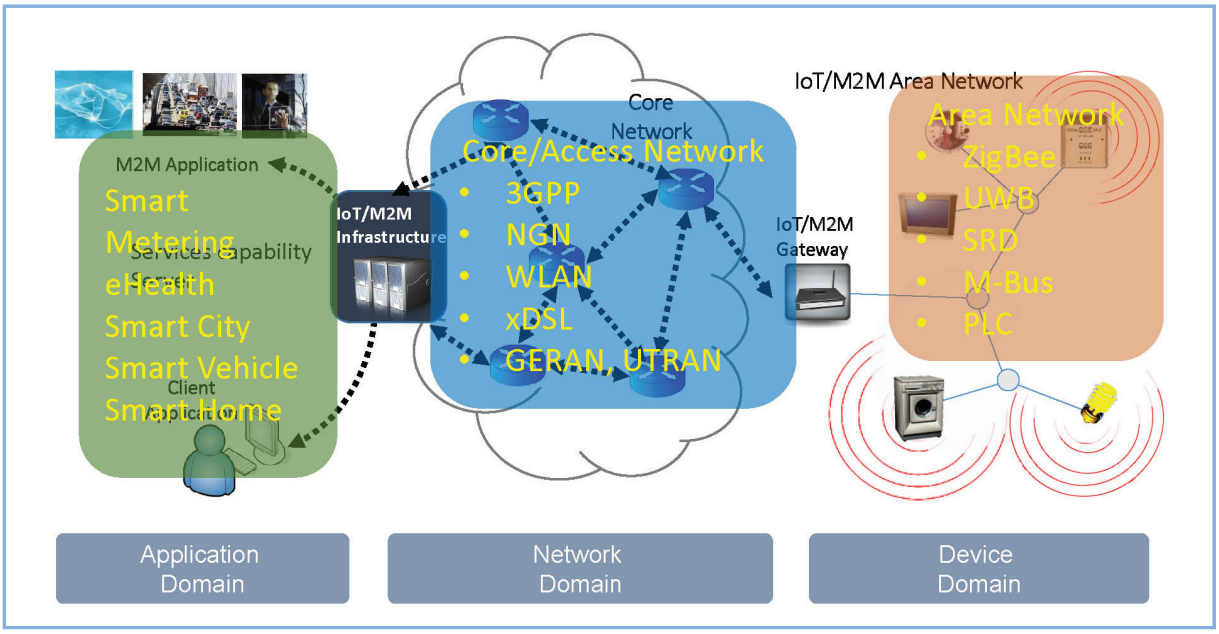
사물인터넷을 이용한 IT 제품은 이미 다양하게 시장에 나와있다. 특히 스마트 홈, 스마트 카 그리고 e-헬스 등과 같은 분야에서는 사물인터넷 서비스 및 디바이스 개발이 한창인 상황이다.

사물인터넷에서는 사물들간의 통신이 이루어져야만 하고, 이를 지원하는 기반 기술이 바로 사물 통신 (Machine-to-Machine Communication)이다. 실제로 현재 사용되어지고 있는 모든 사물인터넷 제품들은 (예를 들면, 구글 글라스, 스마트 워치, 콘넥티드 자동차 등) M2M 기술을 사용하고 있다.

현재 개발되어진 많은 사물인터넷 서비스 그리고 디바이스들은 동일 제조사 또는 동일 서비스 영역에서만 동작을 하는 경우가 대부분이다. 서로 다른 제조사로 부터의 사물들, 그리고 다른 사업 영역에서 사용되어지는 사물들 (예를 들어, 스마트 홈의 가전과 스마트 카) 간의 사물 통신이 이루어지기 위해서는 표준화된 방식이 절대적으로 필요하다.

## 1.2. oneM2M의 개요

oneM2M의 본질은 그 동안 사물인터넷이 성장하지 않았던 이유를 생각하면 쉽게 답을 찾을 수 있는데, 궁극적으로 규모의 경제 실현을 목적으로 두고 있다. 과거 사물인터넷은 다양한 응용 간의 호환을 위한 인터페이스의 부재, 국가별 또는 지역별 또는 시장별로 서로 다른 서비스 요구사항과 세분화된 시장으로 인해 silo형태의 서비스 제공 구조를 좀처럼 벗어나지 못했고, 그로 인해 사물인터넷 서비스 제공에 있어서 디바이스 등 제품과 서비스 플랫폼의 개발 비용이 좀처럼 줄어들지 않아 사물인터넷 성장에 큰 장애가 되었다. 이를 해결하기 위해 사물인터넷에서의 공통 표준 기술 구현을 목적으로 oneM2M이 생겨나게 된다.




〈그림 1-1〉 사물인터넷 도메인

## 1.3. oneM2M 사물인터넷 플랫폼

사물인터넷 시스템을 도메인을 기준으로 구별하면 그림 〈1-1〉과 같이 센서, 액츄에이터, 디바이스가 위치하고 있는 디바이스 도메인 영역과 게이트웨이를 통한 액세스 네트워크 및 코어 망이 위치한 네트워크 도메인 영역, 실제 사물인터넷 서비스를 실현하기 위한 어플리케이션이 위치한 어플리케이션 도메인 영역으로 구분할 수 있다. 실제 사용자에게 보이는 부분인 어플리케이션 영역의 서비스는 스마트 시티, 스마트 홈, 스마트 자동차, 헬스케어, 스마트 그리드 서비스 등이 될 수 있고, 해당 서비스의 실현을 위해서는 사물인터넷 디바이스와 관련된 데이터의 수집과, 제어의 기본 기능에서부터 데이터 분석, 디바이스 관리, 서비스 이용과금 체계 등 확장 기능이 지원되는 플랫폼이 필수적으로 요구된다.

기존의 사물인터넷 관련 서비스를 제공하기 위한 플랫폼은 상당수가 어플리케이션 서비스 종속적인 플랫폼으로 제공되었다. 이러한 플랫폼 개발 트렌드는 사물인터넷 플랫폼의 파편화 및 관련 산업 활성화에 저해요소로 작용하였고 이를 타개하고자 oneM2M의 이름으로 유럽, 미주, 한국, 중국, 일본의 7개의 SDO(Standards Development Organization)들이 함께 표준화한 버티컬 서비스에 종속적이지 않은 공통 플랫폼이 onoM2M 플랫폼이라고 할 수 있다.

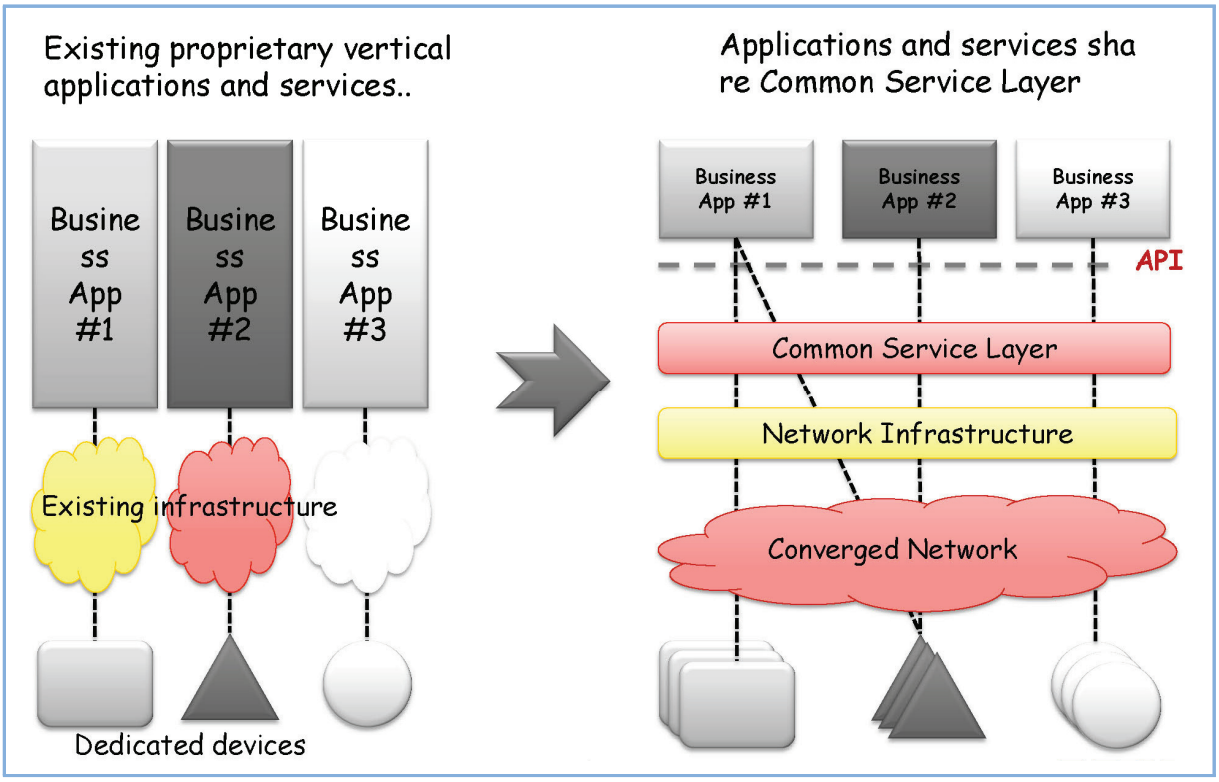



그림 〈1-2〉 사물인터넷 공통 플랫폼으로의 진화

기존 사물인터넷 플랫폼이 서비스 프로바이더가 제공하는 어플리케이션 서비스에 종속적이었던 특정 서비스 중심적인 플랫폼으로 개발되었다면, 그림 〈1-2〉와 같이 oneM2M 표준 플랫폼이 지향하는 바는 어플리케이션 A, B, C 등과 같이 다양한 버티컬 서비스가 존재하는 상황에서 해당 어플리케이션에 종속적이지 않으면서 해당 서비스들을 모두 지원할 수 있는 공통 서비스 플랫폼모델 구조를 가지고 있다. 또한 해당 oneM2M 플랫폼을 기반으로 서비스를 제공 하고자 하는 서비스 프로바이더들은 공통 플랫폼이 제공하는 기본 및 확장 기능들을 활용하여 자사의 서비스를 제공할 수 있다.

## 1.4. oneM2M의 정의

oneM2M은 2012년 7월에 공식적으로 출범하게 되어 사물인터넷에서의 국제 공통 표준 규격 개발이 시작되었다. 2012년 7월 공식 출범이 있기 전까지, 각 세계 표준 개발 기구인 TTA(한국), ETSI(유럽), ATIS/TIA(북미), ARIB/TTC(일본), 중국(CCSA)은 2011년 3월를 시작으로 사물인터넷에 대한 표준을 위한 협력체 신설에 대한 논의를 하였다.

이후, 같은 해의 7월에 서울에서 열린 1차 표준화 국제 회의를 시작으로, 이듬해 3월의 일본에서 열린 4차 회의까지 각 표준화 단체별 사물인터넷 협력 프로젝트 설립에

대한 선결 사항 및 추진 계획 및 협력 분야 선정과 함께 Continua, HGI, OMA 등 응용 분야 참여 등 oneM2M 출범을 위한 주요 쟁점 사항에 대해 논의되었다.

2012년 7월의 oneM2M 출범은 표준 개발 기구(이하 "SDO", Standard Development Organization) 간의 제휴 협약 (SDO partnership agreement)를 기반으로 기존의 SDO별 사물인터넷 표준 개발을 지양하고, oneM2M을 통해 국제 공통 표준을 추진하기로 하였으며, 표준화의 범위, 주요 타겟 서비스 및 조직 세팅 등 본격 활동을 위한 구체화 작업에 들어갔다.

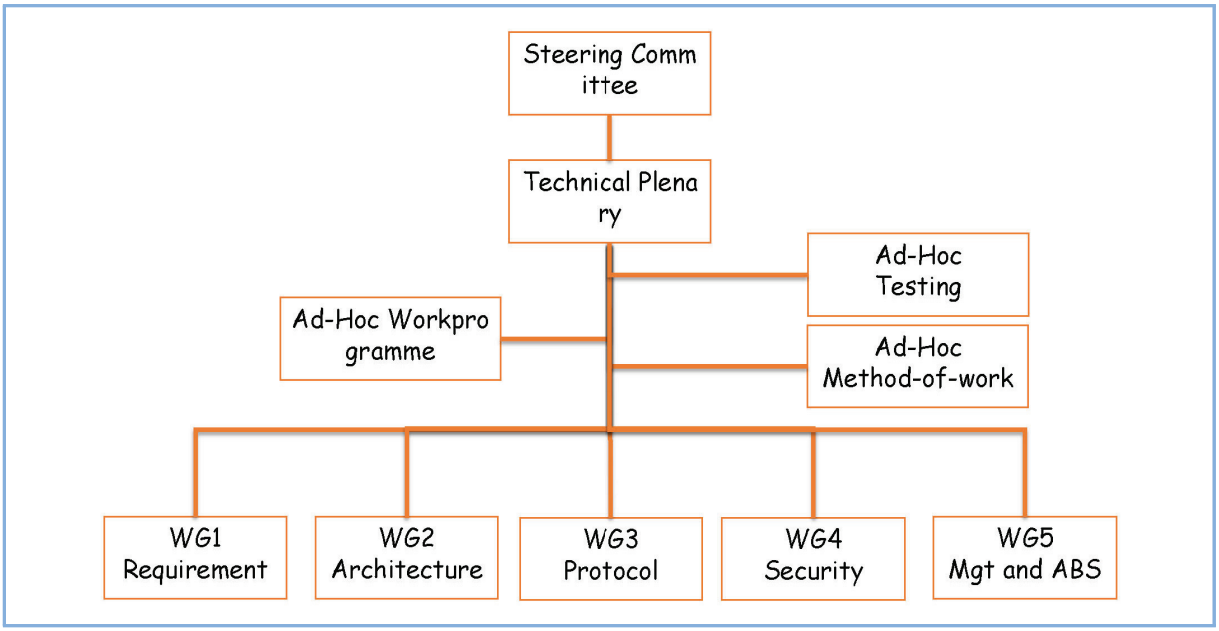
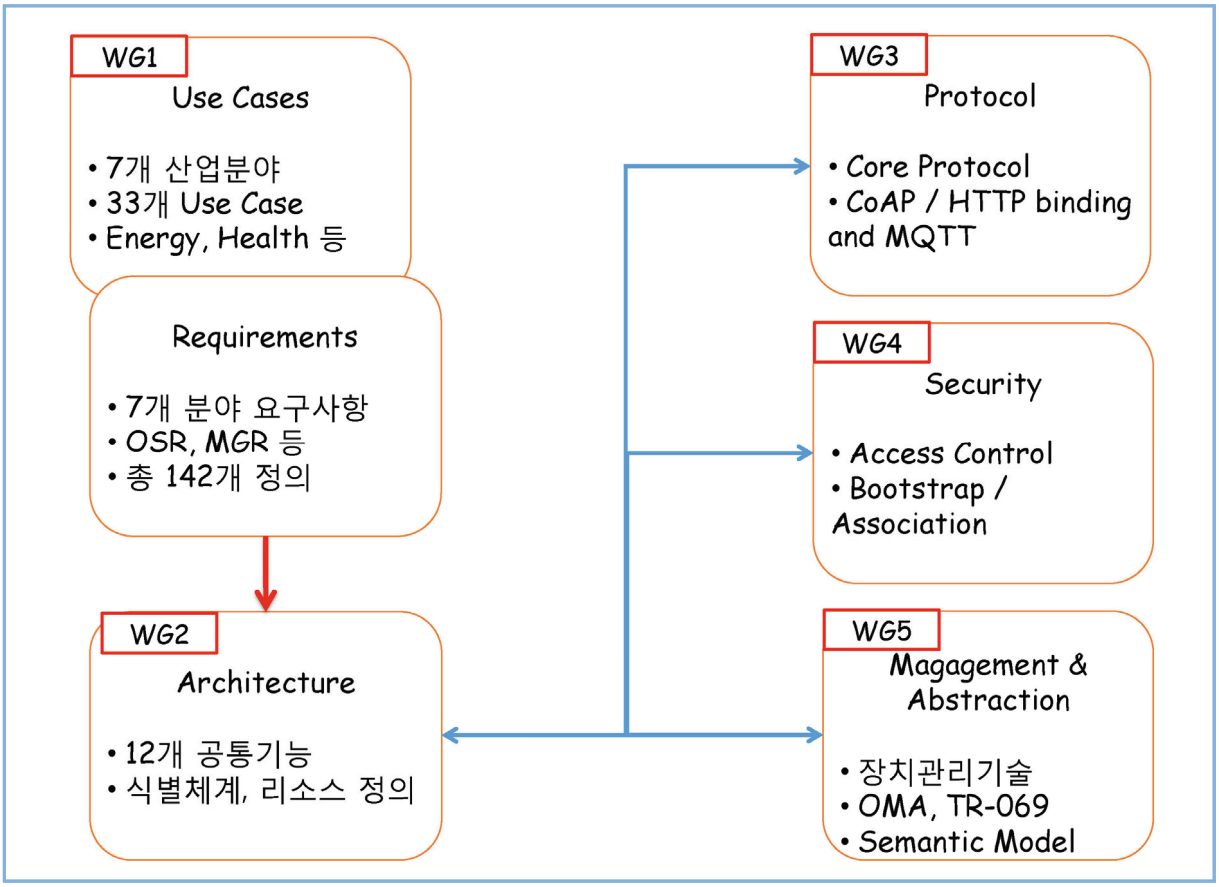



그림 〈1-3〉 oneM2M 표준화 조직 구성도

그림 〈1-3〉에서 보는 바와 같이, oneM2M 표준화 조직 구성은 크게 운영회(Steering Committee)와 기술 총회(Technical Plenary)의 둘로 나뉜다. 운영회에서는 oneM2M의 전반적인 운영을 맡고 있으며, 기술총회에서는 실제 표준화를 담당하고 있으며, 여기에는 요구사항 그룹(WG1), 구조 설계 그룹(WG2), 프로토콜(WG3), 보안(WG4), 장치 관리 및 추상화(WG5)의 워킹 그룹이 있다. 각 워킹그룹에서는 다음과 같은 역할을 담당하고 있다. (그림 〈1-4〉)

- ✓ 요구사항 그룹 (WG1): 다양한 버티컬 산업 영역으로 부터 유스케이스를 수집하고, 이로부터 공통된 서비스 요구사항들을 도출하는 역할을 담당한다.
- ✓ 구조 설계 그룹 (WG2): 공통 서비스 요구사항들로 부터 시스템 설계를 담당한다.
- ✓ 프로토콜 워킹 그룹 (WG3): 서비스 프로토콜들을 담당한다.
- ✓ 보안 그룹 (WG4): 보안 기술을 담당한다.

✓ 장치 관리, 추상화 및 시맨틱 그룹 (WG5): 장치에 대한 관리, 사물들을 데이터 수준에서 재사용할 수 있도록 해주는 추상화 및 시맨틱 관련 기술을 담당한다.




〈그림 1-4〉 oneM2M 표준화 각 WG별 작업 내용

12차 기술총회에서의oneM2M 릴리즈 1.0승인에 이어, oneM2M launch 이벤트 등을 통한 홍보 및 지속적인 워크 아이템을 통해 사물인터넷 서비스의 실제 구현에 있어 완성도를 높이고 있다.

## 1.5. oneM2M 해설서

본 해설서는 ICT 기술의 융합에 따라 세부 분야 표준화전문가도 타 표준의 내용을 많은 부분 참조하고 있음에 따라 기술간, 전문가간 기술 정보교류를 용이하게 할 수 있도록 oneM2M 표준을 쉽게 설명한 해설서를 제공하고자 한다.

표 〈1-1〉 해설서에서 다루는 oneM2M 표준 규격

| 워킹그룹 | 기술규격 |
|---|---|
| 요구사항 (WG1) | TS 0002 - Requirements TS |
| | TS 0011 - Definitions and Acronyms TS |
| 아키텍처 (WG2) | TS 0001 - Architecture TS |
| 프로토콜 (WG3) | TS 0004 - Core Protocol TS |
| | TS 0008 - CoAP Protocol Binding TS |
| | TS 0009 - HTTP Protocol Binding TS |
| | TS 0010 – MQTT Protocol Binding TS |
| 보안 (WG4) | TS 0003 - Security Solutions TS |
| 장치관리, 추상화 및 시맨틱 (WG5) | TS 0005 - Management Enablement (OMA) TS |
| | TS 0006 - Management Enablement (BBF) TS |
| | TR 0007 - Study of Abstraction and Semantics Enablements |

본 해설서에서는 oneM2M release1.0 에 포함된 9개의 기술규격에 시맨틱과 관련된 스터디를 하고 있는 Technical Report를 추가한 총 10개 규격에 대한 영문표준 기술규격별 조사 및 분석을 oneM2M 표준 전문가들에 의해 진행 하였으며, 다음은 이후 각 장별로 다루고 있는 내용에 대한 설명이다.

- ✓ 제 2장: 우리의 일상이 된 사물인터넷 서비스 (oneM2M Use Case and Requirements TS & TR) – WG1에서의 전반적인 활동과 관련 스펙에 대한 내용을 사업자의 관점에서 해설한다.
- ✓ 제 3장: 사물인터넷 표준 플랫폼이 왜 중요한가? (oneM2M Architecture TS) – WG2의 표준 문서에 대해서, oneM2M 구조 스타일, 상세 구조 엔티티, 리소스 구조 등의 내용을 해설한다.
- ✓ 제 4장: Things 들간에 표준으로 소통하는 법 (HTTP, CoAP, MQTT TS & TR) – oneM2M에서 사용되어지는 다양한 프로토콜들 (oneM2M의 Core 프로토콜과 CoAP, HTTP 그리고 MQTT)에 대한 내용을 다룬다.
- ✓ 제 5장: 보안 및 프라이버시 (oneM2M Security TS & TR) – oneM2M의 보안과 관련된 내용을 다룬다.
- ✓ 제 6장: Things들을 관리하고 제어하기 (oneM2M DM TS & TR) – 사물인터넷 디바이스들을 관리하는 다양한 기법들을 정리해 놓은 oneM2M Device Management 표준 규격을 설명한다.
- ✓ 제 7장: 의미로 이해하고 행동하는 사물 인터넷 (oneM2M Semantics TR) – 사물인터넷에 의미론적인 내용을 결합하는 시멘틱 사물인터넷과 관련된 규격인 시멘틱스 TR에 대한 설명을 제공한다.

# 2. 우리의 일상이 된 사물 인터넷 서비스

**관련 oneM2M 표준 문서**

TR-0001: oneM2M Use Case Collection
TS-0002: M2M Requirements
TS-0011: Definitions and Acronyms

## 2.1. oneM2M에서의 구성원 및 기능적인 역할

사물인터넷 서비스를 제공한다는 것은 사물인터넷과 관련한 다른 이해관계자간의 협업이 요구된다. oneM2M에서는 구성원을 유저(사용자), Application 서비스 제공자, M2M 서비스 제공자, 통신망 제공자의 4가지로 구분하고 있으며, 각각의 구성원의 역할은 다음과 같다.

✓ User/End-user
- M2M Solution을 사용하는 end-user(개인 또는 기업)

✓ Application Service Provider
- M2M Application Service 제공
- M2M Application 운영

✓ M2M Service Provider
- Application Service Provider에게 M2M Service 제공
- M2M Common Capability Set(MSCS) 운영

✓ Network Operator
- M2M Service Provider에게 통신망 등 네트워크 제공
- 기저 네트워크(예. 통신망) 운영

## 2.2. oneM2M 유스케이스

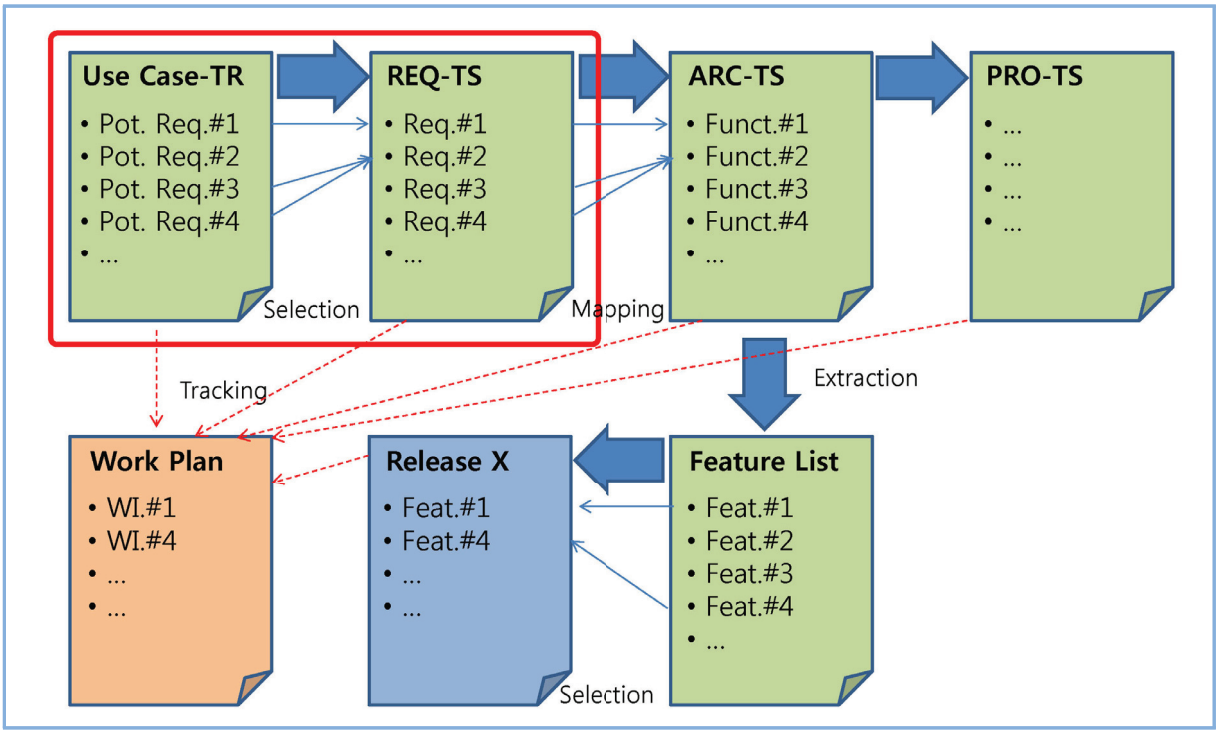



그림 〈2-1〉 oneM2M에서의 유스케이스와 요구사항 도출 관계

다른 표준화도 마찬가지로, 유스케이스들로부터 수집되는 다양한 정보들을 토대로 Requirements(요구사항)을 도출하고 이를 통해 표준 규격을 완성해 나간다 〈그림 2-1〉. oneM2M에서는 해당 작업을 WG1 에서 진행하고 있으며, 아래의 표에서와 같이 oneM2M 표준규격의 첫 단추를 채우는 부분을 맡고 있다.

oneM2M에서는 '14년 8월 release 1.0을 publish하기 위한 첫 단계로 다양한 산업 분야에서의 유스케이스를 취합하였고, 총 7개의 산업분야의 33개 유스케이스가 채택되었는데, 각각의 분야는 에너지, 기업, 헬스, 공공서비스, 주거형 서비스, 교통 및 기타로 구분되어 진다. 각 유스케이스에는 관련한 유스케이스에 대한 설명과 시나리오, 여러 조건 및 가능한 요구사항들이 나와 있다. 아래의 표 〈2-1〉은 oneM2M의 TS-0001에 등록되어 있는 유스케이스를 보기 쉽게 구분한 내용을 보여주고 있다. 위에서 언급된 총 33개의 유스케이스 들에 대해 각각 상세히 알아보기로 하자.

표 〈2-1〉 oneM2M TS-0001의 유스케이스

| Industry Segment | oneM2M Use Cases | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Energy | Wide area Energy related measurement /control system for advanced transmission and distribution sutomation | Analytics for oneM2M | Smart Meter Reading | Environmental Monitoring for Hydro-Power Generation using Satellite M2M | Oil and Gas Pipeline Cellular /Satellite Gateway | | | |
| Enterprise | Smart building | | | | | | | |
| Healthcare | M2M Healthcare Gateway | Wellness services | Secure remote patient care and monitoring | | | | | |
| Public Services | Street Light Automation | Devices, Virtual devices and Things | Car/Bicycle Sharing Services | Smart parking | Information Delivery service in the devastated area | | | |
| Residential | Home Energy Management | Home Energy Management System | Plug-In Electrical Charging Vehides and power feed in home scenario | Real-time Audio/ Video Communication | Event Triggered Task Execution | Semantic Home Control | Semantic Device Plug and Play | |
| Transportation | Vehide Diagnostic & Maintenance services | Neighbourhood Alerting on Traffic Accident | Fleet management services using Digital Tachograph | | | | | |
| Other | Extending the M2M Access Network using Satellites | M2M data traffic management by underlying network operator | Optimizing connectivity management parameters with mobile networks | Optimizing mobility management parameters with mobile networks | Sleepy nodes | Collection of M2M system data | Leveraging Broadcasting/Multicasting Capability of Underlying Networks | Service Provisioning for Equipment with Built-in Device |

### 2.2.1. 에너지 분야 – 전송 고도화 및 자동 분배 기반 WAMS(Wide Area Measurement System)

전기시스템상의 상태측정과 전력 품질 관리를 담당하는 PMU(Phase Measurement Unit, Synchrophasors)는 HV(High Voltage) 전송 및 MV(Medium Voltage) 분산처리를 담당하며, PMU는 시간별/일별/이벤트 기반으로 방대한 양의 통계 정보를 생성

한다. PMU는 모바일 광대역 통신 네트워크(예. 모바일 통신망)를 활용하여 PMU에서 센터측에 데이터를 전달할 수 있음을 착안하여 원격지(remote site)측 TSO/DSO에서의 추가적인 내부 네트워크(internal network) 확장이 필요하지 않음을 보여 주는 유스케이스이다. 즉 소개된 유스케이스로 부터 oneM2M에서는 다음과 같은 요구사항을 도출하게 된다.

- ✓ 데이터 수집 및 보고(Reporting) 기능
- ✓ 기기의 원격제어
- ✓ 정보 수집 및 multiple application으로 정보 전달
- ✓ 데이터 저장 및 공유
- ✓ 보안 및 프라이버시
- ✓ 연결성 유지 (Continuous connectivity)

### 2.2.2. 에너지 분야 – Smart Meter Reading

본 유스케이스는 SGIP(Smart Grid Interoperability Panel)와 OpenSG 사용자 그룹이 북미에서 진행한 내용을 기반으로 하고 있으며, AMI network상에서의 actor(예. Smart Meter)와 actor(예. DAP, Data Aggregation Point)간의 통신에 있어서의 data flow 등을 보여 주고 있다. 요구사항에 대해서는 명시적으로는 아니지만, meta data에 대한 data aggregation, 저장, 추출 등에 대한 내용이 언급되어 있다.

### 2.2.3. Enterprise 분야 - 스마트 빌딩 (Smart Building)

본 유스케이스는 스마트 빌딩에 대한 일반적인 내용과 함께 빌딩 내 설치된 센서, 컨트롤러, 알람, 게이트웨이 등을 통한 자동 빌딩 관리에 대한 내용과 함께 조명관리, 침입 감지, 화재 알람에 대한 서비스 시나리오를 다루고 있다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ 그룹상에서의 디바이스간 또는 특정 상황에서의 디바이스간 chained-action 수행
- ✓ 위치 보고
- ✓ 기기의 그룹 지정 및 관리 (예. 기기 추가/제거 등)
- ✓ 그룹간 포함관계 지원

### 2.2.4. 헬스케어 분야 - 헬스케어 게이트웨이

본 유스케이스는 헬스케어 센서로부터 얻어지는 환자의 데이터를 backend 서버로 전송하는 헬스케어 게이트웨이에 대한 내용으로 게이트웨이와 backend 서버와의 양방향

통신도 다루고 있다. 이 둘간의 통신은 이동통신 망을 통한 연결 또는 이외에도 가능하며, 재난 재해 등으로 인해 일부 기능이 제공되지 않을 수 있다는 사항도 언급하고 있다. 도출된 요구사항은 다음과 같다.

- ✓ Back-end 서버로 센서 측정값을 전송하는 게이트웨이 지원
- ✓ 데이터 전송에서의 버퍼링 지원
- ✓ QoS 정책
- ✓ 로컬 게이트웨어의 이동통신 네트워크 정책 수용 지원
- ✓ 관심 데이터에 대한 구독(subscription) 기능(Subscriber-specific filter)
- ✓ 데이터 전송에서의 보안/인증
- ✓ Wakeup trigger 기능 지원

### 2.2.5. 공공 서비스 분야 - Street Light Automation

본 유스케이스는 조명을 컨트롤 하는 다양한 방식(센싱, 통신, 분석 등)을 통해 공공안전, 에너지 소비 및 CO2 절감, 그리고 조명 관리를 위한 유지 비용을 줄이기 위한 목적으로 도시 자동화의 일환으로 소개되고 있다.

조명 자동화의 유형별 사용 예는 다음과 같다.

- ✓ 개별 거리 조명
  - 조명 센서
  - Power quality 센서
  - 근거리 센서 (응급차량, 보행자 등)
- ✓ 관리 센터에 의한 자동화
  - 정책 기반
  - 주변 분석 기반 (일몰/일출, 날씨 등)
  - 예상 분석 기반
- ✓ 타 서비스 제공자 연계
  - 거리 신호등 서비스 (응급 차량 우선 등)
  - 응급 차량 서비스 (차량 우회 경로 등)
  - 전력 서비스 (Power overload 등)

다음은 이로부터 도출된 요구사항은 다음과 같다.

- ✓ 기기로부터의 정보 수집
- ✓ 기기로부터 수집된 정보를 어플리케이션으로 전달

- ✓ 기기 그룹 제어
- ✓ 어플리케이션 소프트웨어를 기기에 전송
- ✓ 어플리케이션 간 정보공유
- ✓ 과금 기능
- ✓ 통신 round trip 시간 측정 기능
- ✓ 보안

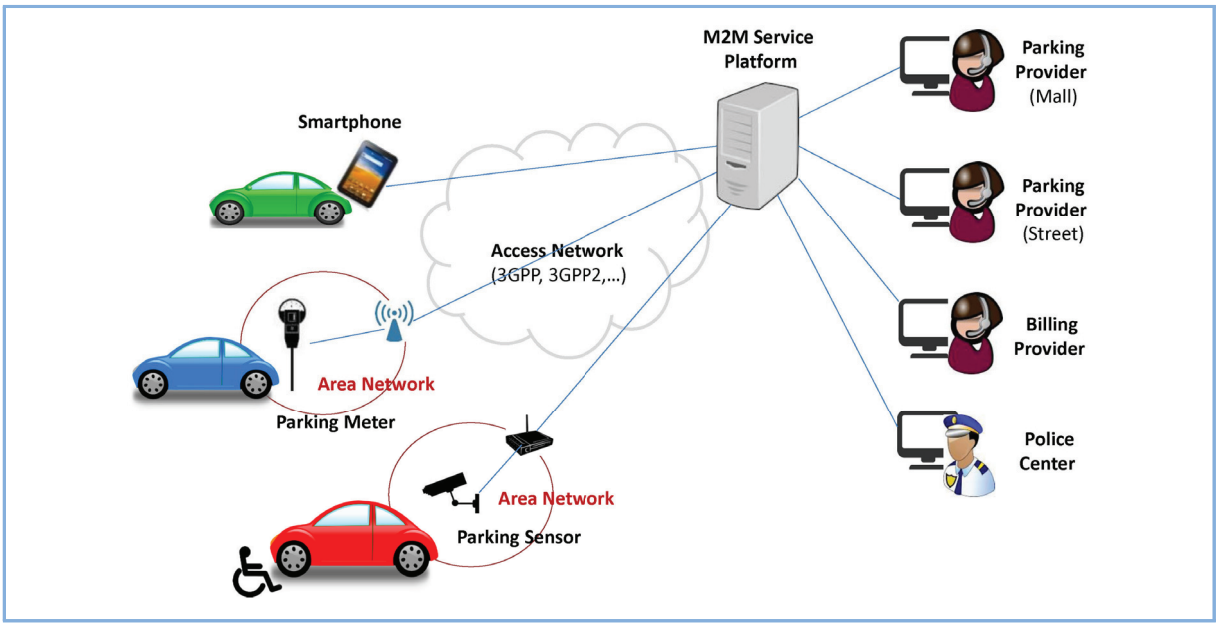



그림 〈2-2〉 스마트 파킹 유스케이스

### 2.2.6. 공공 서비스 분야 - 스마트 파킹 서비스

그림 〈2-2〉에서 보여주고 있는 유스케이스는 주차공간을 쉽게 찾고 불법주차 단속을 위한 스마트 주차 서비스이다. 사용자는 스마트 주차 서비스를 통해 쉽게 주차할 수 있으며, 주차 예약도 가능하다. 주차에 있어서는 first-come-first-served 개념이 일반적인 정책이나 VIP 또는 장애자는 주차 공간이 우선적으로 지급된다. 유스케이스로 부터 도출된 요구사항은 다음과 같다.

- ✓ 서로 다른 어플리케이션 제공자로부터 발생한 과금 데이터와 이력 관리 체계 지원
- ✓ 기기의 요구에 따른 다른 기기를 통한 트리거링 (Triggering) 기능

### 2.2.7. 공공 서비스 분야 -재해/재난지역에서의 정보 전달 서비스

본 유스케이스는 도심에서 재난 또는 재해 발생시, 사용자에게 교통상황이나 대피소 등의 필요한 정보를 신속하고 정확하게 제공하는 서비스에 관한 것으로 서비스 제공자는

현재 상황에 대한 데이터를 지속적으로 업데이트하고 사용자 디바이스에게 전달한다. 게이트웨이는 네트워크상의 트래픽 폭주를 고려하여 자체 DB에 사용자 디바이스로부터 받은 정보를 저장할 수 있다. 도출된 공통 요구사항은 다음과 같다.

- ✓ 어플리케이션 데이터에 대한 네트워크 어플리케이션과 게이트웨이 어플리케이션간의 동기화
- ✓ 게이트웨이에서의 데이터 로컬 저장 기능
- ✓ 게이트웨이의 인가된 디바이스에 reporting 가능 (WAN 제공이 안될 경우)
- ✓ 보안

### 2.2.8. 주거형 서비스 분야 – 홈 에너지 관리 서비스

본 유스케이스는 홈에서의 에너지 소비를 관리하기 위한 것으로, 사용자들은 매일 사용하는 에너지량에 대해 알 수 있고, 원격 제어 기능을 통해 홈기기를 제어할 수 있다. 특히, 홈에서의 에너지 정보 수집과 이를 M2M 시스템에 전달하는 EGW(Energy Gateway)에 대해서도 다루고 있다. 본 유스케이스에서 도출된 요구사항은 다음과 같다.

- ✓ 데이터 수집 및 보고(reporting) 기능
- ✓ M2M 기기 원격제어
- ✓ 정보 수집 및 다수의 어플리케이션으로 전송
- ✓ 데이터 저장 및 공유
- ✓ Privacy, 보안, 인증
- ✓ 데이터 온전성
- ✓ M2M 기기 및 게이트웨이 제어, 관리
- ✓ 서비스 가입 기능
- ✓ 트래픽 흐름제어

### 2.2.9. 주거형 서비스 분야 – HEMS (Home Energy Management System)

본 유스케이스는 HEMS(Home Energy Management System) 기술에 기반하는 서비스를 소개하고 있다. 여러 제조사의 다양한 가전이 LAN 또는 PAN을 통해 연결되고 게이트웨이 기기를 통해 제어된다. 게이트웨이 기기는 가전기기의 상태 정보를 수집하고 이를 관리 서버에 전송하며, 새로운 가전기기를 지원할 수 있도록 업그레이드 기능이 제공된다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ 새로 설치되는 가전기기 감지

- ✓ 가전기기를 자동으로 설정할 수 있는 pre-provisioning
- ✓ 추상적 객체 모델을 이용하여 다양한 판매자의 가전제품 지원
- ✓ 사용자가 직접 조작할 경우 HEMS의 제어를 override 할 수 있는 기능

### 2.2.10. 주거형 서비스 분야 - Plug-In Electrical Charging Vehicles and Power Feed in Home Scenario

본 유스케이스는 전기차량을 가정에서 충전하는 경우, 서로 다른 actor들에 대한 상호작용에 대한 시나리오를 보여주고 있다. 언급되고 있는 actor들로는 Electricity-Network service provider(Electricity-N/W-SP), Electric Vehicle Charging SP(EVC-SP), PEV service provider(PEV SP) 등이 있다. PEV는 load(에너지 소비 매체) 또는 전력 저장소(power storage)로 사용될 수 있다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ SW 업그레이드
- ✓ PEV 상태정보 수집
- ✓ 과금 정보 수집
- ✓ 수집된 데이터를 multiple application에 전송
- ✓ 데이터 저장 / 공유
- ✓ 프라이버시

### 2.2.11. 주거형 서비스 분야 - Semantic Home Control

본 유스케이스는 서로 다른 두 개의 어플리케이션들 간에 리소스의 시맨틱 정보를 이용한 협업(co-operation)에 대한 내용을 다루고 있으며, 이를 BEMS(Building Management System)과 HEMS(Home Energy Management System)의 사례를 들어 설명하고 있다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ 범용 시맨틱 데이터 모델(예. Ontology) 지원
- ✓ 시맨틱 정보에 기반한 M2M 리소스 검색 기능
- ✓ Real-world 엔티티(예. 방, 창문 등) 검색
- ✓ 추상화 디바이스와 실제 물리적 디바이스간의 제어 명령 매핑

### 2.2.12. 주거형 서비스 분야 - 시멘틱 디바이스 Plug and Play

본 유스케이스는 새로운 기기 등록될 경우, M2M 시스템의 시맨틱 정보를 이용하여 자체적으로 자신의 특징을 발견하고 이웃기기와 그 외 사물과의 관계를 설정한다는 내용이다. 예를 들어, 새로 설치된 기기들은 M2M 서비스 제공자에게 자신의 역할과 기능

그리고 시맨틱 정보를 제공하고, 새로 설치된 기기들의 시맨틱 정보와 기존 기기들의 시맨틱 정보를 비교하여 M2M 시스템 상에서 서로간의 관계(relationship)가 정립이 된다. 도출된 요구사항은 다음과 같다.

- ✓ 버티컬 산업에 범용적으로 적용될 수 있는 시맨틱 데이터 모델
- ✓ M2M 객체는 M2M 시스템에게 자신의 시맨틱 정보(description)을 제공
- ✓ 사물간의 시맨틱 관계

### 2.2.13. 차량 서비스 분야 - 차량 진단 및 관리 서비스

본 유스케이스에서 다루어지고 있는 차량 서비스 센터는 차량 소유자에게 주기적인 차량 관리 상태를 제공하여 차량 소유주로 하여금 차량 유지 관리를 하게 함으로써 차량에 발생할 수 있는 잠재적인 문제를 예방할 수 있다는 내용을 담고 있다. 도출된 요구사항은 다음과 같다.

- ✓ 어플리케이션과 M2M 시스템간 주기적 진단(diagnostic) 데이터 교환
- ✓ 알림(notification) 주기 설정
- ✓ 진단을 위한 구문(syntax)과 시맨틱 포맷

### 2.2.14. 차량 서비스 분야 - 차량과 차량간 원격 관리 서비스

본 유스케이스는 차량 원격관리 서비스를 소개하고 있으며, 운전자의 생명과 직접적으로 관련되는 차량 무결성에 대해 강조하고 있다. 측정 과정을 보증하기 위해서 M2M 게이트웨이의 HSM(Hardware Security Module)을 사용할 수 있으며, 원격 관리라는 측면에서 본다면 의료 기기, 가전, 금융 터미널 및 산업제어 기기 등에 적용될 수 있다. 도출된 요구사항은 다음과 같다.

- ✓ 차량 무결성 확인 및 SW/HW/firmware 요소의 설치 프로세스
- ✓ 디바이스 키를 이용한 인증 지원
- ✓ HSM(Hardware Security Module) 지원

### 2.2.15. 차량 서비스 분야 -DTG(Digital Tachograph)기반 Fleet Management Service

본 유스케이스는 디지털 운행 기록계(DTG – Digital Tachograph) 기반 서비스에 내용으로 DTG를 활용한 차량 관련 서비스에 전반적인 내용을 언급하고 있다. DTG는 운행속도, RPM, 브레이크 상태, 운행거리 등의 정보를 제공하는데, 이 정보들은 게이트웨이와 관리 서버로 전송이 되어 차량 위치 추적 및 차량 상태 등 원격 모니터링에 사용된다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ 프로비저닝, 설치, 설정, 등록 방법

- ✓ DTG/FMS 데이터 저장 방식 및 전송 프로토콜
- ✓ 차량 위치 기반 서비스 방법
- ✓ 무선기반 제어, 설정, 오류기록, 관리 방법

### 2.2.16. 기타 서비스 분야 – 데이터 트래픽 관리

본 유스케이스는 기저 네트워크 (underlying network)의 데이터 트래픽 상태에 따라 데이터 패킷의 손실을 방지하기 위해 M2M 데이터 트래픽을 제어 및 관리가 필요하다는 내용을 담고 있다. 기저 네트워크로부터 관련 notification을 수신할 경우, M2M 서비스 플랫폼과 Application 제공자는 설정을 변경하여 데이터 전송 간격 조절, 데이터 전송 정지 등을 수행한다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ 기저 네트워크로부터 데이터 트래픽 상태 정보
- ✓ 수신 및 관련 정보를 어플리케이션 서버에 제공
- ✓ 기저 네트워크 데이터 트래픽 상태에 따른 데이터 전송 제어 및 Device에서의 어플리케이션 행동 제어

### 2.2.17. 기타 서비스 분야 – 모바일 네트워크 연결에 있어서의 최적화된 데이터 전송

본 유스케이스는 서비스 계층에서의 데이터 전송 주기 간격의 변화 감지와 이를 M2M 서비스 플랫폼과 모바일 네트워크간의 인터워킹을 통해 통보하는 방법을 설명하고 있다. 모바일 데이터 어플리케이션은 작은 데이터 패킷을 전송하는데, 이러한 데이터의 잦은 전송은 idle/connected state 변화를 자주 일으켜 네트워크 부하를 가중시키기 때문이다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ M2M 서비스 플랫폼은 underlying network에게 M2M 기기와 관련된 정보 제공 (underlying network 최적화)
- ✓ M2M 서비스 플랫폼은 M2M 어플리케이션으로 부터 받은 데이터에 대한 분석 수행

### 2.2.18. 기타 서비스 분야 – Broadcasting 및 Multicasting 관리

본 유스케이스는 기저 네트워크의 broadcasting/multicasting 기능을 활용하여 효율적인 메시지 전파에 대한 언급을 하고 있다. 이를 통해 교통 사고 발생 시 사고가 발생했음을 인접 차량들에게 알려 2차 사고 및 교통혼잡을 예방할 수 있다. 하지만 다양한 통신 네트워크(3GPP, 3GPP2, WiMax, WiFi)가 존재할 수 있기 때문에 기저 네트워크의 broadcasting/multicasting 기능을 사용하기가 쉽지 않지만 규격화된 인터페이스를 제공함으로써 이를 해결할 수 있다. 이로부터 도출된 요구사항은 다음과 같다.

- ✓ Underlying network의 broadcast/multicasting 기능을 최대한 활용
- ✓ 특정 지역에 broadcast/multicast 메시지 전송 요청 기능 지원
- ✓ Broadcast/multicast 메시지 요청을 위한 authentication, authorization, accounting
- ✓ 기저 네트워크로부터 broadcast/multicast 기능 관련정보 수신

### 2.2.19. 기타 서비스 분야 - 서비스 프로비저닝

본 유스케이스는 M2M 서비스 프로비저닝에 대한 사항을 다루고 있으며, 특히 광의의 개념의 M2M 서비스 프로비저닝은 M2M 서비스 프로비저닝과 네트워크 서비스 프로비저닝으로 구성된다고 하고 있다. M2M 서비스 플랫폼이 이동 네트워크와 관련된 식별자(IMSI, MSISDM, MDN 등)를 관리해 줌으로써 underlying mobile network의 복잡성을 간단히 표현할 수 있는데, network 지원상태를 'available', 'not available'로 표현하는 것이 그 예이다. 본 유스케이스로 부터는 다음과 같은 공통 요구사항이 도출되어졌다.

- ✓ 기기의 M2M 서비스 상태 파악 및 관리
- ✓ 기기 관련 네트워크 식별자의 네트워크 서비스 관리 상태 파악
- ✓ Mobile network identifier의 네트워크 서비스 관리 상태 변경시 기저 네트워크에게 통보
- ✓ 기기의 M2M 서비스 관리 상태가 변경될 경우 M2M 어플리케이션에게 통보
- ✓ 기기의 M2M 서비스 운영 상태 변경

## 2.3. oneM2M 요구사항

앞서 살펴본 oneM2M 유스케이스를 기반으로 Overall System 요구사항, Management 요구사항, 추상화, 보안 등 8개 분야 총 142개의 기능적인 요구사항이 정리가 되었으며 다음과 같다.

### 2.3.1. Overall System 요구사항

사물인터넷 서비스를 제공하기 위해 기본적으로 충족되야 하거나 고려되야 하는 항목을 정리한 부분으로, 총 항목수는 72개이며, 데이터 전송, 통신방식, 응용과의 상호 연동, 권한 및 데이터 저장, 그룹관리 등이 있다. 주요 내용은 다음과 같다.

- ✓ 데이터/정보의 저장과 검색과 관련된 정책과 관리 방안
- ✓ 사물인터넷 시스템의 응용과 데이터/정보와의 상호 연동
- ✓ 통신망이 제공하는 서비스 노출, 기저 네트워크가 제공하는 서비스 재사용
- ✓ M2M 어플리케이션 간 상호 작용
- ✓ 서로 다른 M2M Service Provider들이 관리하는 어플리케이션 간 연동

- ✓ 단일 또는 다수의 M2M 어플리케이션과 단일 또는 다수의M2M Device/Gateway와의 상호 작용 지원
- ✓ Continuous/Non-continuous connectivity 지원
- ✓ 다수의 통신 패턴(예. 대용량 데이터 전송, 스트리밍 통신 등) 지원
- ✓ 이종 M2M Area Network 지원
- ✓ 서비스 구독(subscription) 관리
- ✓ Actuator 구동, 센서 제어 기능
- ✓ 기기의 그룹/멤버십(membership) 관리

### 2.3.2. Management 요구사항

해당 요구사항은 디바이스와 게이트웨이의 관리 기능에 해당하는 부분으로, 총 항목 수는 17개이며, 디비이스 또는 게이트웨이의 환경설정, 모니터링, 진단 등이 있다. 주요 내용은 다음과 같다.

- ✓ M2M Gateway/Device의 관리 및 설정 기능
- ✓ M2M Area Network 발견 기능
- ✓ 서로 다른 관리 기술(예. OMA DM, BBF TR069)을 통해 구동된 기기 관리
- ✓ 다수의 기기에 대한 그룹 관리
- ✓ M2M Area Network의 M2M 기기에 대한 모니터링/진단, 리부팅/재설정 기능
- ✓ M2M Area Network에 연결된 기기의 software management 기능
- ✓ M2M Area Network 사용승인 관련 기능
- ✓ M2M Area Network에 연결되 기기들의 topology를 수정할 수 있는 기능
- ✓ M2M Device의 M2M Service 상태 관리
- ✓ 기기에 기록된 이벤트, 정보 로그(log) 회수
- ✓ Firmware 업데이트 및 관리

### 2.3.3. 추상화 요구사항

추상화와 관련한 사물인터넷 요구사항은 총3개이며, 세부 항목은 다음과 같다.

- ✓ 데이터 표현(representation)을 위한 범용 구조(generic structure) 제공
- ✓ M2M 어플리케이션, M2M Device/Gateway 그리고 기타 기기에서 사용되는 정보 모델(Information Model) 간 변환기능(translation mechanism) 제공
- ✓ 가상 기기와 사물을 나타낼 수 있는 기능 제공

### 2.3.4. 시맨틱 요구사항

사물인터넷 서비스는 시맨틱 관계 및 정보(description)을 기반으로 mash-up, 분석 등이 가능하다. 요구사항으로 정의된 총 항목수는 7개 이며, 세부 항목은 아래와 같다.

- ✓ Resource와 M2M 어플리케이션의 semantic description 관리(예. create, retrieve, update, delete)
- ✓ Semantic description을 위한 common modeling language 지원
- ✓ 이종 modeling 언어 간 상호 연동(interworking) 기능 제공
- ✓ Semantic description 기반 M2M Resource 발견
- ✓ M2M System 영역 밖에 있는 semantic description을 access할 수 있는 기능
- ✓ Semantic description에 기반한 M2M 데이터 분석(analytics) 기능
- ✓ Semantic mash-up 기능 제공(가상 기기 생성, 신규 M2M 서비스 제공 등)

### 2.3.5. 보안 요구사항

보안 요구사항은 데이터 인증, 무결성, Dos 공격 등 디바이스, 네트워크, M2M시스템, 어플리케이션의 전반에 대한 보안을 다루고 있으며, 요구사항으로 정의된 총 항목 수는 26개이다. 주요 내용은 다음과 같다.

- ✓ DOS(Denial of Service), 위장(Impersonation), Replay 공격에 대한 보호체계
- ✓ 데이터의 비밀유지(confidentiality), 온전성(integrity) 보장
- ✓ USIM(Universal Subscriber Identity Module)/UICC(Universal IC Card) 그리고 네트워크 계층 보안 연계 체계
- ✓ 기저 네트워크, M2M Service, M2M 어플리케이션 서비스 간 상호 인증 체계
- ✓ 서비스 비승인 접근 시도에 대한 대책/보호 조치
- ✓ 보안 증명서(security credential) 제공(provisioning) 및 관리
- ✓ 보안 증명서 관련 남용, 복제, 절도 등에 대한 대책/보호 조치
- ✓ oneM2M Device의 HW, SW, firmware 온전성 확인을 위한 장치
- ✓ HSM(Hardware Security Module) 사용
- ✓ 지리적 위치 정보에 대한 비밀유지 체계

### 2.3.6. 과금 요구사항

과금과 관련한 사물인터넷 요구사항은 총6개이며, 세부 항목은 다음과 같다.

- ✓ 과금과 관련된 정보수집 기능(자원사용 정보)

- ✓ 서비스와 관련되어 수집된 과금 정보를 쉽게 연계할 수 있는 기능
- ✓ 차별화된 QoS가 적용된 데이터 사용에 대한 과금 데이터 기록 조정
- ✓ 기저 네트워크가 제공하는 과금체계 재사용
- ✓ 과금 정보를 청구(billing)영역으로 전달
- ✓ 과금 관련 이벤트 생성

### 2.3.7. 운용 요구사항

운용과 관련한 사물인터넷 요구사항은 총6개이며, 세부 항목은 다음과 같다.

- ✓ M2M 어플리케이션 모니터링 및 진단 기능
- ✓ M2M 어플리케이션에 대한 Software Management 기능
- ✓ M2M 어플리케이션 실행상태 제어(start, stop, restart)
- ✓ 기저 네트워크를 이용한 트래픽(traffic) 스케줄링(인터페이스 제공 시)
- ✓ M2M Device/M2M Gateway의 사용, 트래픽 특징 등에 대하여 M2M Application과 정보교환
- ✓ M2M Device/M2M Gateway의 사용, 트래픽 특징 등에 대하여 기저 네트워크와 정보교환

### 2.3.8. 통신 요청 처리 요구사항

통신 요청과 관련한 사물인터넷 요구사항은 총5개이며, 세부 항목은 다음과 같다.

- ✓ M2M 어플리케이션에게 통신 서비스를 제공하는 M2M Gateway와 M2M Device의 메시지 버퍼링(buffering)
- ✓ 버퍼링 된 메시지 전달(forwarding)
- ✓ QoS 파라미터, 우선순위 등이 적용된 M2M 어플리케이션의 통신 요청 수용
- ✓ 여러 소스로부터 전달되는 다수의 메시지를 M2M Gateway/M2M Device 내에서 동시에 처리
- ✓ M2M Session과 관련된 context 유지

oneM2M 요구사항 정리 당시 많은 논의가 있었지만, 그 중의 하나가 용어 정리 부분이었다. 세계 각국의 맴버사와 서로 다른 이해관계자들이 모였기 때문에, 저마다의 용어 사용으로 용어 통일하는 데에 많은 논의가 있었고, 그 결과 TS-0011의 Definitions and Acronyms에서와 같이 범용적이고 통일된 용어로 정리할 수 있었다.

oneM2M 요구사항문서를 포함하여 architecture 등 추후 다루어질 규격에서 TS-0011인 Definition and Acronym을 참조한다면 많은 도움이 될 것이다.

# 3. 사물 인터넷 표준 플랫폼이 왜 중요한가?

## 관련 oneM2M 표준 문서

TS-0001: FUNCTIONAL ARCHITECTURE

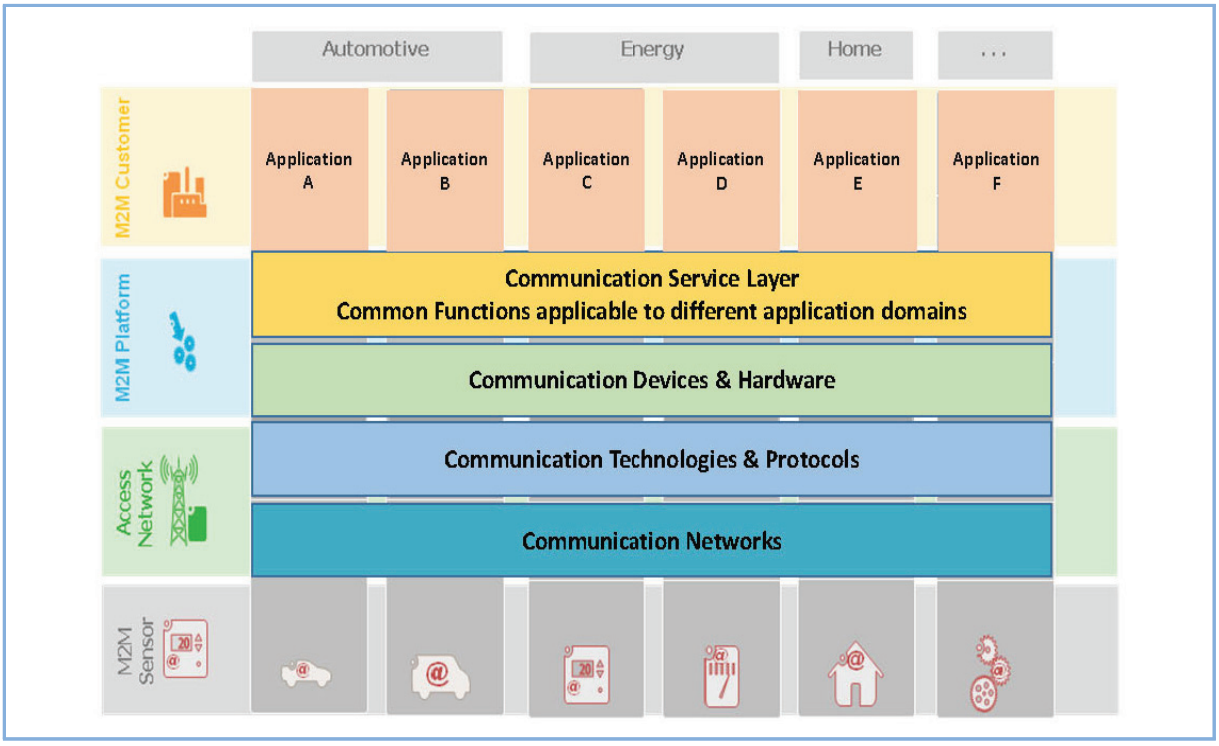



그림 〈3-1〉 사물인터넷 공통 플랫폼이 위치하는 영역

oneM2M 사물인터넷 플랫폼은 제 1장에서 설명한 바와 같이 M2M 센서, 액세스 및 코어 네트워크, M2M 플랫폼, M2M 어플리케이션의 수직적 모델 구조에서 하부의 센서, 디바이스 및 네트워크 기술은 상당수 기존의 기술을 활용하는 구조를 갖는다. oneM2M 단체에서 표준화를 진행하고 있는 대상은 다양한 버티컬 어플리케이션에서 요구되는 공통 기능을 지원하는 M2M 플랫폼 영역이다.

해당 oneM2M에서 개발하고 있는 공통 플랫폼은 Common Service Entity (CSE) 라는 용어로 정의되는데 즉, 다양한 어플리케이션을 위한 공통 서비스 기능을 제공할 수 있는 계층이라는 의미를 갖는다. CSE는 물리적으로 서비스 프로바이더가 제공하는 서버 인프라 시스템 뿐만 아니라 센서, 디바이스에 올라가는 미들웨어 스택 및 액세스 네트워크와 연결하는 게이트웨어의 미들웨어 스택으로 구현이 가능하다. 즉 oneM2M의 표준화 대상은 서버, 게이트웨이, 디바이스에 탑재될 수 있는 미들웨어 스택에 대한 표준화를 진행하는

것이고 이를 통해서 서버, 게이트웨이, 디바이스 전 영역을 기반으로 사물인터넷 서비스를 제공하고자 하는 것이다.

공통플랫폼을 소프트웨어 레이어의 관점에서 보면 해당 Common Service Entity (CSE)는 M2M 어플리케이션 프로그램과 기기의 오퍼레이팅 시스템 소프트웨어 사이에서 동작되는 소프트웨어로 정의할 수 있다. 다양한 버티컬 어플리케이션에서 요구되는 기능들을 제공하기 위한 수평적 플랫폼의 역할을 담당하며 해당 기능으로서는 원격 설정, 동작지시, 연결, 데이터 수집, 데이터 보관, 디바이스 관리, 보안등의 공통적인 기능을 제공함으로써 M2M 디바이스 및 M2M 어플리케이션 사이에서 서비스 제공을 위한 종단간 데이터 전달 및 컨트롤 서비스를 지원한다.

## 3.1. oneM2M 표준 플랫폼의 장점

기존의 서비스 프로바이더 및 어플리케이션 종속적인 플랫폼과 비교하여 oneM2M 수평적 플랫폼의 장점은 다음과 같다.

- ✓ 기존에 각 서비스 프로바이더 별로 그리고 어플리케이션 별로 종속적으로 개발되었던 사물인터넷 플랫폼의 파편화를 방지할 수 있다. 그리고 수평적 공통플랫폼의 도입으로 규모의 경제를 통한 사물인터넷 산업의 활성화를 제공할 수 있다.
- ✓ 수평적 공통플랫폼의 도입은 사물인터넷 어플리케이션 서비스 개발의 복잡성을 낮추어 주는 효과가 있다. 이는 공통 플랫폼을 통해서 어플리케이션이 요구하는 다양한 기능들을 공급할 수 있기 때문이고 이를 통해서 서비스 프로바이더는 서비스 개발 시에 개발 비용을 감소시킬 수 있다.
- ✓ 수평적 공통 플랫폼은 농업, 물류, 의료, 자동차, 가전 등 서비스 및 산업 환경에 구애받지 않으며 소프트웨어적으로 구현된 컴포넌트를 재사용할 수 있기 때문에 이를 통한 개발 비용과 더불어 Trouble Shooting을 위한 운영비용을 감소시킬 수 있다.
- ✓ 표준화된 수평적 공통플랫폼은 하부의 네트워크를 액세스하고 해당 네트워크에서 제공하는 서비스를 활용하는 부분에서 전세계 네트워크를 유용하도록 설계되므로 서비스 및 사업기회의 확장성을 제공해 준다.
- ✓ 사물인터넷 공통플랫폼에 대한 운영적인 측면에서 표준화된 기술은 관련 산업 및 서비스 프로바이더 관계자들에게 공통의 이해도를 제공하고 따라서 기술에 대한 운영측면에서 문제발생에 대한 대처 및 문제에 대한 보편화된 솔루션을 제공하므로 운영비용 감소를 가져올 수 있다.

즉, oneM2M 표준 플랫폼을 통한 장점은 다음 세가지 키워드로 정리할 수 있다. 사물인터넷 플랫폼의 파편화 방지, 이를 통한 개발 및 운영 비용감소, 이를 통한 산업 활성화이다. 그리고 이러한 장점을 기반으로 oneM2M 사물인터넷 공통 서비스 표준 플랫폼이 개발된 것으로 볼 수 있다.

## 3.2. oneM2M 공통 플랫폼 아키텍쳐 모델

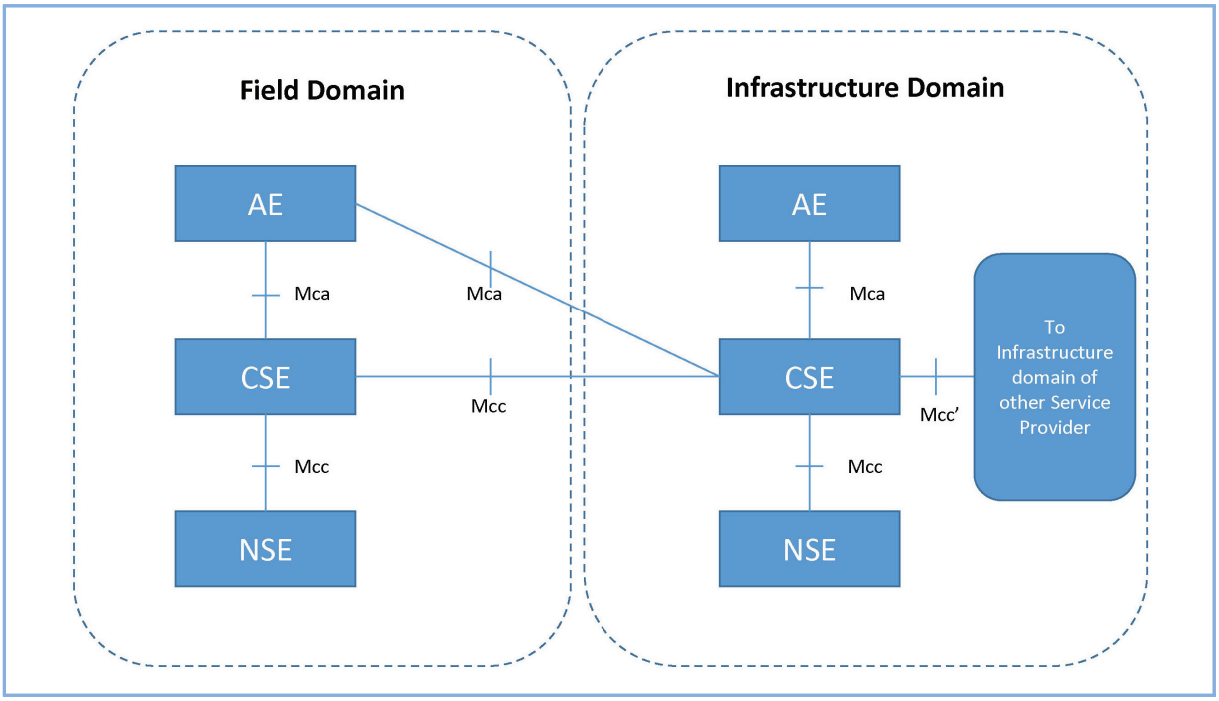



그림 〈3-2〉 oneM2M 공통플랫폼 기능 모델

oneM2M 공통 플랫폼은 레이어 아키텍쳐 스타일로서 표현하면 상단의 M2M 어플리케이션 레이어와 하단의 M2M 네트워크 서비스 레이어 사이에 위치하고 있으며 네트워크 서비스 레이어가 제공하는 네트워크 서비스를 유용하면서 M2M 기반 다양한 어플리케이션들에게 공통 기능을 제공하기 위한 공통 서비스 레이어로 표현될 수 있다.

또한 그림 〈3-2〉와 같이 oneM2M 플랫폼 아키텍쳐는 기능 엔티티 모델로 표현이 가능하며 센서, 액츄에이터, 디바이스, 게이트웨이가 위치하는 필드 도메인 영역과 서비스 프로바이더의 서버가 위치하는 인프라스트럭쳐 도메인 영역으로 구분이 가능하다. 또한 각각의 도메인에서 레이어 아키텍쳐 모델에서 살펴본것과 같이 어플리케이션 레이어인 Application Entity (AE), 공통 서비스 플랫폼 레이어인 Common Service Entity (CSE), 네트워크 서비스 레이어인 Network Service Entity (NSE)로 분리하여 기능 모델을 정의하고 있다.

그림 〈3-2〉에서와 같이 필드 도메인 및 인프라스트럭쳐 도메인에서 위치하는 기능 엔티티라고 명명되는 각각의 엔티티에 대한 정의는 다음과 같다.

- ✓ Application Entity (AE): M2M 서비스를 제공하기 위한 어플리케이션 기능 로직을 의미며 각각의 AE는 유일한 AE-ID로 구별된다. AE에 대한 예로서는 관제시스템, 스마트그리드 시스템, 헬스케어 시스템을 위한 어플리케이션으로 매칭이 가능하다.
- ✓ Common Service Entity (CSE): oneM2M 공통 서비스 플랫폼의 공통 서비스기능을 제공하는 부분으로서 컴퓨터 시스템으로서는 미들웨어 소프트웨어에 해당한다. 해당

CSE에는 12개의 Common Service Function (CSF) 공통 서비스 기능을 포함하고 있으며 해당 공통 기능은 참조 포인트를 통해서 외부에 노출되어 서비스를 제공할 수 있다. 해당 CSE는 AE를 구별하는 방식과 마찬가지로 CSE-ID를 통해서 유일하게 식별 가능하다.

- ✓ Network Service Entity (NSE): CSE가 위치한 미들웨어의 하부 네트워크 서비스에 대한 추상화 영역으로 CSE에게 네트워크 서비스를 제공한다. 제공 가능한 네트워크 서비스의 예로서는 디바이스 관리, 위치관리, 디바이스 트리거링 서비스 등이며 해당 서비스를 위해서는 네트워크 서비스가 관여된다.

또한 그림 〈3-2〉에서 나타난 각각의 엔티티들간의 인터페이스를 의미하는 참조 포인트는 'Mca, Mcc, Mcn, Mcc''으로 4개의 참조 포인트가 정의되고 있으며 해당 참조포인트를 통해서 AE, CSE, NSE 들 간의 연동이 이뤄진다.

- ✓ Mca (M2M Communication with AE) 참조 포인트: AE 와 CSE 간의 연동 포인트를 가리키며, 해당 AE가 CSE에서 제공하는 공통 서비스 기능을 이용하기 위한 API 의 연결 포인트이고, CSE 와 AE간의 통신을 위한 연결 포인트이다.
- ✓ Mcc (M2M Communication with CSE) 참조 포인트: 두 개의 CSE간의 연동 포인트를 가리키며, CSE와 다른 CSE간의 서비스 공개 및 통신을 가능하게 하는 연결 포인트이다.
- ✓ Mcn (M2M Communication with NSE) 참조 포인트: CSE와 NSE간의 연동 포인트를 가리키며, CSE가 NSE에서 제공되는 네트워크 서비스 기능을 이용할 수 있는 연결 포인트이면서 네트워크 망으로의 데이터 전달 연결 포인트이다.
- ✓ Mcc' (M2M Commmunication with CSE of different M2M Service Provider) 참조 포인트: 서로 다른 서비스 프로바이더에 종속적인 CSE간의 연동 포인트를 가리키며, 서비스 프로바이더 간 CSE사이의 서비스 공개 및 통신을 지원하는 연결 포인트이다.

oneM2M 아키텍쳐 모델은 필드 도메인, 인프라스트럭쳐 도메인에 위치하는 CSE와 AE의 역할 및 기능에 대한 구분을 기반으로 4종류의 노드타입을 정의하고 있다. 이는 해당 그림 〈3-3〉에 oneM2M 노드 구성모델에 표시되어 있다.

- ✓ Infrastructure Node (IN): 인프라스트럭쳐 도메인에 위치하고 있는 IN-CSE를 포함하는 논리적인 기기이다. 해당 IN은 서비스 프로바이더 당 한 개의 IN을 지원하는 것으로 정의되며 IN은 한 개의 CSE와 0개 이상의 AE를 포함하는 형태로 구성된다. 논리적 기기인 IN에 매칭되는 물리적 기기로는 서버를 예로 들 수 있다. IN은 Mcc 참조 포인트를 통해서 한 개 이상의 MN과 한 개 이상의 ASN과 연동되며 Mca 참조포인트를 통해서 한 개 이상의 ADN과 연동될 수 있다. Mcc'을 통해서 다른 서비스 프로바이더 영역에 위치한 IN과 연동이 지원된다.

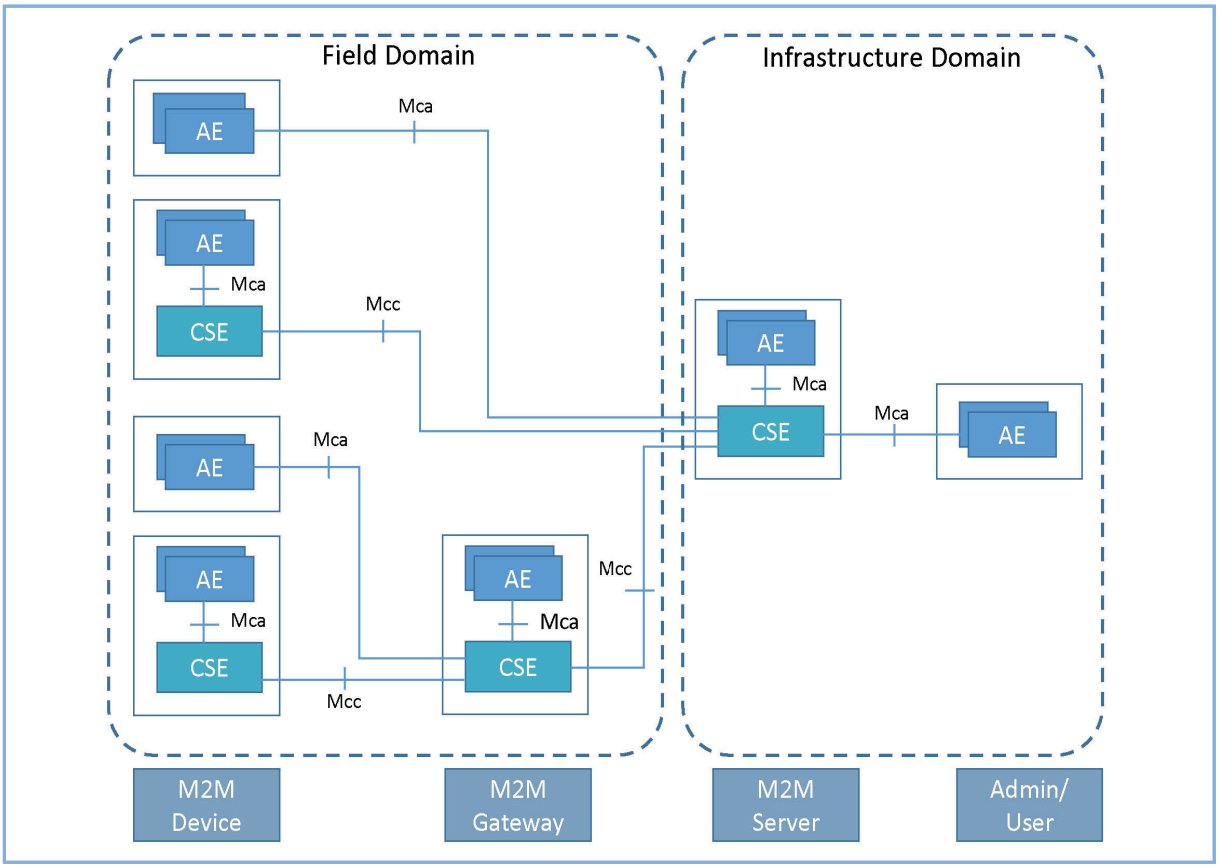



그림 〈3-3〉 oneM2M 노드 구성 모델

- ✓ Middle Node (MN): 필드 도메인에 위치한 MN-CSE를 포함하는 논리적 기기이다. 해당 MN은 한 개의 CSE와 0개 이상의 AE 를 포함하는 형태로 구성된다. 논리적 기기인 MN에 매칭되는 물리적 기기로는 게이트웨이를 예로 들 수 있다. MN은 Mcc 참조포인트를 통해서 적어도 하나의 IN 또는 MN과 연동되며 Mcc 참조포인트를 통해서 ASN과 연동 및 Mca 참조포인트를 통한 ADN과 연동될 수 있다.
- ✓ Application Service Node (ASN): 필드 도메인에 위치한 ASN-CSE와 ASN-AE를 포함하고 있는 논리적 기기이다. 해당 ASN은 한 개의 CSE와 1개 이상의 AE를 포함하는 형태로 구성되며 논리적 ASN에 매칭되는 물리적 기기로는 M2M 디바이스를 예로 들 수 있다. ASN 은 Mcc 참조포인트를 통해서 한 개의 MN 또는 한 개의 IN에 연동된다.
- ✓ Application Dedicated Node (ADN): 필드 도메인에 위치한 ADN-AE를 포함하고 CSE를 포함하지 않는 논리적 기기이다. 즉 해당 ADN는 CSE가 없고, 1개 이상의 AE를 포함한다. 논리적 ADN에 매칭되는 물리적 기기로는 센서 및 액츄에이터와 같은 자원 제약적인 M2M 디바이스를 예로 들 수 있다. ADN은 Mca 참조포인트를 통해서 MN 또는 IN과 연동되는 구조를 갖는다.

## 3.3. oneM2M 공통 서비스 기능 작업

앞서 설명한 바와 같이 oneM2M 공통 서비스 플랫폼 개발과 관련하여 아키텍쳐 표준 문서에서 다루고 있는 부분은 Common Service Entity (CSE)에 대한 기능들을 표준화 하는 것이다. 그리고 해당 기능들은 oneM2M이 지향하고 있는 리소스 기반 아키텍쳐 구조를 기반으로 각각의 리소스 타입으로 해당 공통 서비스 기능들이 표현되고 있다. oneM2M 표준화 작업 진행방식은 전체적으로 유스케이스 및 요구사항을 다루고, 이를 통해 필요한 기능들을 도출해 내어 시스템 아키텍쳐를 디자인하고, 상세 프로토콜을 개발하는 3GPP 표준화 방식에서 취한 Stage 기반의 접근방식으로 표준 개발을 진행하였다.

이러한 방식을 바탕으로 Stage 1에서는 공통 플랫폼이 지원해야 하는 다양한 서비스 어플리케이션에 대한 유스케이스를 취합하는 작업들이 진행되었으며 Stage 2에서는 Stage 1에서 도출된 요구사항을 만족할 수 있는 기능 요소들을 정의하고 해당 기능을 표현할 수 있는 리소스 타입 및 메시지 플로우를 정의하는 작업들이 진행되었다. Stage 3에서는 아키텍쳐를 고려한 RESTful 스타일의 서비스 계층 프로토콜 디자인 및 서비스 계층과 하위 계층 간의 프로토콜 바인딩을 정의하였다. 그리고 보안 및 디바이스 관리, 추상화, 시맨틱스 작업은 Stage 1, 2, 3 전반에 걸쳐 표준화가 진행되었다.

## 3.4. oneM2M 공통 서비스 기능

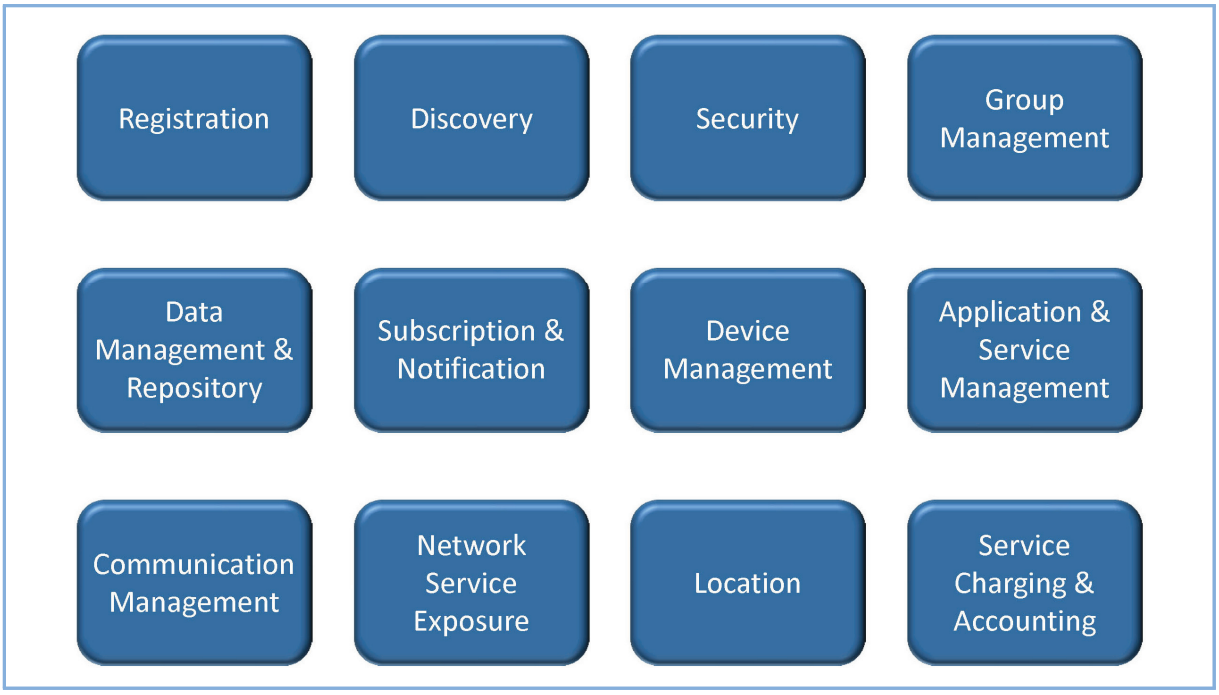



그림 〈3-4〉 oneM2M 공통서비스 플랫폼의 공통 서비스 기능

앞장에서 살펴본 oneM2M Stage 작업결과 Stage 2에서 oneM2M 공통 서비스 기능이 도출 되었으며 해당 공통 서비스 기능을 oneM2M 리소스 정보 모델 기반 아키텍쳐에 따라 리소스 타입으로서 표현 개발 하였다. oneM2M 공통 서비스 기능은 총 12개로 정의되며 해당 기능에 대한 이해와 이에 대한 리소스 타입에 대한 이해가 oneM2M 아키텍쳐 표준 문서의 핵심부분이다. 그림 〈3-4〉에서는 oneM2M CSE에 위치한 12개의 공통 서비스 기능 (Common Service Function)을 도식화 하고 있으며 각각의 기능은 리소스 타입으로 표현되고 Mcc, Mca, Mcn 참조 포인트를 통해서 서비스가 제공된다.

### 3.4.1. Registration (REG) CSF

- ✓ REG CSF는 AE와 CSE 또는 CSE와 CSE 간의 등록을 담당하며, 이러한 등록 관계를 통해서 oneM2M 엔티티간의 접속 및 접근이 가능하며 oneM2M 엔티티간의 데이터 전달을 통한 oneM2M 서비스 구성이 가능해진다.
- ✓ AE 또는 CSE가 상대 CSE와 등록 관계를 통해서 맺어진 관계에 대해서 등록을 요청한 엔티티는 Registree, 등록을 받아주어 관련 정보를 생성한 상대 엔티티를 Registrar로 표현 한다. 그리고 실제 AE, CSE 간의 요청 메시지 흐름은 Registree에서 Registrar로 전달되는 구조를 갖는다. 통신을 통한 데이터 전달/획득을 위해서 등록이 선행되어야 한다.

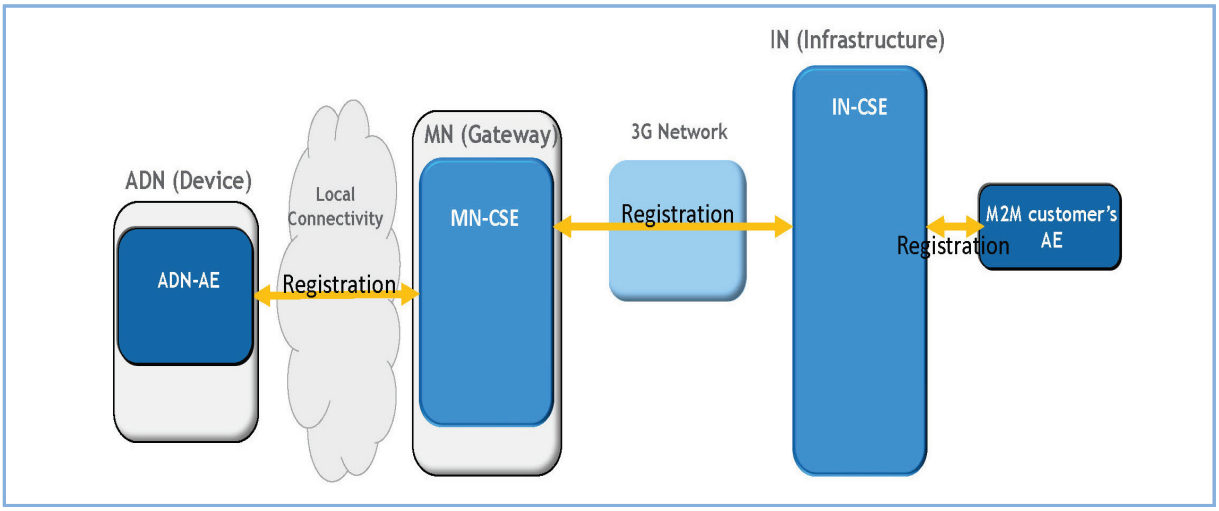



그림 〈3-5〉 oneM2M 엔티티간의 Registration

- ✓ 그림 〈3-5〉에 표현된 것과 같이 ADN-AE는 MN-CSE에 등록 요청을 하며 해당 MN-CSE는 등록 요청 메시지를 처리하여 ADN-AE와 MN-CSE 사이에 관계를 맺는다. MN-CSE와 IN-CSE관계와 M2M Customer AE와 IN-AE의 관계도 Registree-Registrar 관계가 형성된다.

### 3.4.2. Communication Management and Delivery Handling (CMDH) CSF

- ✓ CSE간, AE와 CSE간, 데이터 전달시에 NSE를 통한 데이터 전달 서비스를 제공하는 기능을 담당한다. CMDH CSF는 메시지 전달을 위해서 언제 보낼 지 그리고 어떤 네트워크 연결을 활용하여 보낼지를 결정한다. 또한 메시지를 바로 보낼 수 없거나 특정 설정에 의해서 메시지를 모아서 전달하도록 요청 되면 해당 메시지에 대해서 메시지 버퍼링 및 Aggregation 전달을 지원한다.
- ✓ CMDH CSF는 Policy 기반의 데이터 요청 및 전달을 지원하며 필드 도메인에 위치한 AE 또는 CSE가 서비스 프로바이더의 서비스에 가입할 때 관련 Policy를 표시함으로써 설정될 수 있다.
- ✓ 그림 〈3-6〉과 같이 Mcc 참조 포인트를 통한 데이터의 전달 시에 CMDH CSF가 관여하여 어떤 기저 네트워크를 활용할 지 또는 어떠한 서비스 레이어 관점의 데이터 전달 경로를 설정할지를 결정하며 데이터의 Store-and-Forward 기반의 데이터 전달을 지원한다.

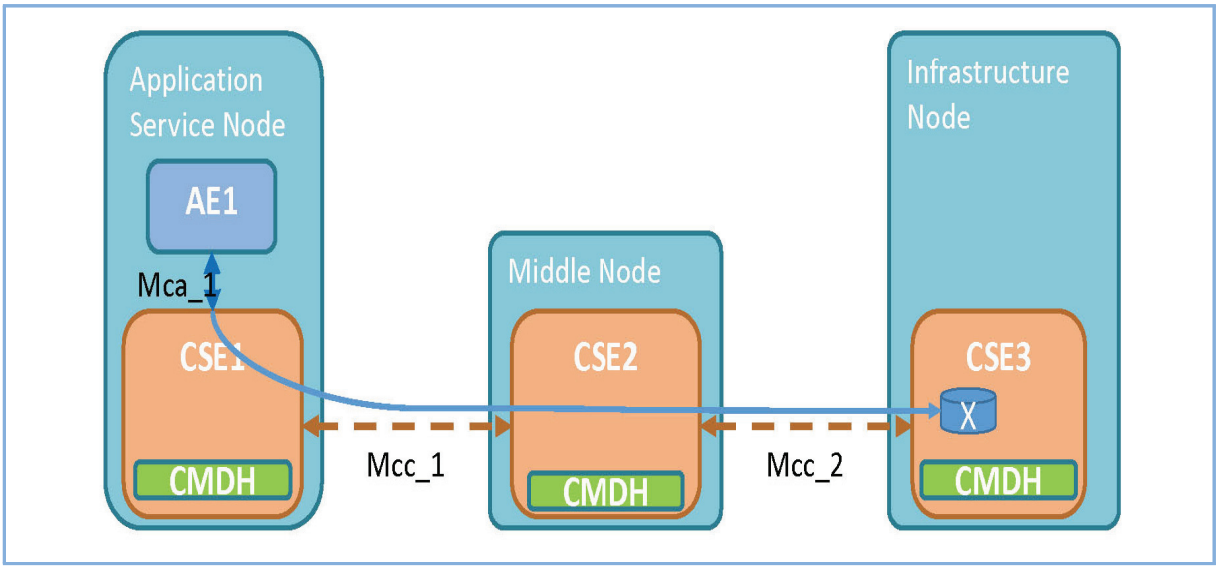



그림 〈3-6〉 CMDH CSF 서비스 제공 예

### 3.4.3. Data Management and Repository (DMR) CSF

- ✓ DMR CSF는 기본적으로 데이터 저장소의 기능을 제공하는 것이다. 또한 데이터의 타입, 시맨틱 정보, 시간, 위치와 관련한 데이터의 분류 및 데이터 포맷의 변경 및 데이터 처리 기능을 제공한다.
- ✓ DMR CSF 에서 관리하는 데이터는 다양한 리소스 타입에 대한 실제 인스턴스들이다. 저장되는 데이터는 다음과 같이 분류할 수 있다. 어플리케이션 데이터, 서비스

구독정보 데이터, 과금 데이터, 위치 정보 데이터, 디바이스 정보 데이터, 시맨틱 정보 데이터, 접근 보안 관련 데이터, 통신 파라미터와 관련한 데이터들이 될 수 있다.

✓ CSE에 위치한 DMR 공통 서비스 기능은 요청 및 전달 데이터의 보관 및 처리를 담당한다. 해당 데이터는 AE 및 CSE 간의 데이터 전달 시에 기본적인 저장 처리 메커니즘을 제공하는 것으로 볼 수 있다.

### 3.4.4. Device Management (DMG) CSF

✓ DMG CSF는 디바이스 관리 기능을 제공하는 공통 서비스 기능이다. 디바이스 관리 대상은 MN (M2M 게이트웨이), ASN (M2M 디바이스) 및 M2M Area Network에 위치한 센서 및 액츄에이터와 같은 자원 제약적인 디바이스를 포함한다. 디바이스 관리에 관한 정보는 리소스 타입으로 표현되며 실제적인 oneM2M에서 디바이스 관리 기능을 담당하는 프로토콜은 디바이스 관리를 위한 프로토콜로서 많이 사용되고 있는 BBF TR-069, OMA-DM, OMA-LWM2M 프로토콜을 활용하도록 정의하고 있다. 그리고 이를 위해서 디바이스 관리 리소스와 실제 프로토콜간의 맵핑이 정의된다.

✓ 디바이스 관리라고 함은 디바이스에 설치된 펌웨어에 대한 관리, 디바이스 하드웨어 자원관리 (배터리, 메모리 등), 디바이스 동작 설정 관리, 진단을 포함한다. DMG CSF가 담당하는 기능은 Device Configuration Function (DCF), Device Diagnostics Monitoring Function (DDMF), Device Firmware Management Function (DFMF), Device Topology Management Function (DTMF)으로 분류할 수 있다.

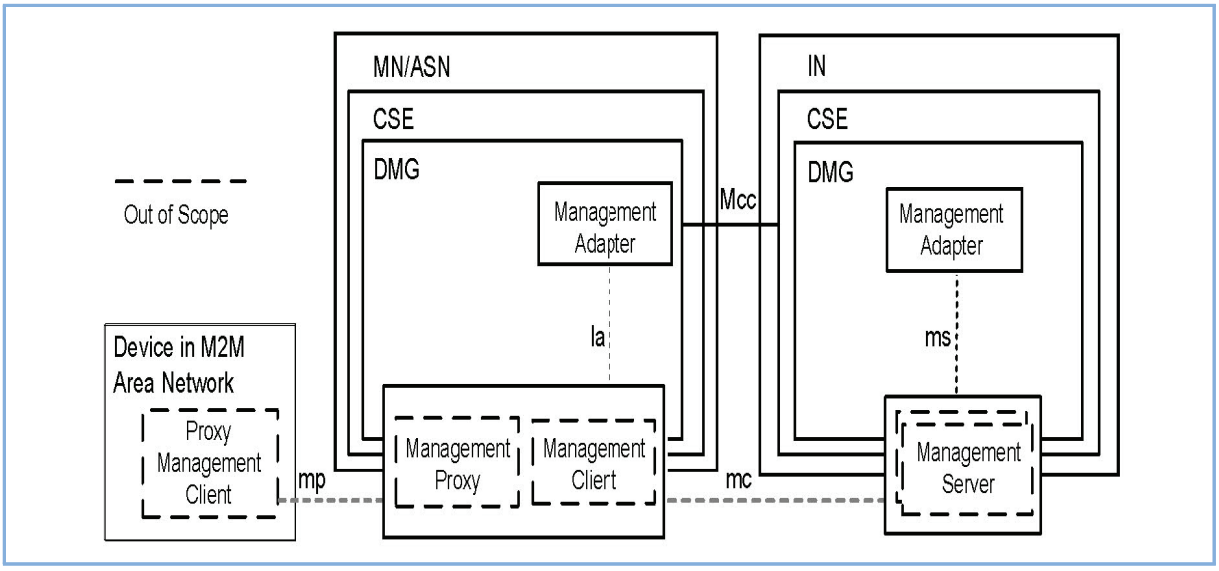



그림 〈3-7〉 디바이스 관리 모델

- ✓ 그림 〈3-7〉에서 보듯이 oneM2M에서 디바이스 관리를 위한 기능은 DMG CSF가 담당하며 실제 프로토콜이 Management Server, Management Adapter, Management Client 및 Proxy 상에서 운용된다. DMG CSF에서 담당하는 디바이스 관리 리소스는 Management Adapter를 통해서 실제 프로토콜인 BBF TR-069, OMA-DM, OMA-LWM2M으로 변환이 되며, Management Server와 Management Client 및 Proxy를 통해서 명령이 내려가서 MN, ASN, M2M Area Network에 위치한 디바이스의 관리를 수행하게 된다.
- ✓ oneM2M은 디바이스 관리를 위한 리소스 타입 및 실제 장치관리 표준과의 맵핑에 관한 표준을 정의하고 있으며 실제 디바이스 관리를 위한 프로토콜 및 해당 디바이스 관리 아키텍처의 구체적인 부분은 Out-of-Scope로 정의하고 있다.

### 3.4.5. Application and Service Layer Mangement (ASM) CSF

- ✓ ASM CSF는 ADN, ASN, MN, IN에 위치한 AE와 CSE 소프트웨어에 대한 관리 기능을 담당한다. 해당 기능은 AE와 CSE의 재설치, 업데이트, 고장탐지, 설정에 관한 기능 제공을 담당한다.
- ✓ ASM CSF의 기능은 Configuration Function (CF)와 Software Management Function (SMF)으로 분류되며 SMF의 기능으로는 CSF, AE에 관한 소프트웨어 패키지 라이프 사이클 관리 (Install, Installed, Updating, Uninstalling and Uninstalled)를 담당한다.

### 3.4.6. Discovery (DIS) CSF

- ✓ DIS CSF는 기본적으로 oneM2M 리소스 및 애트리뷰트에 담긴 서비스 정보에 대한 검색 기능을 제공한다. 해당 기능의 제공은 리소스 및 어트리뷰트에 대한 라벨에 담긴 스트링값의 매칭 검색방법을 통해서 제공되며, 검색에 대한 쿼리를 보내는 Originator 는 Discovery Request 를 전달할 때, Filter Criteria를 통해서 라벨 매칭 및 검색된 정보의 필터링에 대한 룰을 제공한다. Filter Criteria에 담기는 정보는 예를 들면, 정보의 생성시간, 매칭 스트링값, 최대 검색반환갯수, 검색 결과의 순서(오름차순, 내림차순) 등과 같다.
- ✓ Discovery Request 를 수신한 CSE는 자신의 리소스에 대한 검색을 수행 후에 Filter Criteria에 따라서 값을 반환하게 되는데, 검색된 리소스의 URI 리스트를 검색 반환 값으로 전달한다.
- ✓ Discovery Request에 대한 메시지 수신 시에 CSE는 자신을 목적지로 한정하지 않는 Discovery 요청 메시지인 경우에 대해서 자신에게 등록된 AE 또는 CSE로 검색 요청 메시지를 전달하여 추가적인 검색이 가능하도록 지원이 가능하며 이는 정책적으로 설정 될 수 있다.

### 3.4.7. Group Management (GMG) CSF

✓ GMG CSF는 리소스들의 그룹핑을 담당하는 그룹 리소스에 대한 관리 기능을 제공한다. 그리고 그림 〈3-8〉과 같이 Originator는 요청메시지를 그룹 리소스에 한번 보냄으로써, 그룹 리소스에 속한 멤버 리소스들 모두에게 전달이 가능하다. 이를 Bulk 오퍼레이션이라고 한다.

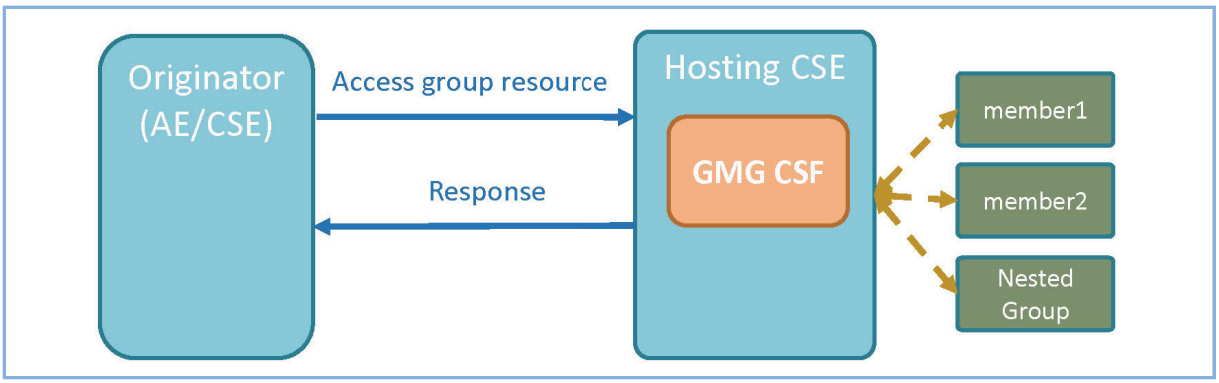



그림 〈3-8〉 GMG CSF

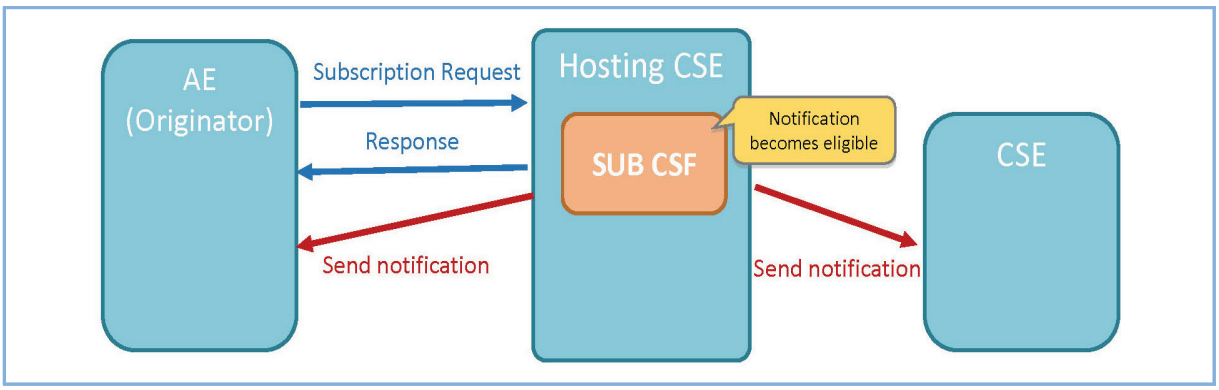



그림 〈3-9〉 SUB CSF

✓ 또한 Bulk 오퍼레이션을 수행한 후 그룹에 속한 멤버 리소스들에 대한 결과 응답을 GMG CSF가 취합하여 Bulk 오퍼레이션을 요청한 Originator에게 응답을 줄 수 있다.

✓ 그룹 리소스에 속한 리소스들은 같은 액세스 권한을 가진 멤버들로만 그룹핑이 되는 것을 가정하고 있으며, 같은 액세스 권한을 가진 멤버들로만 구성되는 것을 GMG CSF가 검증하는 역할을 담당한다.

✓ 그룹 리소스에 멤버로서 그룹리소스를 포함할 수 있기 때문에 연속적인 그룹에 대한 재귀적 처리를 담당하며 마찬가지로 연속적인 그룹에 대한 응답도 취합하여 Originator에게 전달이 가능하다.

### 3.4.8. Subscription and Notification (SUB) CSF

- ✓ SUB CSF는 리소스에 대한 구독정보를 관리하고 리소스 및 애트리뷰트에 대한 업데이트 발생, Child 리소스의 생성, 삭제 등 해당 리소스에 대한 트랙킹 및 관련한 변화 정보를 통지하는 역할을 담당한다. 그림 〈3-9〉에서 보듯이 해당 리소스에 대한 변화가 발생하였을 때, 이에 대한 변화 정보는 해당 리소스에 Subscribe한 구독자에게 통보 된다. 따라서 구독정보를 생성할 때, 통지를 받기 위한 정보 즉 Notification URI 정보에 통보 메시지를 전달받을 리소스 URI 정보가 담긴다.
- ✓ 구독자는 어떠한 이벤트가 발생하였을 때에 통지를 받기를 원하는 지를 명시하게 되어 있고, 이에 대한 정보는 Filter Criteria에 표현된다. 이를 바탕으로 SUB CSF는 리소스 및 애트리뷰트의 업데이트 발생, Child 리소스의 생성 및 삭제가 발생하였을 경우에 구독자가 명시한 Notification URI 정보를 바탕으로 통보하게 되며 구독과 관련하여 구독자는 싱글 리소스에 대한 변화를 구독할 수도 있고 그룹 리소스에 구독함으로서 그룹에 속한 리소스 리스트 각각에 구독을 하지 않고서도 해당 그룹에 속한 리소스 리스트들에 대한 변화를 통보 받을 수 있다.
- ✓ 각 통보 메시지가 전달될 때, 언제, 어떻게 전달될지에 관한 정보는 앞서 살펴 본 CMDH CSF를 통해서 지원된다.

### 3.4.9. Location (LOC) CSF

- ✓ LOC CSF 는 AE의 위치 정보 요청에 대해서 ASN, MN과 관련된 위치정보를 획득하기 위한 방법을 제공하는 CSF이다. 여기에서 위치정보를 요청하는 AE는 LOC CSF가 속한 CSE와 같은 노드에 위치하는 로컬의 경우와 또는 다른 노드에 위치하는 리모트의 경우를 포함하며 이에 대한 처리 및 위치 정보 전달은 LOC CSF가 담당한다.
- ✓ LOC CSF가 제공하는 위치관련 서비스는 세 가지 방법을 지원하고 있으며 위치정보 제공 서버를 활용하는 방법, GPS와 같은 위치수신 하드웨어를 디바이스 자체적으로 활용하는 방법, 앞의 두 가지 방법이 모두 지원되기 어려운 하드웨어 리소스가 제약적이고, 네트워크 리소스가 제약적인 디바이스를 위한 공유 방식의 위치정보 제공 방법을 제공하고 있다.
- ✓ 그림 〈3-10〉은 공유방식의 위치정보 제공방법을 표현하고 있다. 해당 그림에서 MN은 MN과 연결된 ADN 0, 1, 2, 3에 대한 토폴로지 정보를 바탕으로 AE가 자신(ADN0)에 대한 위치정보를 요청했을 때 ADN0의 상대적인 위치를 기반으로 해당 위치정보를 제공하게 된다.

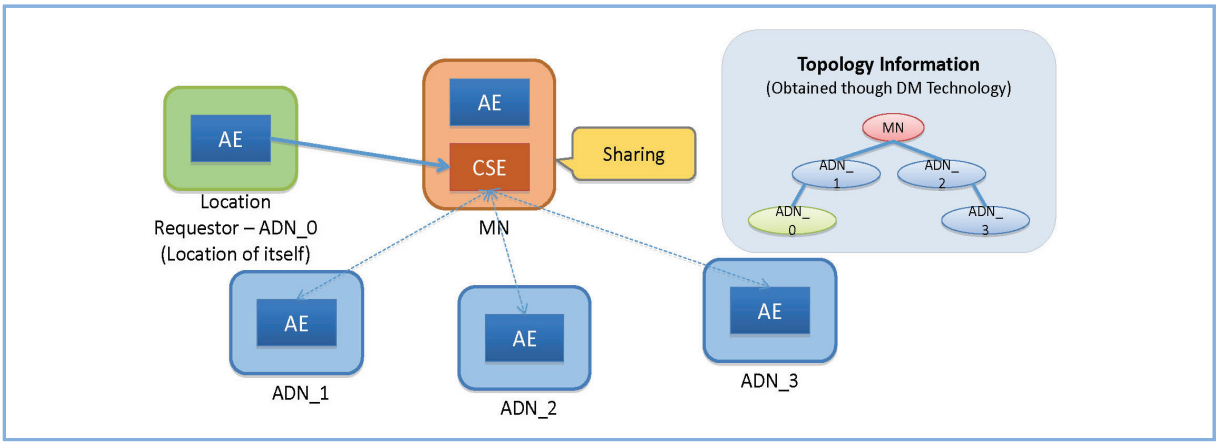



그림 〈3-10〉 LOC CSF의 공유 방식의 위치정보 제공

### 3.4.10. Network Service Exposure, Service Execution and Triggering (NSSE) CSF

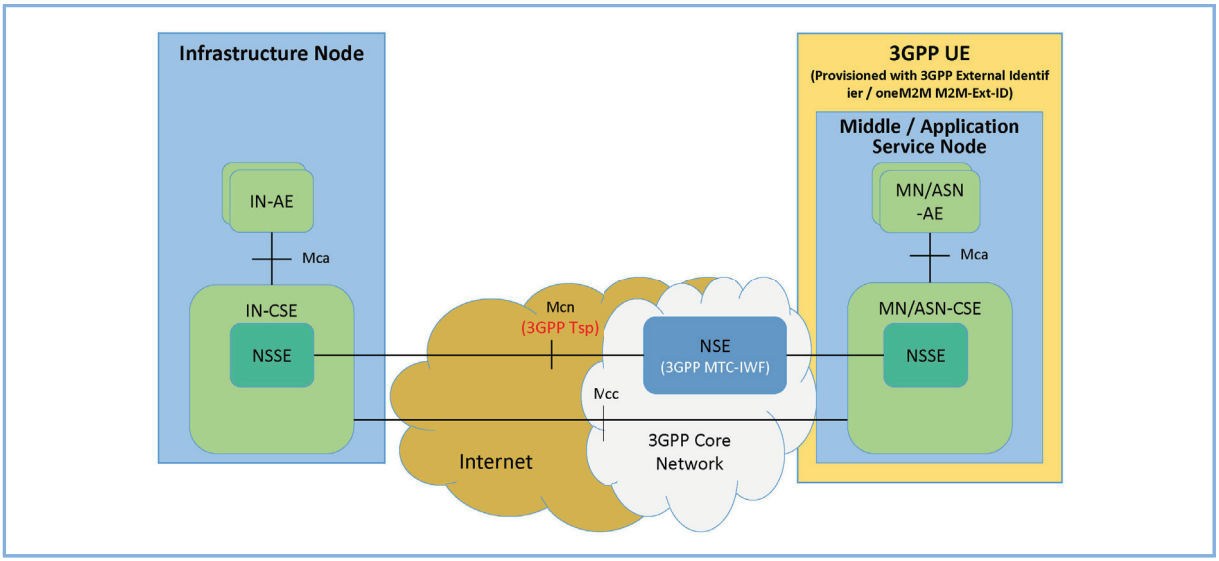



그림 〈3-11〉 Device Triggering

- ✓ NSSE CSF는 기저 네트워크와 관련된 통신을 관리하며 Mcn 참조포인트를 통한 네트워크 액세스 서비스를 제공한다. NSSE를 통하여 기저 네트워크를 사용하는 AE 또는 CSE는 NSSE를 통한 추상화된 정보를 바탕으로 구체적인 네트워크 기술에 대한 직접적인 정보관리가 간소화 될 수 있다.
- ✓ NSSE CSF가 제공하는 기능들을 다음과 같다. 디바이스 Triggering, 네트워크를 통한 위치정보 통지, 디바이스 관리 및 Policy 기반 네트워크 접근 관리 등을 제공받을 수 있다.
- ✓ NSSE CSF는 Mcn 참조포인트를 통한 네트워크 연결 및 세션 관리를 지원하며 CMDH CSF에게 기저 네트워크와 관련한 정보를 제공하여 Policy 기반의 데이터 전달을 지원한다.

✓ 그림 〈3-11〉은 IN-CSE를 통한 필드 도메인에 위치한 ASN/MN 디바이스를 wake-up하기 위한 Triggering 하는 메커니즘을 나타내고 있으며 해당 Triggering 메커니즘은 NSSE와 기저 네트워크를 통해서 제공된다.

### 3.4.11. Security (SEC) CSF

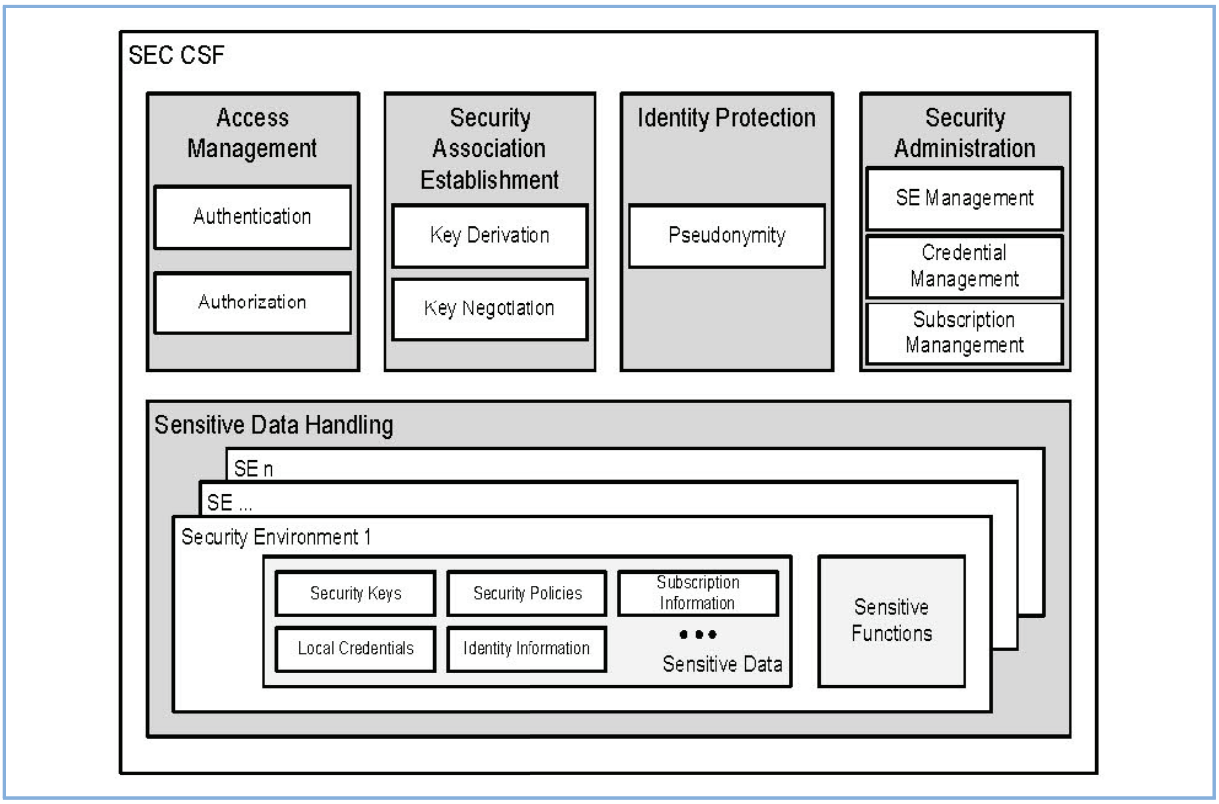



그림 〈3-12〉 SEC CSF

✓ SEC CSF는 oneM2M 공통 서비스 플랫폼의 공통 기능으로서 보안 메커니즘을 제공하는 기능을 담당한다. 제공하는 보안 메커니즘은 그림 〈3-12〉 의 SEC CSF 기능에서 보듯이 Sensitive Data handling, Security Administration, Access control management, Identity Protection, Security Association 설정과 같다.

✓ Sensitive Data Handling: oneM2M 시스템에서 보안키 및 인증서 정보의 노출을 방지하고, 개인 프라이버시 정보 및 보안 키 및 알고리즘과 관련한 정보를 보호하는 기능을 제공한다.

✓ Access Control: 리소스에 대한 CRUD 오퍼레이션을 통한 접근 시에 요청자가 해당 리소스에 접근이 가능한지를 보안적으로 체크하는 메커니즘과 해당 리소스에 접근하기 위한 접근 정보 생성 및 갱신 삭제를 담당한다.

✓ Security Association: M2M 노드 사이에 데이터 전달 시, 기밀성, 무결성, 가용성에 대한 기능을 제공하며 이를 위해서 Key 생성 및 교환, 인증, 보안 알고리즘을 관리한다.

### 3.4.12. Service Charing and Accounting (SCA) CSF

- ✓ SCA CSF는 oneM2M 공통 서비스 플랫폼을 통해서 제공되는 서비스에 대한 과금 체계 및 방법에 관한 기능을 제공한다.
- ✓ 과금 모델과 관련해서는 서비스 프로바이더의 서비스에 가입하는 방식을 통한 가입 기반 과금 방식과, 리소스에 접근 시마다 이벤트가 발생하여 이를 과금에 활용하는 이벤트기반 과금 방식의 두 가지 방식을 SCA CSF는 지원하고 있다.
- ✓ SCA CSF는 세가지 기능으로 구성되는 데, 해당 기능은 과금 관련한 정책 및 설정을 관리하는 Charging management function, 과금 이벤트에 대한 처리 및 정보 기록을 관리하는 Charging triggering function, 오프라인 과금 정보를 트랜잭션 시마다 기록하는 Offline charging function을 포함한다.

## 3.5. oneM2M 메시지 흐름

oneM2M 공통 서비스 플랫폼은 리소스 기반 아키텍쳐 구조를 가지며 Common Service Entity (CSE)가 포함하고 있는 12개의 CSFs를 리소스 타입으로 구현하고 있다. 또한 AE 및 CSE간 리소스에 대한 접근은 기본적으로 Mcc 및 Mca 기반의 참조 포인트 사이에 요청/응답 모델 기반으로 데이터 교환이 이뤄진다. 따라서 해당 요청/응답 메시지를 통해서 oneM2M의 리소스에 대한 접근을 통해서 실제 엔티티간의 oneM2M 서비스가 이뤄진다고 볼 수 있다.

그림 〈3-13〉은 oneM2M의 기본적인 요청/응답 메시지 플로우를 나타낸 것으로서 해당 요청 응답 모델에서 Originator는 AE 또는 CSE가 되고, 요청 메시지를 수신하는 Receiver는 CSE 또는 AE가 될 수 있다. 또한 AE와 CSE간의 메시지 교환은 Mca 참조 포인트를 통해서 그리고 CSE 간의 메시지 교환은 Mcc 참조포인트를 통해서 발생한다. 요청/응답 메시지 안에는 oneM2M 데이터를 전달하기 위해서 이를 위한 메시지 파라미터들이 정의된다.

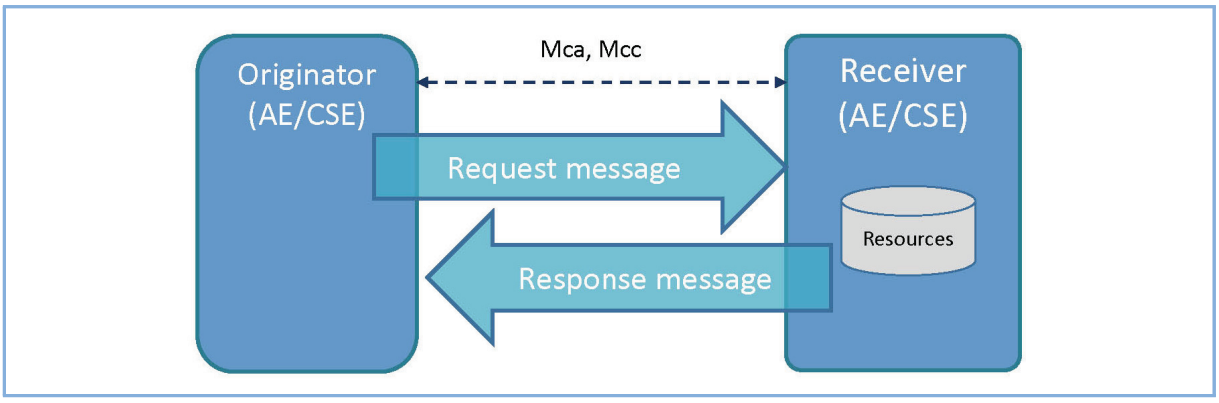



그림 〈3-13〉 기본 요청/응답 메시지 플로우

### 3.5.1. oneM2M 요청 메시지

요청 메시지는 목적지(Target) 리소스에 대한 오퍼레이션을 기준으로 5개의 오퍼레이션을 지원하고 있다. 리소스의 생성을 요청하는 CREATE, 리소스의 대한 값을 가져오는 것을 요청하는 RETRIEVE, 리소스에 대한 값을 갱신하기 위한 UPDATE, 리소스에 대한 삭제를 위한 DELETE, 그리고 리소스의 값을 담아서 통보하는 NOTIFY 가 그것이다.

이러한 오퍼레이션을 약자로서 CRUDN 이라고 칭할 수 있고 해당 오퍼레이션을 기준으로 요청 메시지에는 기본적으로 포함되는 강제 파라미터와 비강제적인 옵션 파라미터가 존재한다. 요청 메시지에 대한 파라미터의 종류 및 강제/옵션에 대한 요약 표는 아래 표 〈3-1〉과 같다.

표 〈3-1〉 요청 메시지 파라미터 요약

| Request message parameter\Operation | Create | Retrieve | Update | Delete | Notify |
|---|---|---|---|---|---|
| ■ Operation (op) - operation to be executed | M | M | M | M | M |
| ■ To (to) - the address of the target resource on the target CSE | M | M | M | M | M |
| ■ From (fr) - the identifier of the message Originator | M | M | M | M | M |
| ■ Request Identifier (ri) - uniquely identifies a Request message | M | M | M | M | M |
| ■ Resource Type (ty) - of resource to be created | M | N/A | N/A | N/A | N/A |
| ■ Name (nm) - of resource to be created | O | N/A | N/A | N/A | N/A |
| ■ Content (cn) - to be transferred | M | O | M | N/A | M |
| ■ Request Expiration Timestamp (rqet) - when the request message expires | O | O | O | O | O |
| ■ Result Expiration Timestamp (rset) -when the result message expires | O | O | O | O | O |
| ■ Response Type (rt) - type of response that shall be sent to the Originator | O | O | O | O | O |
| ■ Result Content (rc) - the expected components of the result | O | O | O | O | N/A |
| ■ Event Category (ec) - indicates how and when the system should deliver the message | O | O | O | O | O |
| ■ Delivery Aggregation (da) - aggregation of requests to the same target CSE is to be used | O | O | O | O | O |
| ■ Filter Criteria (fc) - conditions for filtered retrieve operation | N/A | O | N/A | N/A | N/A |
| ■ Discovery Result Type (Disrestype) - format of information returned for Discovery operation | N/A | O | N/A | N/A | N/A |

요청 메시지에 포함되는 파라미터에 대한 설명은 다음과 같다.

- ✓ Operation: CRUDN 오퍼레이션에 대한 표시
- ✓ To: 목적지(Target) 리소스에 대한 URI를 표시
- ✓ From: Originator에 대한 Identifier를 표시
- ✓ Request Identifier: 요청 메시지에 대한 Identifier Number를 표시하여 응답 메시지에 대한 매칭을 수행하기 위한 용도로 활용함
- ✓ Resource Type: CREATE 오퍼레이션 요청 메시지에 포함되는 파라미터로서 생성하고자 하는 리소스 타입을 표시
- ✓ Name: CREATE 오퍼레이션 요청 메시지에 포함될 수 있는 파라미터로서 생성하고자하는 리소스 이름을 표시
- ✓ Content: 요청 메시지에 포함되는 컨텐츠 데이터를 표시
- ✓ Request Expiration Time Stamp: 요청 메시지에 대한 만료 시간을 표시
- ✓ Result Expiration Time Stamp: 응답 메시지에 대한 만료 시간을 표시
- ✓ Response Type: 세가지 타입을 지원하며 요청 메시지의 응답 처리에 관한 정의를 표시함 (blockingRequest, nonBlockingRequestSynch, nonBlockingRequestAsynch)
- ✓ Result Content: 응답 메시지에 담기는 내용에 대한 요구사항을 표시함 (attributes, attributes+child resources, child-resource, nothing)
- ✓ Event Category: CMDH CSF와 관련있는 파라미터로서 요청메시지에 대한 스케줄링 정보를 표시함 (immediate, bestEffort, latest)
- ✓ Delivery Aggregation: 요청 메시지에 대한 전달 시에 같은 목적지로 향하는 메시지에 대한 aggregation을 수행할 지의 여부를 표시함
- ✓ Filter Criteria: 리소스 검색을 위해서 사용됨, 리소스 검색 시 사용되는 매칭 조건 값을 표시함
- ✓ Discovery Result Type: 리소스 검색 시 반환되는 결과값의 형식에 대한 정의를 표시함 (Hierarchical URI, Non-Hierarchical URI, CSE-ID+Resource-ID)

### 3.5.2. oneM2M 응답 메시지

Originator 에서 보낸 요청 메시지에 대한 처리를 한 후 수신 엔티티는 응답 내용을 포함한 응답 메시지를 작성하여 Originator에게 전달하게 되는데 해당 응답 메시지는 강제적인 파라미터와 조건부 파라미터들을 포함한다. 강제 파라미터는 다음과 같다.

- ✓ Response Code: 응답 메시지는 해당 요청 메시지에 대한 처리 성공, 실패, 단순한 확인의 형태로 응답 코드를 파라미터 값으로 반환하는 것을 표시함 (Successful, Unsuccessful, Acknowledgement)

✓ Request Identifier: 요청 메시지에 포함된 Request Identifier와 동일한 값을 설정해 해당 파라미터로서 응답 메시지에 포함되어 전송한다. 응답 메시지의 Request Identifier는 Originator에서 요청-응답 메시지에 매칭에 사용함

조건부 파라미터는 다음과 같다.

✓ Resource Content: 응답 메시지에 포함되는 데이터로서 성공응답과 실패응답에 따라서 그리고 요청 메시지에 담긴 오퍼레이션의 종류에 따라서 다음의 내용을 담게 된다.

- 성공케이스인 경우:
  - Create: Content는 생성된 리소스의 컨텐츠 및 어드레스 정보
  - Update: Content는 변경된 리소스의 컨텐츠 정보
  - Delete: Content 는 옵션으로 삭제된 리소스를 가질 수 있음
  - Retrieve: Content는 Retrieve 요청한 리소스의 정보 및 Discovery 요청인 경우 결과로서 발견된 리소스 URI 리스트의 정보가 담긴다.
- 실패케이스인 경우:
  - Content는 실패원인 및 에러정보가 담긴다.

### 3.5.3. CREATE 요청과 해당 응답 메시지 예

- Request
  - ***op***: C (Create)
  - ***to***: address of the target resource
  - ***fr***: Originator ID
  - ***ri***: Request Identifier
  - ***ty***: resource type to be created
  - ***cn***: mandatory attributes for the resource
  - ***nm***: resource name
- Response
  - ***rs***: result (status codes, etc.)
  - ***ri***: Request Identifier
  - ***cn***: address and content of created resource (if successful)

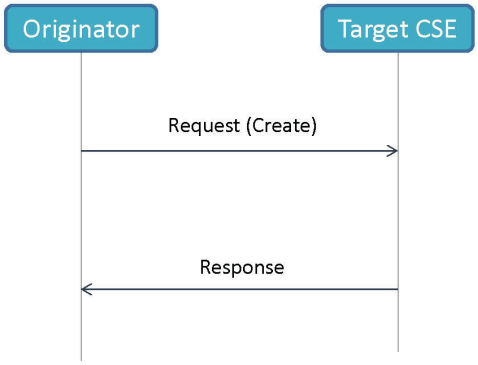



그림 〈3-14〉 CREATE 요청/응답 예시

그림 〈3-14〉의 예와 같이 Originator에서 요청한 메시지에 대한 Receiver로부터 응답 메시지에 포함되는 파라미터들은 다음과 같을 수 있다. 요청 메시지에는 오퍼레이션 파라미터로서 CREATE를 포함하며 To 파라미터로서 목적지 리소스의 주소 정보를 포함한다. From 정보로서 요청한 엔티티의 ID를 포함하며 Request Identifier 정보로서

임의의 값을 설정하여 해당 파라미터에 담게 된다. 또한 CREATE 오퍼레이션의 경우는 생성하고자 하는 리소스에 대한 타입을 명시하는 Resource Type 파라미터를 포함하며 Content 파라미터로서 생성하고자 하는 리소스의 정보를 포함한다. 또한 옵션으로 생성하고자 하는 리소스에 대한 이름을 Name 파라미터에 설정할 수 있다.

해당 요청 메시지에 대한 응답 메시지에는 Response Code에 성공, 실패의 정보를 포함하며 요청 메시지와 동일한 Request Identifier를 포함하여 Originator 에서 응답 메시지가 어느 요청 메시지에 대한 응답인지를 확인할 수 있게 된다. 또한 CREATE 요청에 대한 성공적인 응답 메시지인 경우에는 생성된 리소스의 정보를 포함할 수 있으며 만약 실패 응답 메시지인 경우는 실패이유에 대한 정보를 Content 파라미터에 포함하여 전달할 수 있다.

### 3.5.4. Blocking및 Non-Blocking 메시지 흐름

oneM2M 요청 파라미터에는 Response Type 파라미터를 포함할 수 있고 해당 정보는 Blocking Request, Non-Blocking Synchronous Request, Non-Blocking Asynchronous Request 로 설정할 수 있다.

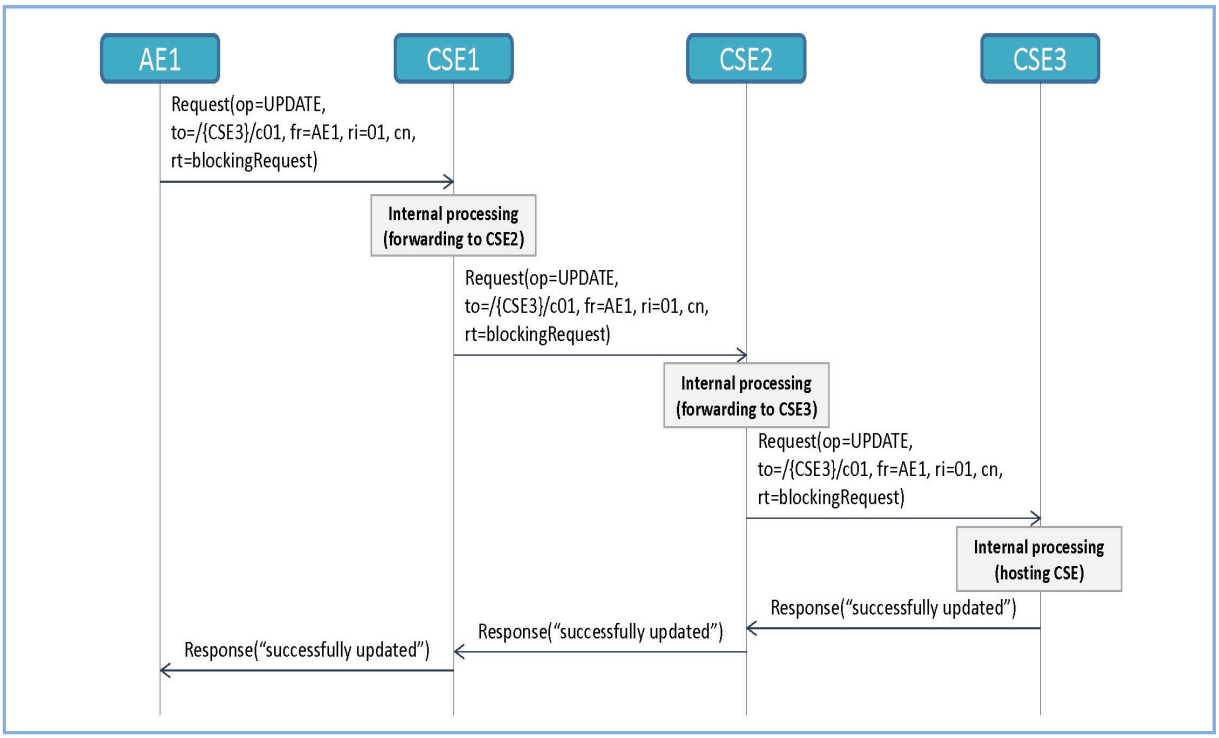



그림 〈3-15〉 Blocking Request 에 대한 메시지 흐름

Blocking Request 와 관련해서는 Originator의 요청 메시지에 대한 처리가 Receiver 로부터 응답이 오기 전까지 Originator에서 해당 요청 메시지에 대한 세션을 유지하며 기다리는 것으로서 그림 〈3-15〉의 메시지 흐름을 갖는다. 해당 예는 AE1이 Originator 가

되고 목적지가 CSE3가 되는 상황에서 CSE3 내에 있는 리소스에 대한 업데이트를 요청하고 있고 Response Type 파라미터로서 blockingRequest를 포함하고 있음을 볼 수 있다.

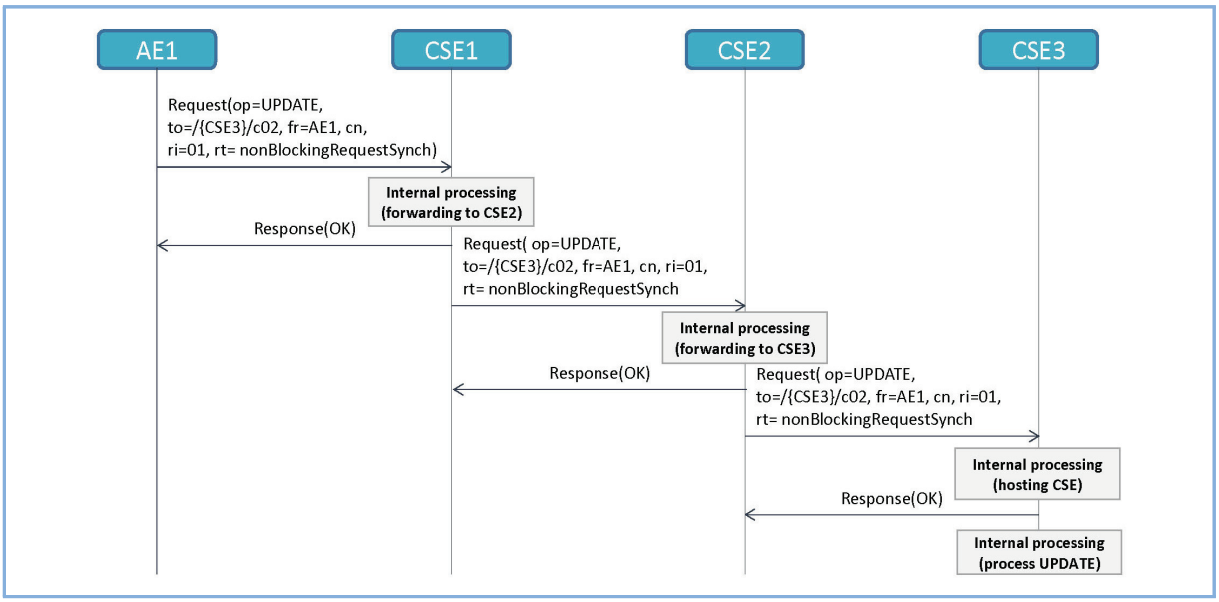



그림 〈3-16〉 Non-Blocking Request 에 대한 메시지 흐름

Non-Blocking Request와 관련해서는 Originator의 요청 메시지에 대한 목적지에서 해당 요청을 처리하여 응답 메시지를 전달받기 까지 계속 세션을 유지하는 경우가 아니라 요청 메시지에 대한 확인 응답 메시지만 받고 난 후 바로 세션이 끊어진다. 이후에 요청에 대한 처리가 완료되면 목적지로부터 Originator로 처리 완료된 메시지를 받거나 (Non-Blocking Asynchronous) 또는 요청 메시지에 대한 확인 응답 메시지에 Reference Resource 정보를 받아서 Originator가 처리가 완료되었는지를 Reference Resource에 접근하여 확인하는 방식 (Non-Blocking Synchronous)을 지원하고 있다. 그림 〈3-16〉는 Non-Blocking Request에 대한 메시지 흐름을 표시하고 있으며 Blocking Request 와 달리 요청 메시지에는 Response Type으로서 nonBlockingRequestSynch 정보를 포함하고 있으며 요청 메시지에 대한 확인응답을 바로 받는 방식을 볼 수 있다.

## 3.6. 장치 등록하기

IoT 장치를 구매하고 가장 우선적으로 하는 일은 장치의 전원을 켜는 일이다. 장치의 전원을 켜면 장치는 해당 장치가 등록되어야 하는 게이트웨이 또는 서버를 찾아 등록을 하게 된다. oneM2M에서는 정의하는 장치는 ASN, MN, IN이 존재하는데, 각 논리적인 장치에 포함한 AE 및 CSE에 따라 서로 다른 자원을 등록하므로, 장치 등록이 수행된다.

AE가 CSE에 등록될 때 〈AE〉 자원을 사용하여 AE에 대한 정보를 CSE에 저장하도록 한다. CSE에 등록된 〈AE〉 자원은 외부에서 해당 〈AE〉에 대해서 이해할 수 있도록 AE의 정보를 포함하는데 이는 아래 표에 정리하였다.

표 〈3-2〉 〈AE〉 자원의 속성 정보

| 속성 | 설명 |
|---|---|
| AE-ID | 시스템에서 유일하게 식별 가능한 AE-ID |
| name | 사람이 이해할 수 있는 AE 이름 |
| ontologyRef | 해당 AE가 이해할 수 있는 Ontology 정보. Application 간에 Data를 공유/재사용 할 때 각자 지원하는 Ontology 정보를 공유하기 위함 |
| pointOfAccess | Registrar CSE가 AE에 Request를 Re-targeting하기 위한 네트워크 주소 (E.g., IP, FQDN) |
| nodeLink | 해당 CSE의 Node 정보를 가지고 있는 〈node〉 에 대한 Link |

CSE가 다른 CSE에 상호 등록되는 경우는 〈remoteCSE〉 자원을 사용한다. 〈AE〉 자원이 등록되는 이유와 동일하게 CSE가 상호 등록될 때는 CSE 간 서로의 정보를 〈remoteCSE〉에 저장한다. 위에서 상호 등록이라는 표현을 사용했는데, 이는 아래 그림 〈3-17〉과 같이 Originator CSE가 Receiver CSE에 등록을 요청할 때, Originator CSE의 정보를 포함하여 Receiver CSE에 〈remoteCSE〉 자원을 생성한다. 이와 동일하게 등록 요청에 대한 응답에 Receiver CSE가 자신의 정보를 포함하고 이를 바탕으로 Originator CSE에 〈remoteCSE〉 자원을 생성함으로써 상호 간 CSE에 대한 정보를 공유하게 된다.

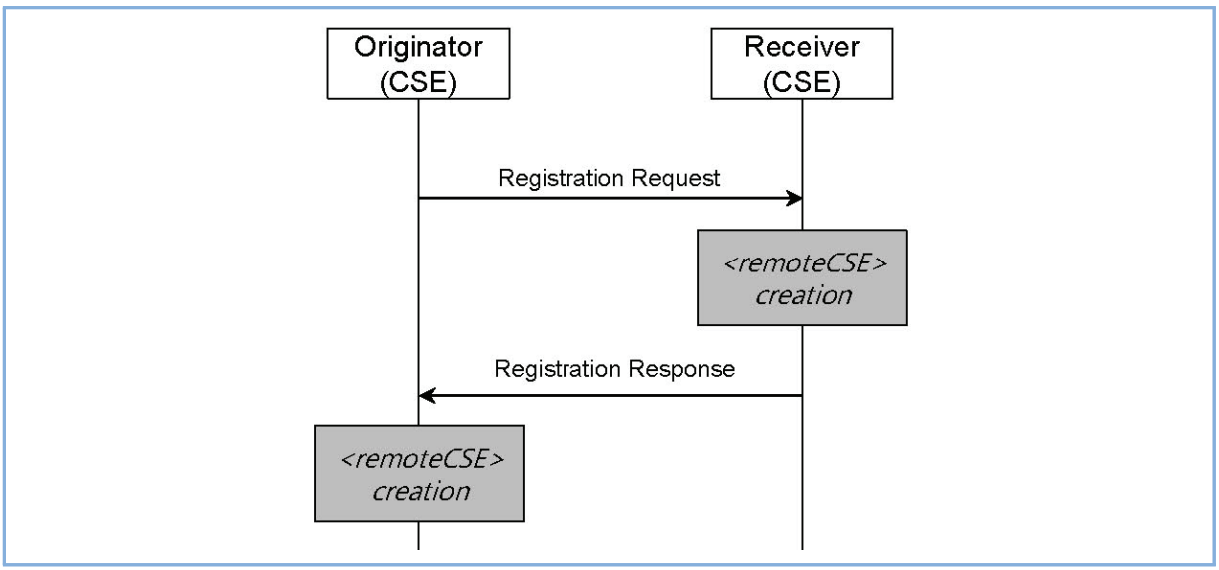



그림 〈3-17〉 CSE 간 상호 등록하는 절차

아래 표 〈3-3〉은 〈remoteCSE〉 자원에 대한 주요 정보를나타낸다.

표 〈3-3〉 〈remoteCSE〉 자원의 속성 정보

| 속성 | 설명 |
|---|---|
| cseType | CSE Type (IN-CSE, MN-CSE, ASN-CSE) 정보<br>IN-CSE는 필수로 해당 정보 명시 |
| CSE-ID | 시스템에서 유일하게 식별 가능한 CSE-ID |
| M2M-Ext-ID | Device Triggering을 위한 Target Device ID (e.g., MSISDN) |
| Trigger-Recepient-ID | 3GPP Device Triggering의 Application Port ID에 해당 |
| pointOfAccess | 해당 CSE에 등록한 AE/CSE가 연결을 맺기 위한 네트워크 주소 (E.g., IP, FQDN) |
| nodeLink | 해당 CSE의 Node 정보를 가지고 있는 〈node〉 에 대한 Link |
| requestReachability | NAT 사용 등 환경에서 다른 엔티티로부터 직접(Asynchronous) Request를 수신할 수 없는 경우 설정. 〈pollingChannel〉 와 연계 |

## 3.7. 데이터를 저장하고 공유하기

AE 및 CSE간 등록 절차를 통해서 특정 응용 서비스 (예, 스마트 홈)를 위한 oneM2M 기반 네트워크 구성이 완료된다. 네트워크를 구성하는 장치들은 일반적으로 장치의 상태에 대한 데이터를 저장하고 공유하므로 응용 서비스를 진행한다. 예를 들어 스마트 홈 응용 서비스에서 가정 내 세탁기는 자신의 상태 정보(예, 탈수 중, 1시간 후 예약 세탁 진행)를 저장하고 공유하여 자신이 상태를 알려주거나 외부 장치가 자신의 상태 변경하여 제어할 수 있도록 구성이 된다. 이는 자원의 상태를 전달하면서 동작 제어를 유도하는 REST 시스템의 동작의 특징이라고 할 수 있다.

oneM2M에서는 응용 서비스 동작 간 발생하는 데이터를 저장하고 공유하는 데 〈container〉와 〈contentInstance〉 자원을 사용한다. 두 개의 자원은 우리가 일반적으로 사용하는 운영체제의 파일시스템에서 사용하는 디렉토리와 파일의 개념으로 이해하면 쉽다. 실제 데이터가 저장되는 파일과 한 개 또는 복수 개의 파일이 저장되고 계층적으로 구성이 되는 디렉토리를 각각 〈contentInstance〉와 〈container〉 자원으로 매칭할 수 있다. 따라서 디렉토리가 안에 포함한 파일들의 대표 정보 (예, 전체 크기, 파일의 수, 최근 수정된 시간 등)를 나타내는 것처럼 〈container〉는 안에 있는 데이터의 대표 정보를 포함하고, 파일이 각 파일의 정보 및 실제 데이터를 포함하는 객체인 것처럼 〈contentInstance〉 자원은 실제 데이터를 포함하고 각 데이터의 정보를 포함한다. 아래 표 〈3-4〉과 표 〈3-5〉는 각 〈contentInstance〉 자원과 〈container〉 자원의 자식 자원 및 속성 정보를 나타낸다.

표 〈3-4〉 〈contentInstance〉 자원의 속성 정보

| 속성 | 설명 |
|---|---|
| typeOfContent | content에 저장된 데이터의 유형 정보 (예, 동영상, text 등) |
| contentSize | content의 Byte 크기 |
| content | contentInstance에 저장되는 실제 데이터 |

표 〈3-5〉 〈container〉 자원의 자식 자원 및 속성 정보

| 자식 자원 | 설명 |
|---|---|
| 〈container〉 | 하위 계층 데이터 저장소 |
| **속성** | **설명** |
| creator | container를 생성한 AE 또는 CSE의 ID |
| maxNrOfInstances | 해당 container에 저장될 수 있는 최대 〈contentInstance〉 자원의 숫자 |
| maxByteSize | 해당 container의 최대 허용 Byte 크기 |
| maxInstanceAge | 해당 container 내 〈contentInstance〉 자원이 저장될 수 있는 최대 시간 (초 단위) |

그림 〈3-18〉은 위에서 설명한 〈container〉 자원과 〈contentInstance〉 자원의 활용을 나타내고 있다. 스마트 홈 응용 서비스에서 방에 온도와 습도 정보를 나타내는 구조에서의 해당 자원을 활용하는 것을 도식화 하였다.

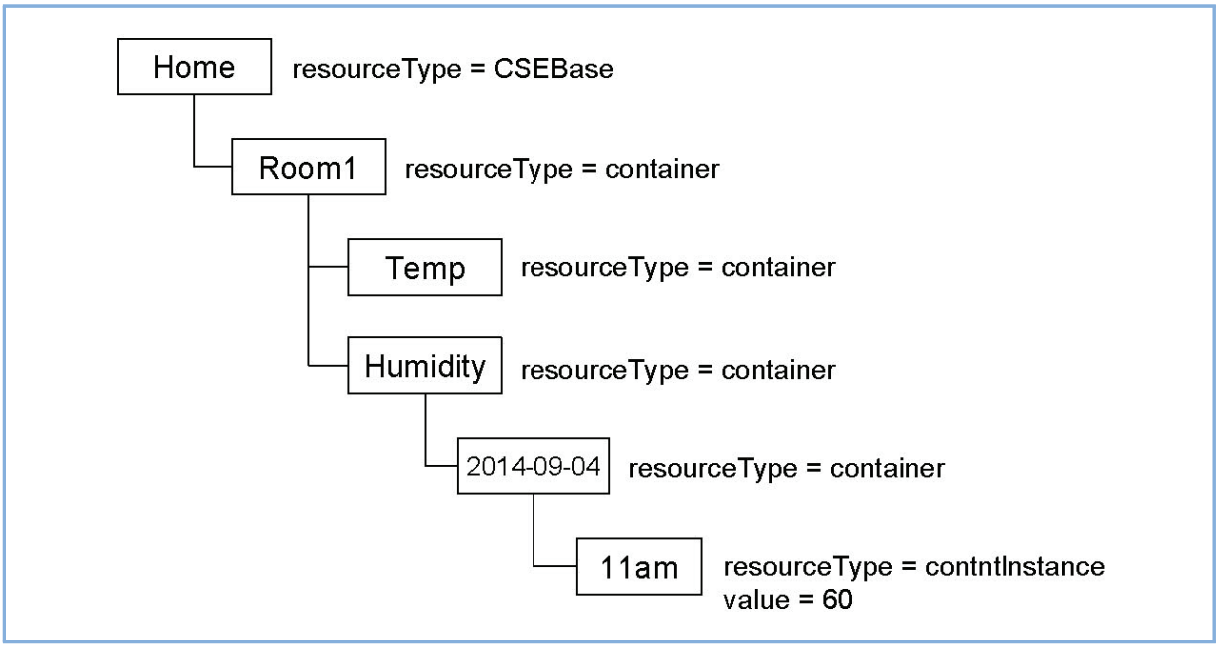



그림 〈3-18〉 CSE 간 상호 등록하는 절차

## 3.8. 자원 접근 제어하기

스마트 홈 응용 서비스를 예로 들어보자. 집에 있는 세탁기 상태 정보를 〈container〉와 〈contentInstance〉 자원에 저장하여 외부에서 해당 정보를 접근하도록 한다고 가정한다. 이에 따른 문제는 해당 상태 정보를 누구나 다접근하고 변경할 수 있도록 할 것이냐는 것이다. 답은 당연히 아니다가 되어야 할 것이다.

oneM2M에서는 해당 접근을 권한을 표현하는 〈accessControlPolicy〉 자원을 정의하여 CSE가 해당 자원에 담겨 있는 정보를 토대로 해당 자원에 접근 결과를 응답하도록 한다. 〈accessControlPolicy〉 자원에 포함된 권한에는 누가 (Request Originator), 무엇에 대한 (Target Resource) 어떠한 요청(Operation)을 할 수 있는지를 작성한다.

표 〈3-6〉은 〈accessControlPolicy〉 속성 정보를 나타낸다. 〈accessControlPolicy〉는 언급한 바와 같이 특정 자원에 대한 접근 권한을 나타내기 때문에 자원마다 기술되어야 하는 것이 맞다. 하지만, 실제적으로 서비스를 진행할 때는 자원마다 접근 권한이 동일한 경우가 다수 있으며, 동일한 접근 권한이 중복되는 것을 피하기 위해 자원 마다 접근 권한을 두지 않고, 접근 권한을 〈accessControlPolicy〉 자원에 기술한 후에 해당 접근 권한을 사용하는 자원은 해당 〈accessControlPolicy〉의 링크 정보를 포함한다. 해당 정보를 위해서 모든 자원은 공통 속성으로 'accessControlPolicyIDs'를 포함한다. 해당 속성에는 자원에 접근 권한을 포함한 〈accessControlPolicy〉 자원의 URI가 포함된다.

**표 〈3-6〉 accessControlPolicy〉 자원의 속성 정보**

| 속성 | 설명 |
|---|---|
| privileges | 해당 〈accessControlPolicy〉를 Link하고 있는 (i.e., accessControlPolicyIDs attribute) 자원에 대한 엑세스 권한 정보 |
| selfPrivileges | 해당 〈accessControlPolicy〉 자원 자체에 대한 엑세스 권한 정보 |

그림 〈3-19〉은 자원의 접근 권한을 〈accessControlPolicy〉 자원 기반으로 제어하는 방식을 나타낸다. 온도 센서 응용(AE1)이 외부 장치의 온도 값을 가져오려고 한다면 우선 온도 값을 요청할 텐데 해당 요청에는 기본적으로 요청의 발신자 (Originator), 목적지 (Target)과 필요한 동작(Operation)을 명시한다. 해당 요청을 수신한 외부 장치의 온도 값을 저장하고 있는 CSE1는 목적지 자원의 accessControlPolicyIDs 속성에 명시된 〈accessControlPolicy〉 자원의 URI을 통해 해당 자원에 대한 접근 권한 정보를 확인하여 해당 요청에 대한 접근을 허가하거나 거절할 수 있다. 그림에 명시된 예에 따르면, CSE1에 저장된 〈container1〉 및 〈subscription1〉 자원에 접근 권한 정보는 〈accessControlPolicy〉 타입 자원인 〈ACP1〉에 명시되어 있으며, 명시된 내용은 AE1은 연결된 자원에 RETRIEVE 동작 요청을 할 수 있다는 것이다.

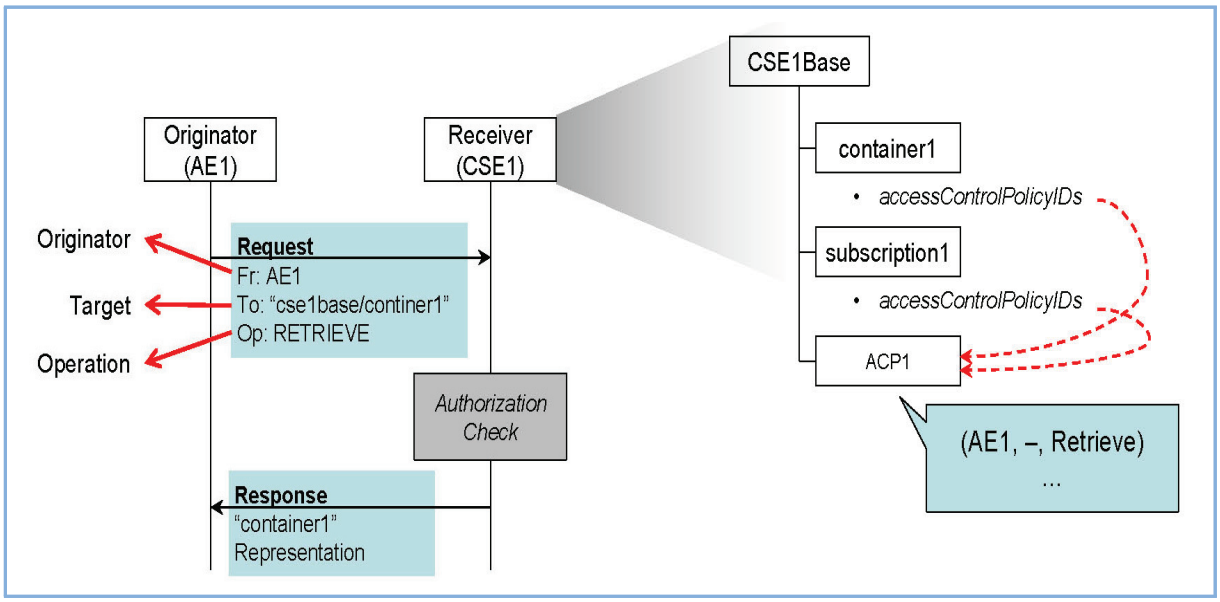



그림 〈3-19〉 자원에 대한 접근 권한 제공 방식

추가적으로 자원에 대한 접근 권한은 조건 기반으로도 기술 될 수 있는데, 명시된 시간, 특정 위치와 같은 조건을 명시하면 조건에 따라 접근 권한이 적용될 수 있다.

## 3.9. 장치 위치 획득하기

IoT 응용 서비스를 좀 더 매력적으로 만들기 위해서는 위치 기반의 응용 부분을 포함하는 것도 좋은 방법이다. oneM2M 서비스 플랫폼은 그런 측면에서 장치의 위치를 제공하고, 이를 관리하도록 하는 기능을 제공하여 서비스의 질을 향상할 수 있도록 한다.

언급한 바와 같이 oneM2M에서 장치 위치를 획득하고 관리하는 방법은 2가지 자원을 기반으로 수행된다. 우선 〈locationPolicy〉 자원은 장치의 위치를 획득하는 방법을 기술하는 자원으로 AE가 특정 장치의 위치를 획득하고자 할 때, 위치 획득 하는 방법이나 장치의 식별자를 포함한다. 추가적으로 필요한 자원은 〈container〉 자원인데 데이터를 보관하는 자원인 만큼 획득된 위치를 저장하는데 사용이 된다. 표 〈3-7〉은 〈locationPolicy〉 자원의 속성 정보를 나타낸다.

표 〈3-7〉 〈locationPolicy〉 자원의 속성 정보

| 속성 | 설명 |
|---|---|
| locationSource | 위치정보를 획득하는 방법 설정<br>1. 위치 서버로 부터 획득 (E.g., OMA RESTful NetAPI for Terminal Location, OMA MLP)<br>2. GPS 모듈로 부터 획득<br>3. 타 단말로 부터 위치 정보 공유 방식 |

| | |
|---|---|
| locationUpdatePeriod | 위치 정보 업데이트 주기 정보 |
| locationTargetID | 위치 서버로 단말 위치 획득하는 경우 사용하는 타겟 단말의 식별자 정보 |
| locationServer | 위치 서버의 주소 정보 |
| locationContainerID | 위치 정보가 저장되는 〈container〉 자원의 URI |
| locationContainerName | 위치 정보가 저장되는 〈container〉 자원의 이름 |
| locationStatus | 현재 Location 정보 요청에 대한 상태 정보 |

그림 〈3-20〉은 위치 정보 획득 방식을 기술한 〈locationPolicy〉 자원과 해당 방식을 기반으로 획득한 위치 정보를 저장하는 〈container〉 자원 간 연결을 도식화하였다. 서로 다른 자원의 URI를 속성에 저장하여 해당 자원 간 논리적인 연결을 가능하게 한다.

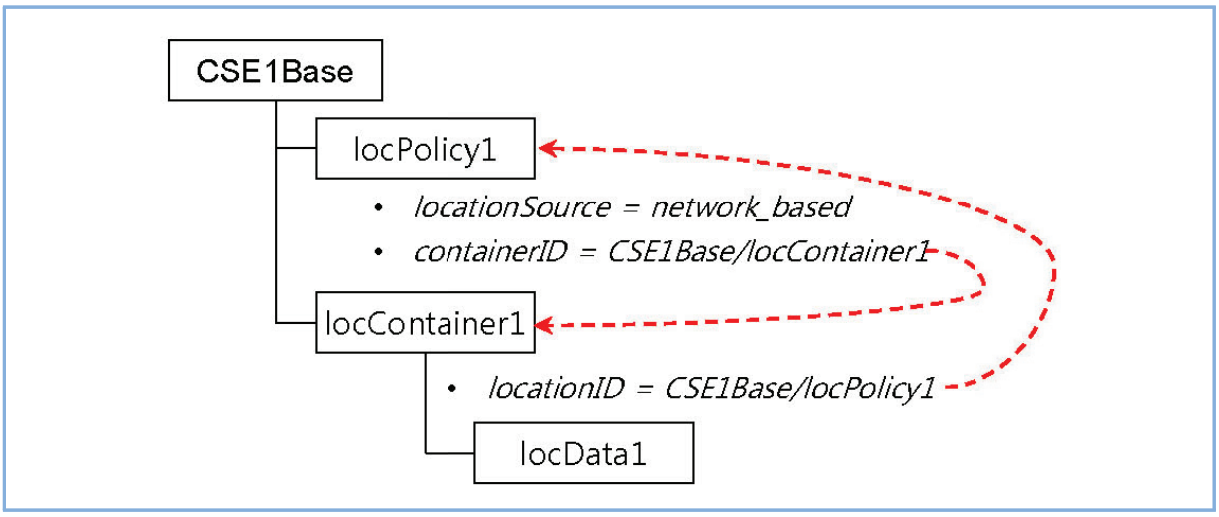



그림 〈3-20〉 〈locationPolicy〉 자원과 〈container〉 자원의 연결

## 3.10. 데이터/서비스 공지하기

공지(Announce) 기능은 생성된 자원의 바로가기를 만드는 것과 비슷하다. CSE는 각기 서비스 플랫폼으로써 데이터/서비스를 자원으로 저장하며 어플리케이션은 이를 이용하여 응용 서비스를 제공한다. 서로 다른 어플리케이션 간에 자원을 공유하기 위해서는 먼저 다른 어프리케이션이 생성한 자원을 검색할 수 있어야 한다. (다음 절에 자세히 기술되어 있다.) 해당 자원이 어떤 CSE에 저장되어 있는지 사전에 알고 있으면 검색이 용이하다. 공지 기능은 특정 자원이 어디에 있는지 다른 CSE에 바로가기를 생성하여 해당 CSE에서 이 원본 자원을 검색할 수 있는 기능을 제공한다.

그림 〈3-21〉은 AE01이 데이터 저장소(〈container〉 자원)인 자원인 cont1을 CSE1에 생성하여 CSE2에 이를 공지하고, AE02가 CSE02를 통해 cont1이 CSE1에 저장되어 있음을 알아내는 과정을 나타낸다.

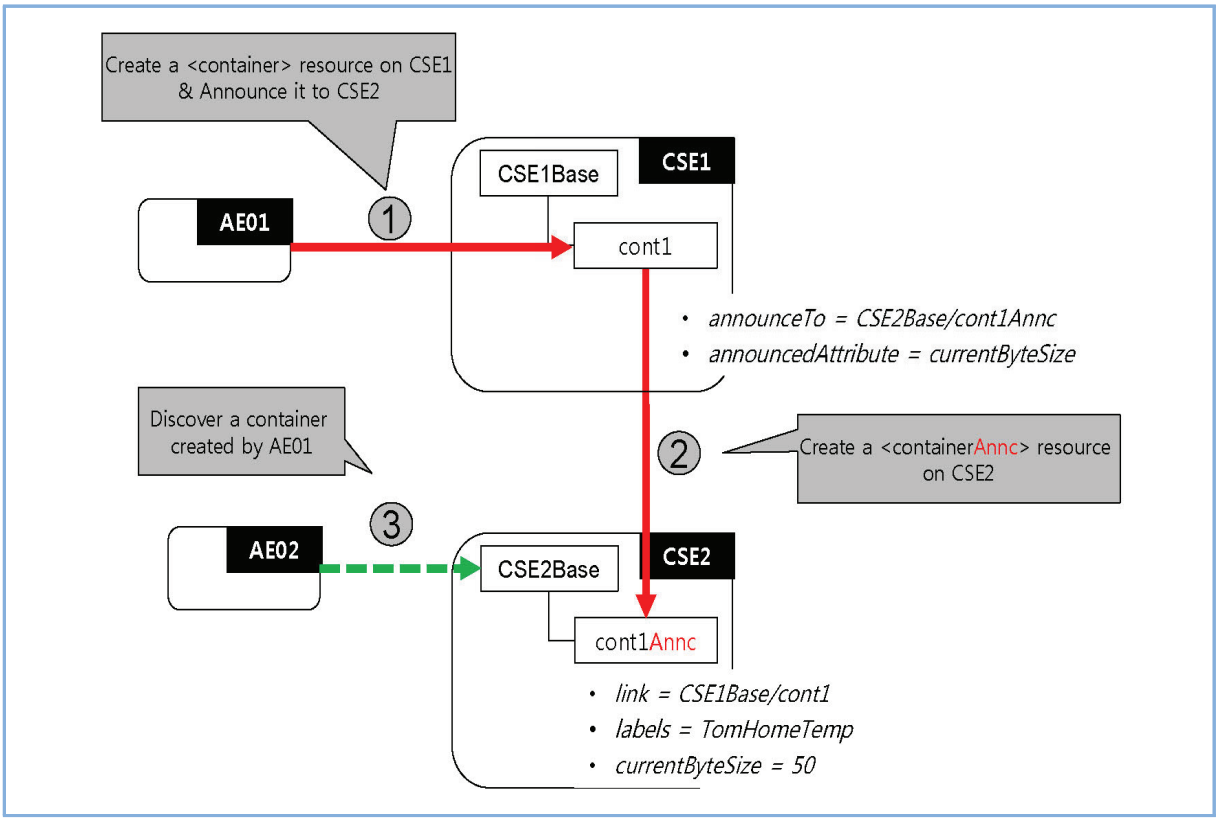



그림 〈3-21〉 공지 생성 및 공지 발견 절차

과정 1에서 AE01은 CSE1에 cont1 자원을 생성하고 이 자원을 CSE2에 공지하도록 요청한다. 과정2에서 CSE1은 AE01의 요청에 따라 CSE2에 저장소 공지 자원(〈containerAnnc〉 자원 타입)인 cont1Annc를 생성한다. 이후 과정 3에서 AE02가 cont1Annc의 라벨 정보를 통해 cont1Annc 자원을 검색하고 이 자원을 획득하면 CSE1Base/cont1이라는 cont1의 주소 정보를 가질 수 있다.

## 3.11. 필요한 데이터/서비스 검색하기

발견(Discovery) 기능은 특정 CSE가 가진 자원에 대한 검색 기능이다. 즉, 발견의 대상이 엔티티가 아닌 자원이라는 특징을 가진다. oneM2M은 자원 기반의 아키텍처를 취하여 서비스를 자원으로 형상화 하므로 특정 서비스를 발견하는 동작은 해당 자원을 검색하는 동작으로 구성된다. 요청자는 특정 자원을 검색하기 위한 조건을 명시하여 요청 메시지를 전송하고, 수신자는 조건에 맞는 자원을 자신이 가진 자원 중에 검사하여 해당 자원의 목록을 반환한다.

다른 기능과는 다르게 발견 기능을 위한 별도의 자원 타입은 존재하지 않는다. 즉 요청자는 자원 발견을 위한 요청을 "발견" 자원에 전송하지 않고 임의의 자원에 전송할 수 있다. 이 자원의 위치는 자원 검색의 시작점이라고 할 수 있다. 앞서 살펴본 것과 같이

CSE는 자원을 트리(tree)형식으로 저장하고 있다. 따라서 여기서 말하는 검색 시작점은 자원 트리 상의 특정 자원을 의미하고 자원 검색의 범위는 시작점이 가리키는 자원 및 해당 자원에 속한 모든 하위 자원을 포함한다. 아래 도면은 요청자가 Home/Room1/Humidity에 대한 발견 요청을 수행하는 과정을 나타낸다. 이 때 수신자는 Humidity 및 x3, a1 자원에 대한 검색을 수행한다.

아래 예시에서 확인할 수 있듯이 발견 기능은 획득동작 (retrieve operation)을 사용한다. 발견 기능에는 특정 조건을 만족하는 자원을 검색할 수 있도록 필터링 기능을 제공하는데 이는 요청 파라미터 중 fc(filter criteria)로 정의되어 있다. fc는 다양한 필터 조건으로 구성되어 있다. 예시처럼 특정 자원 타입을 검색할 수도 있고 특정 시간 이후에 생성된 자원만을 검색할 수도 있다. 그리고 검색 결과를 제한할 수도 있다. 필터 조건은 여러 개를 동시에 사용할 수 있다. 동일한 조건을 여러 개 사용하는 경우는 OR 논리 연산을 적용하고 다른 조건을 여러 개 사용하는 경우는 아래 예시와 같이 AND 논리 연산을 적용하여 결과를 반환한다. 예를 들어 "resourceType=AE&resourceType=container"인 경우 AE 자원 타입 또는 container 자원 타입의 자원을 검색한다.

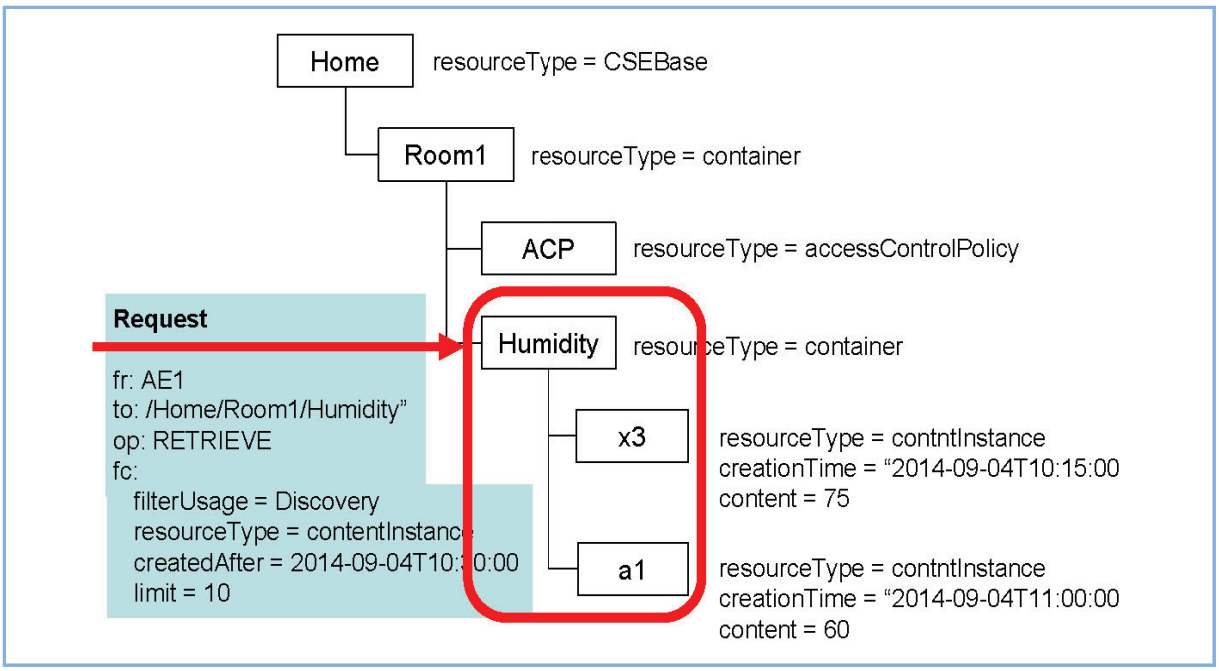



그림 〈3-22〉 발견 기능 예시

일반적인 자원 획득 기능과 발견 기능 모두 획득 동작을 사용하므로 이를 구분하기 위하여 발견 기능 사용 시에는 filter criteria의 filterUsage를 Discovery로 설정한다.

## 3.12. 관심 있는 데이터 변경 시 통지 받기

구독(Subscription)은 어플리케이션 간 데이터 교환에 유용하게 쓰이는 기능이다. CSE에는 많은 데이터/서비스가 자원으로 저장되어 있고 어플리케이션은 이러한 자원을 이용할 수 있다. 어플리케이션이 특정 자원에 관심이 있어 해당 자원의 변경 정보를 통지 받고자 할 경우 CSE가 제공하는 구독 기능을 이용할 수 있다. 이때 구독할 수 있는 정보는 구독 대상 자원에 대한 다른 엔티티의 접근 내역, 자원의 속성 변경 및 자녀 자원의 변경을 포함한다. 발견 기능과 비슷하게 특정 조건을 만족하는 변경 사항만 통지 받기 위해 통지 이벤트 조건(notificationEventCriteria)를 설정할 수 있다.

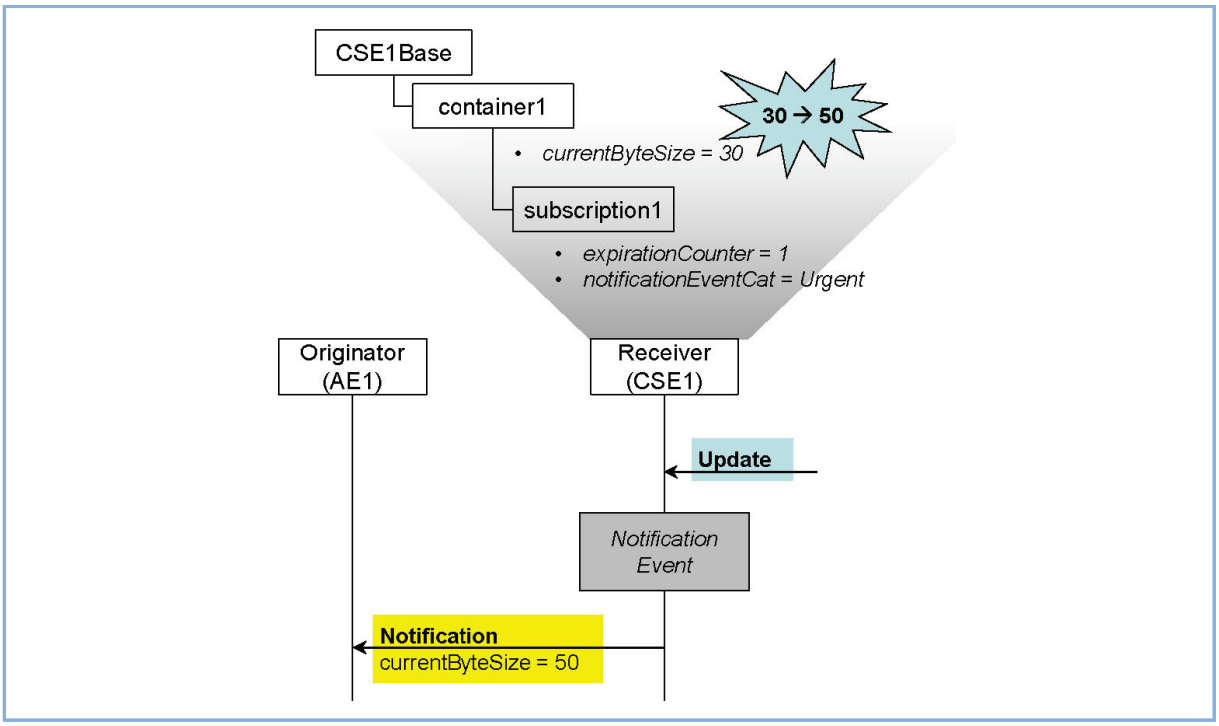



그림 〈3-23〉 구독/통지 기능 예시

구독 기능은 구독 및 통지에 대한 설정 정보를 구독 자원으로 생성하여 이용할 수 있다. 이 자원은 구독 대상 자원의 자녀 자원으로 생성한다. 아래 도면과 같이 요청자가 container1 자원에 관심이 있어 추후 container1 자원의 변경 사항에 대한 통지를 받고자 하는 경우 통지 타입의 자원을 생성한다. 예를 들어 container1 자원의 크기가 30Byte에서 50Byte로 변경되는 이벤트가 발생하면 CSE1은 이러한 정보를 통지 메시지로 전송한다.

구독 자원의 속성은 크게 구독 속성(예: expirationCounter), 통지 속성(예: notificationEventCat), 통지 주소(예: notificationURI) 및 통지 이벤트 필터(예: notificationEventCriteria)로 나눌 수 있다.

표 〈3-8〉는 〈subscription〉 자원의 주요 속성을 나타낸다.

표 〈3-8〉 〈subscription〉 자원의 주요 속성 정보

| 속성 | 설명 |
|---|---|
| eventNotificationCriteria | Subscribed-to 자원에 발생하는 Event 중 특정 조건을 만족하는 Notificaction을 수신하고자 할 때 설정 |
| expirationCounter | 해당 Counter 만큼의 Notification 전송 시 Subscription 자원 삭제됨 |
| notificationURI | Notification이 전송되는 URI |
| batchNotify | 특정 시간/개수의 Notification을 통합하여 Batch Notification으로 수신하고자 할 때 설정 |
| rateLimit | 특정 시간 동안 특정 개수 이상의 Notification 수신을 제한하기 위해 설정. |
| notificationEventCat | Notification에 설정되는 QoS 카테고리 정보 |
| pendingNotification | Reachability 및 Schedule에 의해 전송되지 못한 Notification을 재전송하기 위한 Policy 정보 |
| notificationStoragePriority | Notification이 바로 전송되지 않고 저장될 때 저장 우선 순위 정보 |
| notificationContentType | Notification 메시지에 포함되는 데이터 구성. (E.g., Modificatied Attribute Only, Whole Resource) |

## 3.13. 다중 데이터/서비스를 단일 그룹으로 접근하기

그룹 기능은 그룹 멤버인 하나 이상의 자원에 대한 간편한 접근 기능을 제공한다. 예를 들어 100개 자원에 대한 획득 100번의 획득 요청을 해당 그룹 자원에 대한 단일 획득 요청으로 대신하여 100개 자원의 정보를 획득하는 것이다. 따라서 이러한 그룹 멤버 단일 접근 기능을 이용하기 위해서는 우선 멤버 자원들을 그룹 자원으로 생성해야 한다.

그룹 자원은 기본적으로 멤버 목록과 그룹 요청을 전송할 수 있는 가상 자원으로 구성된다. 그룹 기능을 이용하려는 요청자는 우선 그룹 멤버 자원의 목록과 그룹 속성을 명시하여 그룹 자원 생성을 요청한다. 성공적으로 그룹 자원이 생성되면 가상 자원(fanOut)이 자동적으로 생성되며 해당 자원에 생성/획득/갱신/삭제/통지 요청을 전송하면 그룹을 가진 CSE는 이 요청을 해당 그룹의 모든 멤버 자원으로 전파(fan-out)한다. 이 때 전파 요청의 요청자 정보(fr)은 본래 요청자(아래 그림의 경우 AE01)의 정보가 유지된다.

전파된 요청을 수신한 멤버들은 이를 본래 요청자가 전송한 요청으로 간주하고 접근 제어를 거쳐 해당 요청을 수행한다. 이후 요청 수행 결과는 다시 그룹 소유 CSE로 전달되며, 그룹 소유 CSE는 멤버로부터 전달된 응답 메시지를 종합하여 본래 요청자에 전달한다.

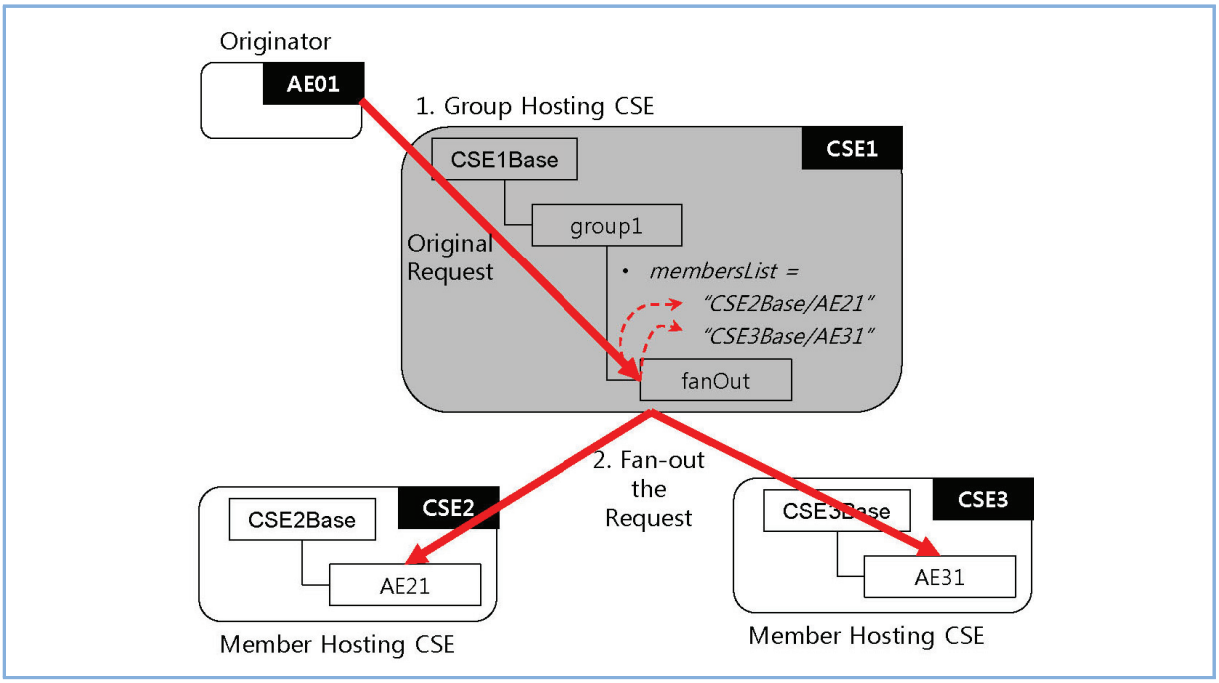



그림 〈3-24〉 그룹 기능 예시

그룹을 통해 그룹 멤버 자원에 자녀 자원을 생성할 때 구독 자원에 대해서는 추가적인 기능을 제공한다. 요청자는 그룹을 통해 모든 멤버에 동일한 구독 자원을 생성할 수 있고 이를 통해 그룹 구독 생성자는 그룹 멤버의 모든 변경 사항을 동일한 통지 주소로 수신할 수 있다. 이 때 각 멤버로부터의 개별 통지 메시지는 그룹 소유 CSE가 종합하여 전달해줄 수 있다.

표 〈3-9〉는 〈group〉 자원의 주요 자원 및 속성을 나타낸다.

표 〈3-9〉 〈group〉 자원의 자식 자원 및 속성 정보

| 자식 자원 | 설명 |
|---|---|
| 〈fanOutPoint〉 | 모든 그룹 멤버에 Batch Request를 수행하기 위한 주소 |
| **속성** | **설명** |
| memberType | 멤버 자원의 자원 타입, 동일하지 않을 경우 "mixed" |
| membersIDs | 멤버 자원에 대한 링크 |
| currentNrOfMembers | 현재 멤버 수 |
| maxNrOfMembers | 최대 멤버 수 |
| membersAccessControlPollicyIDs | fanOut 자원을 통해 모든 멤버에 Request를 fan-out하기 위한 권한 정보 |
| memberTypeValidated | 모든 멤버의 Resource Type이 Validation되면 TRUE |
| consistencyStrategy | memberTypeValidated가 FALSE인 경우 그룹 관리 방법 (E.g., Validation 실패한 멤버 삭제) |

## 3.14. 어플리케이션 간 데이터 공유 예시

그림 〈3-25〉는 지금까지 서술한 oneM2M 플랫폼이 제공하는 기능들이 동작하는 과정을 두 어플리케이션이 데이터를 공유하는 예시를 통해 나타낸다. 온도 센서 어플리케이션은 온도 센싱 데이터를 게이트웨이의 데이터 컨테이너에 저장한다. 그리고 이 데이터는 서버에 연결된 외부 어플리케이션이 해당 컨테이너의 데이터를 획득하는 과정이다.

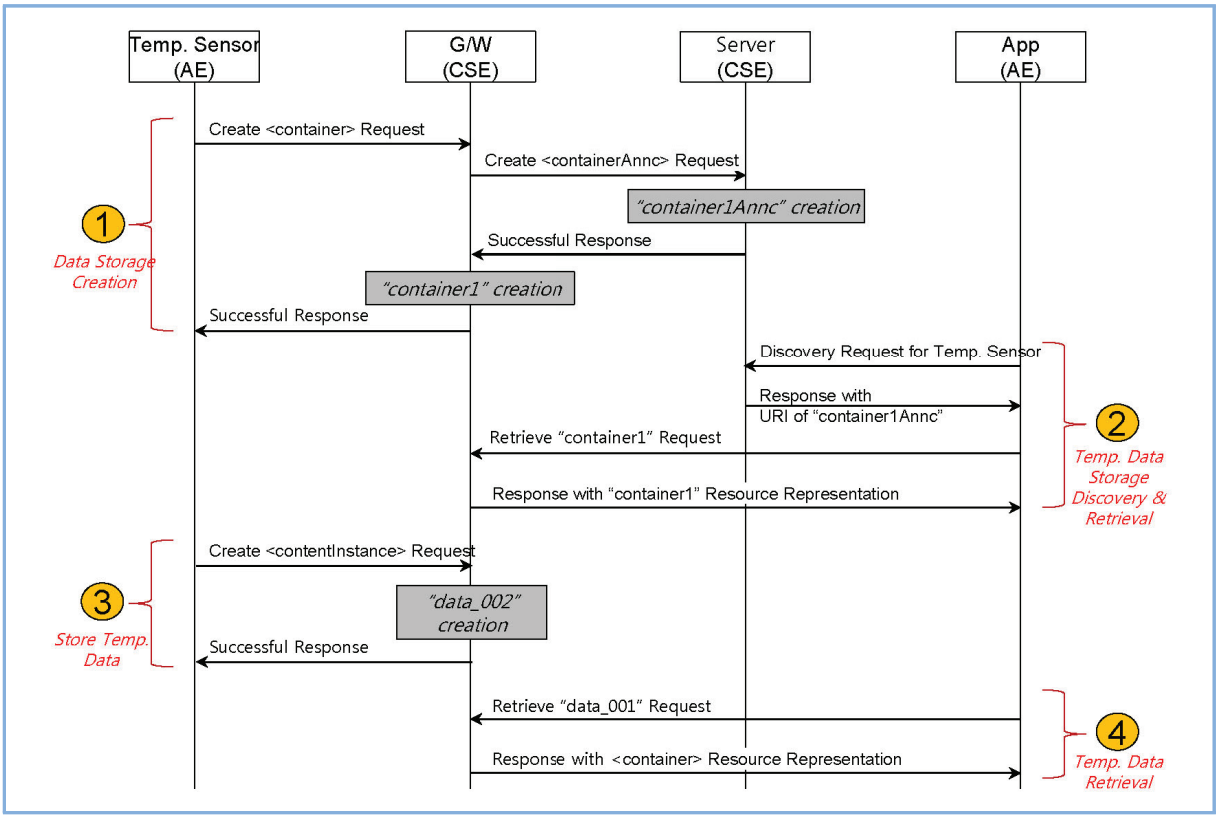



그림 〈3-25〉 데이터 공유 예시

- ✓ 블록1. 센서 어플리케이션은 데이터 저장을 위한 컨테이너 자원을 게이트웨이 CSE에 생성한다. 또한 컨테이너 생성 시 다른 어플리케이션이 이를 쉽게 발견할 수 있도록 서버 CSE에 컨테이너의 공지(announce) 자원을 생성한다. 이 때 컨테이너 및 컨테이너 공지 자원에는 검색 키워드로 사용될 수 있는 라벨(label) 정보를 포함하고 있다. 이 라벨 정보는 현재 생성 중인 센서 데이터에 대한 키워드를 포함한다고 가정한다.
- ✓ 블록 2. 외부 어플리케이션은 특정 조건의 센서 데이터에 관심 있으며 관련 정보를 검색 조건으로 포함하는 발견 요청을 서버 CSE에 전송한다. 서버 CSE는 센서 어플리케이션이 생성한 컨테이너의 공지 자원 정보를 매칭하여 이 자원의 주소를 반환한다. 외부 어플리케이션은 공지 자원에 포함된 원본 컨테이너 자원 정보를 획득하여 게이트웨이에 저장되어 있는 컨테이너의 속성 정보를 획득한다.

- ✓ 블록 3. 센서 어플리케이션은 센서로부터 획득한 온도 데이터를 컨테이너에 컨텐트 인스턴스 자원으로 생성한다.
- ✓ 블록 4. 외부 어플리케이션은 컨테이너에 저장되어 있는 인스턴스 자원을 획득한다.

도면에는 도시되어 있지 않지만 외부 어플리케이션이 컨테이너에 구독 자원을 생성하는 경우 컨테이너에 센서 어플리케이션이 온도 데이터를 컨텐트 인스턴스 자원으로 저장할 때마다 해당 자원을 통지 메시지로 획득하는 동작도 가능하다.

## 3.15. 요약

지금까지oneM2M서비스 플랫폼이 IoT/M2M 어플리케이션에 어떠한 기능을 제공할 수 있는지 살펴보았다. 여기에는 장치/어플리케이션 등록부터 데이터 저장/공유, 접근 관리, 위치 정보 획득, 검색, 구독/통지 및 그룹과 같은 기본적인 기능을 서술하였다. 실제 oneM2M 아키텍처 기술 규격은 장치 관리, 과금, 3GPP 네트워크 연동 등의 보다 다양한 기능을 포함하고 있다.

oneM2M 표준 플랫폼은 특정 IoT/M2M 서비스 영역을 위한 수직적(vertical) 플랫폼이 아닌 다양한 서비스 영역에서 사용할 수 있는 수평적(horizontal) 플랫폼을 지향하여 다양한 어플리케이션에서 사용되는 공통적인 기능을 플랫폼의 기능으로 제공한다. 따라서 oneM2M 플랫폼을 이용하는 어플리케이션은 개발이 용이하다. 또한 oneM2M 플랫폼 기반 어플리케이션 간에 서로 자원(데이터/서비스)을 공유할 수 있으며 이를 통해 다양한 통합(convergence) 서비스/어플리케이션을 제공할 수 있다.

# 4. 사물들 간에 서로 소통하는 방식 (프로토콜)

### 관련 oneM2M 표준 문서

TS-0004: Core Protocol
TS-0008: CoAP Protocol Binding
TS-0009: HTTP Protocol Binding
TS-0010: MQTT Protocol Binding
TR-0009: oneM2M Protocol Analysis

본장에서는 oneM2M에서 개발된 프로토콜 문서들에 대해 살펴보도록 한다. 현재 oneM2M에서 개발된 프로토콜 관련 문서들은 1개 기술 보고서 및 4개의 기술 규격서가 개발 완료되거나 진행 중에 있다. (2014년 11월을 기준으로 TR-0009 (완료), TS-0004, 0008, 0009, 0010 (개발 중))

먼저, 기술 보고서로는 oneM2M이 적용될 수 있는 산업분야에서 현재 사용되고 있는 통신 프로토콜에 대한 소개 및 분석하는 프로토콜 분석 기술 보고서(Protocol Analysis)가 있다. 기술 보고서에서는 CoAP(Constrained Application Protocol), MQTT(Message Queuing Telemetry Transport), TIA TR-50 Protocol, XMPP(eXtensible Messaging and Presence Protocol), WebSocket, Bluetooth, DDS(Data Distribution Service), Modbus, DNP3(Distributed Network Protocol-3), UPnP(Universal Plug and Play) Cloud, ISA100.11a, WirelessHART 등과 같은 프로토콜에 대하여 분석하였다. 또한 기술 규격서로는 oneM2M에서 정의하는 위한 통신 프로토콜을 위한 코어 프로토콜 기술 규격서(Core Protocol), oneM2M 프리미티브(Primitive)와 전송 계층 프로토콜(Transport Layer Protocol) 메시지 간의 매핑(Mapping)을 정의하는 CoAP 프로토콜 바인딩 기술 규격서(CoAP Protocol Binding), HTTP 프로토콜 바인딩 기술 규격서(HTTP Protocol Binding) 및 MQTT 프로토콜 바인딩 기술 규격서 (MQTT Protocol Binding)가 있다. 특히, 본 장에서는 기술 규격서를 중심으로 사물들 간의 통신이 어떤 식으로 이루어지는지, 그리고 코어 프로토콜 기술 규격서와 각 바인딩 기술 규격서 간의 관련성에 대해 살펴보도록 한다.

## 4.1. 프로토콜 개요

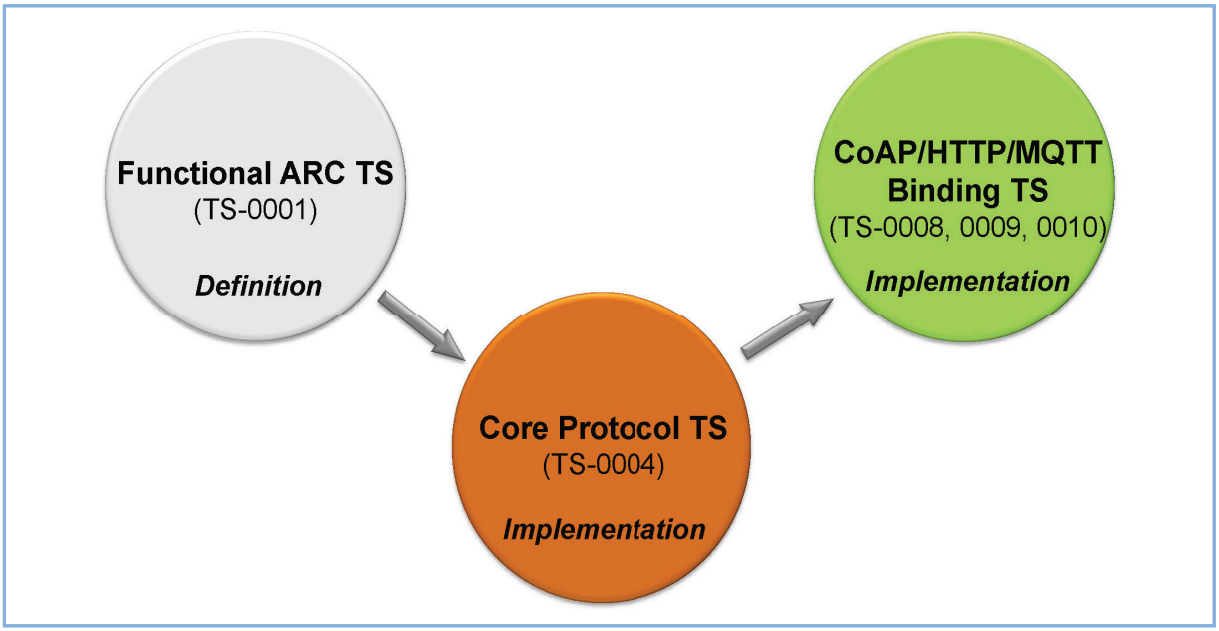



그림 〈4-1〉 프로토콜 기술 규격서 개요

그림 〈4-1〉은 아키텍처 기술 규격서(TS-0001) 및 프로토콜 기술 규격서들(TS-0004, 0008, 0009 및 0010) 간의 관계를 보여준다. 앞서 2장에서 살펴본 것과 같이 아키텍처 기술 규격서는 oneM2M에서 규정하는 기능적 구조를 정의하는 문서이다. 코어 프로토콜 기술 규격서(TS-0004)는 아키텍처 기술 규격서에서 정의한 사항들에 대하여, 개발자들이 실제 구현을 할 때 어떤 식으로 개발이 진행되어야 하는지 정의한 문서이다. 즉, 아키텍처 기술 규격서와는 달리 코어 프로토콜 기술 규격서에서는 새로운 정보 및 절차 등을 규정하는 것이 아니라, oneM2M에서 규정하는 정보 및 절차 등을 개발자가 어떤 데이터 타입을 사용하여 어떤 메시지 형태로 표현할 지에 대하여 서술한다.

만약, oneM2M에 의하여 구현된 메시지가 계층에 관계없이 통신이 가능하다고 가정하면, 코어 프로토콜 기술 규격서만으로 구현된 oneM2M 시스템을 이용하여 서비스가 제공될 수 있을지도 모른다. 그러나 실제 통신 환경은 물리 계층(Physical Layer), 데이터 링크 계층(Data Link Layer), 네트워크 계층(Network Layer) 및 전송 계층(Transport Layer) 등과 같이 다양한 통신 계층들로 구성되어 있다. 특히, oneM2M에서 정의되는 기술 규격은 전송 계층 상위의 존재하는 어플리케이션 및 서비스 계층(Application/Service Layer)을 대상으로 한다. oneM2M은 초기 릴리즈에서 CoAP, HTTP 그리고 MQTT 프로토콜들을 바인딩 대상 프로토콜로 선정하였다. 그러므로 CoAP, HTTP 및 MQTT 바인딩 기술 규격서(TS-0008, 0009, 0010)는 oneM2M이 각 프로토콜을 사용하여 통신이 이루어지는 상황을 고려하여, 해당 프로토콜의 메시지와 oneM2M 프리미티브 간의 매핑을 정의한다.

## 4.2. 프리미티브 및 데이터 타입

아키텍처 기술 규격서에서 서비스를 위하여 정의된 자원(Resource) 및 속성(Attribute) 등은 프로토콜 기술 규격서 측면에서 프리미티브(Primitive)와 데이터 타입(Data Type)을 통하여 표현된 후, 발신자(Originator)와 수신자(Receiver) 사이에서 송수신된다. 그러므로 프로토콜 기술 규격서에 대한 설명에 앞서 프리미티브와 데이터 타입에 대하여 먼저 살펴보도록 한다.

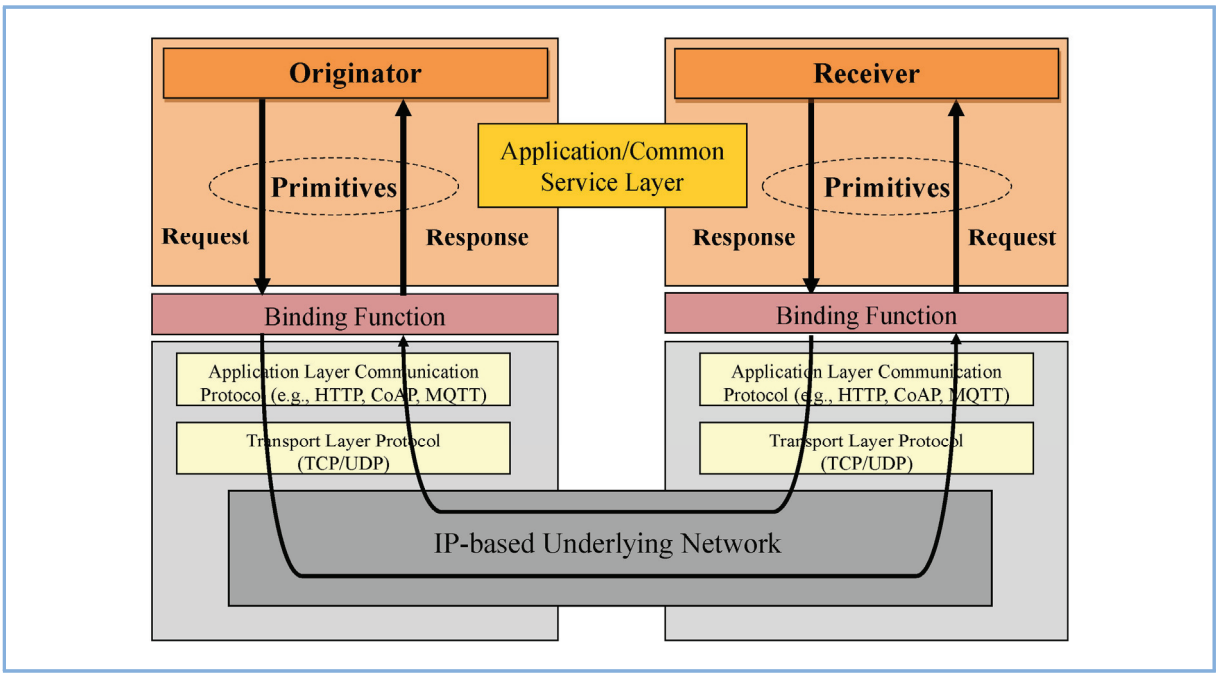



그림 〈4-2〉 프리미티브 개요

그림 〈4-2〉는 코어 프로토콜 기술 규격서에 정의된 프리미티브의 개요를 보여준다. 어플리케이션/공통 서비스 계층(Application/Seervice Lauer)에 있는 발신자로부터 생성된 요청(Request)은 해당 응용 계층 통신 프로토콜의 바인딩 함수(Binding Function)를 거쳐 전송 메시지(Transport Message)로 변환된다. 변환된 전송 메시지는 기저 네트워크(Underlying Network)를 통하여 수신자 측에 전달되고, 역바인딩 과정을 통하여 요청으로 변경되어 수신자에게 전달된다. 이 후, 수신자는 요청에 대한 특정 동작을 수행한 후, 해당 요청에 대한 응답(Response)을 발신자에게 전송한다. 이 때, 응답의 변환 과정은 요청과 마찬가지로 바인딩 및 역바인딩 과정을 거치고 발신자에게 전송된다.

그림 〈4-2〉에서 살펴볼 수 있듯이 oneM2M에서 프리미티브는 어플리케이션/공통 서비스 계층에 있는 발신자 및 수신자로부터 요청된 요청, 그리고 해당 메시지에 대한 응답을 의미한다. 구체적으로 oneM2M에서 사용되는 프리미티브는 요청 및 응답을 처리하기 위해 필요한 속성들을 포함하는 컨트롤 부분(control part) 및 부가적인 내용 부분

(optional content part)으로 구성되지만, 본 장에서는 설명의 편의성을 위하여 두 부분을 구분하지 않고 사용한다.

다음으로 데이터 타입이란 프로그래밍에 사용되는 데이터가 어떤 타입인지를 구분하기 위해 사용되는 것을 의미한다. 대표적인 데이터 타입은 글자(String), 정수(Integer), 참/거짓(Boolean) 등이다[1]. oneM2M에서 사용되는 데이터 타입은 크게 공통 데이터 타입(Simple Data Type)과 추가 데이터 타입(Additional Data Type)으로 구분할 수 있다.

먼저, 공통 데이터 타입은 산업 시장에서 널리 사용되고 있는 XML Schema Language (XSD)를 재사용하는 데이터 타입이다[2]. oneM2M 코어 프로토콜 기술 규격서에서는 공통 데이터 타입을 접두사 'xs:'를 사용하여 표현한다. 예를 들어, 'xs:interger'는 XSD에 정의된 정수형 데이터 타입을 의미한다. 표 〈4-1〉은 oneM2M에 있는 대표적인 공통 데이터 타입을 정리한 것이다.

**표 〈4-1〉 공통 데이터 타입**

| 데이터 타입 | 설명 |
|---|---|
| xs:string | XML에서 글자를 표현 |
| xs:boolean | 논리적인 참/거짓을 표현 |
| xs:decimal | 십진수로 표현되는 실수를 표현 |
| xs:duration | 시간 주기를 표현 |
| xs:anyURI | Internationalized Resource Identifier Reference (IRI)를 표현 |
| xs:hexBinary | arbitrary hex-encoded binary data를 표현 |
| xs:token | Tokenized string을 표현 |
| xs:language | BCP 47에 정의된 formal natural language identifiers를 표현 |

다음으로 공통 데이터 타입 이외에 oneM2M에서 정의한 추가 데이터 타입이다. oneM2M 코어 프로토콜 기술 규격서에서는 추가 데이터 타입을 접두사 'm2m:'을 사용하여 표현한다. 예를 들어, 'm2m:resourceType'은 oneM2M에서 정의한 자원 타입(Resource Type)을 나타내는 데이터 타입을 의미한다. 특히, 추가 데이터 타입은 특정 순서로 배열된 값의 집합으로 이루어진 열거 데이터 타입(Enumeration Data Type), 공통 데이터 타입과 추가 데이터 타입이 복합적으로 구성된 복합 데이터 타입(Complex Data Type)이 있다. 표 〈4-2〉는 대표적인 추가 데이터 타입을 정리한 것이다.

1) http://en.wikipedia.org/wiki/Data_type
2) http://www.w3.org/2001/XMLSchema

표 〈4-2〉 추가 데이터 타입

| 데이터 타입 | 설명 |
|---|---|
| m2m:id | oneM2M에서 사용되는 ID |
| m2m:acpType | AccessControlPolicy identifier와 관련된 데이터 타입. 해당 데이터 타입은 URI이거나 opaque token으로 표현 |
| m2m:resourceType | 해당 자원을 타입과 관련된 데이터 타입. 사용 가능한 값은 다음과 같음<br>1 – accessControlPolicy<br>2 – AE<br>3 – container<br>4 – contentInstance<br>5 – CSEBase<br>6 – delivery<br>7 – eventConfig<br>8 – execInstance<br>9 – fanOutPoint<br>10 – group<br>11 – localPolicy<br>12 – m2mServiceSubscription<br>13 – mgmtCmd<br>14 – mgmtObj<br>15 – node<br>16 – nodeInfo<br>17 – pollingChannel<br>18 – remoteCSE<br>19 – request<br>20 – schedule<br>21 – statsCollect<br>22 – statsConfig<br>23 – Subscription |
| m2m:responseType | 〈request〉 자원 및 요청 프리미티브에 있는 rt 파라미터를 위해 사용되는 데이터 타입, 사용 가능한 값은 다음과 같음<br>1 – nonBlockingRequestSynch<br>2 – nonBlockingRequestAsynch<br>3 – blockingRequest |
| m2m:cseTypeID | 〈CSEBase〉 자원의 cseType 속성을 위해 사용되는 데이터 타입, 사용 가능한 값은 다음과 같음<br>1 – IN-CSE<br>2 – MN-CSE<br>3 – AEN-CSE |
| m2m:eventCat | 〈delivery〉 자원에서 사용되는 ec 파라미터 및 〈subscription〉 자원의 eventCat 속성을 위해 사용되는 데이터 타입 |

앞에서 언급한 것과 같이 코어 프로토콜 기술 규격서는 아키텍처 기술 규격서에서 정의한 사항들에 대하여, 개발자들이 실제 구현을 할 때 어떤 식으로 개발이 진행되어야 하는지 정의한 문서이다. 이를 프리미티브 및 데이터 타입을 통해 살펴보면, 코어 프로토콜 기술 규격서는 아키텍처 기술 규격서의 자원, 속성 및 명령어 등을 데이터 타입으로 표현한 후 이를 oneM2M 요청 및 응답 프리미티브 형태로 전송하는 방법을 정의한다.

## 4.3. 코어 프로토콜

oneM2M 프로토콜 기술 규격서들 가운데 코어 프로토콜에 대하여 자세히 살펴본다. 코어 프로토콜 규격서에서 oneM2M의 동작은 크게 발신자 및 수신자 절차로 구분되어 설명된다. 그리고 각각의 절차는 자원에 관계없이 공통적으로 적용될 수 있는 부분에 대하여 포괄적 절차(Generic Procedure) 및 자원에 따라 다르게 설정되는 절차(Resource Type-specific Procedure)를 사용하여 설명한다.

### 4.3.1. 발신자 측면에서 코어 프로토콜 절차

먼저 그림 〈4-3〉은 발신자 측면에서 절차를 보여준다. 발신자 측면에서 단계는 6단계로 구성되며, 각 단계는 'Orig-X.0'으로 표현된다. (여기서 X는 임의의 정수를 의미한다.) 특히, 발신자 측면의 절차 가운데 'Orig-1.0'는 자원 타입 및 명령어에 따라 다르게 설정될 수 있는 절차를 의미하며, 나머지 단계들은 공통적으로 적용되는 단계이다.

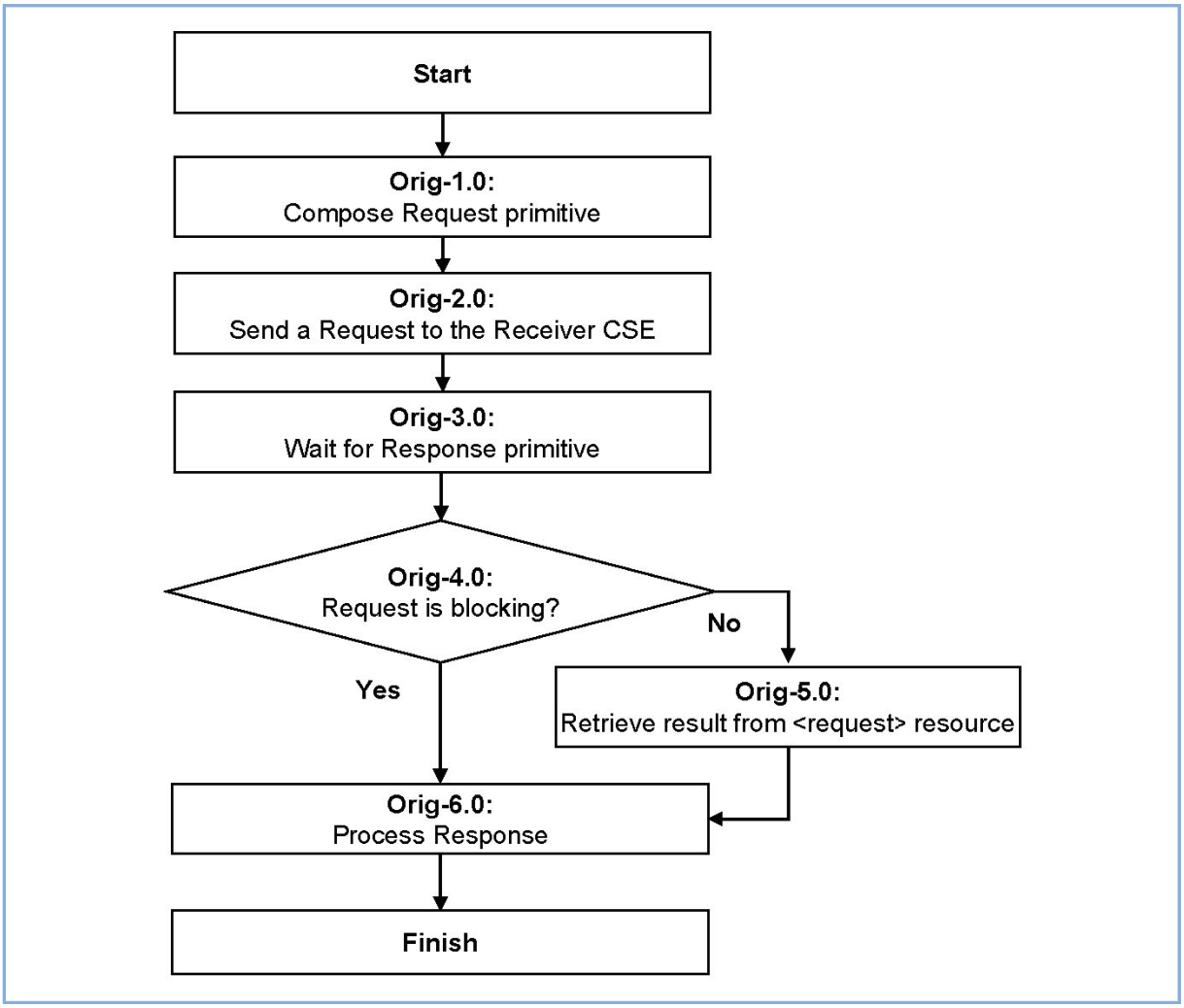



그림 〈4-3〉 발신자 측면에서 포괄적 절차

#### Orig-1.0 "Compose Request Primitive"

발신자는 요청 프리미티브를 생성한다. 이 때, 생성되는 요청 프리미티브는 자원 타입 및 명령어에 따라 각각 다르게 생성될 수 있다. 그러나, 자원의 타입에 관계없이 요청 프리미티브에는 발신자 및 수신자를 의미하는 파라미터 'From(fr)' 및 'To(to)'가 반드시 포함되어야 하며, 추가적으로 발신자의 특성에 따라 선택적인 속성값들이 포함될 수 있다. 또한 생성된 요청 프리미티브는 해당 프로토콜 바인딩 기술 규격서에 맞추어 매핑된다.

### Orig-2.0 "Send a Request to the Receiver CSE"

발신자는 생성된 요청 프리미티브를 수신자 공통 서비스 엔티티로 전송한다. 이 때, 수신자에 해당하는 공통 서비스 엔티티가 없는 경우에는 해당 요청이 거절된다. 또한 수신자에 해당하는 공통 서비스 엔티티가 두 개 이상으로 설정된 경우에도 해당 요청은 거절된다. (oneM2M에서는 단일 수신자인 경우만을 고려한다.)

### Orig-3.0 "Wait for Response primitive"

발신자는 요청 프리미티브를 전송한 후, 해당 요청에 대한 응답이 도달할 때까지 기다린다. 이 때, 서버 정책(Server Policy) 또는 기저 전송 기술(Underlying Transport Technology)에 의하여 설정된 시간 동안 해당 요청에 대한 응답이 없는 경우에는 요청에 대한 응답이 타임 아웃(Time Out)이 된 것으로 처리된다.

### Orig-4.0 "Request is blocking?"

발신자는 요청 프리미티브가 블로킹 모드인지 확인한다. 발신자가 요청 프리미티브의 블로킹 여부를 확인하는 과정은 프리미티브에 따라 추가적으로 〈request〉 자원 타입에 대한 명령이 수행되기 때문이다. 이 때, 해당 요청이 블로킹 모드가 아닌 경우(non-blocking mode)에는 'Orig-5.0' 단계로 이동하고, 블로킹 모드(blocking mode)인 경우에는 'Orig-6.0' 단계로 이동한다.

### Orig-5.0 "Retrieve result from the 〈request〉 resource"

요청 프리미티브가 블로킹 모드가 아닌 경우, 발신자는 이전에 생성한 요청을 검색(Retrieve)하기 위하여 'Orig-5.0' 단계를 수행한다. 이 때, 발신자는 요청 프리미티브를 표 〈4-3〉과 같이 설정한 후, 해당 요청 프리미티브를 수신자 공통 서비스 엔티티에게 전송한다.

표 〈4-3〉 요청 프리미티브 설정 (Origi-5.0)

| 파라미터 이름 | 값 |
|---|---|
| Primitive Type (primitiveType) | REQUEST |
| Operation (op) | Retrieve |
| To (to) | 〈request〉 자원의 URI 세트 |
| From (fr) | 발신자 ID |
| Request Identifier (ri) | 요청 메시지의 ID |
| Content (cn) | Retrieve에 사용되는 속성 이름 (optional) |

### Orig-6.0 "Process Response"

발신자는 수신자로부터 전송받은 응답 프리미티브를 처리한다.

### 4.3.2. 수신자 측면에서 코어 프로토콜 절차

다음으로 수신자 측면에서 절차를 살펴본다. 그림 〈4-4〉는 수신자 측면에서 절차를 나타낸 것이며, 해당 절차는 통신 방식에 따라 두 가지로 구분되어 진행된다. 앞의 발신자 측면과 마찬가지로, 수신자 측면에서 각 단계는 'Recv-X.0'으로 표현된다. (여기서 X는 임의의 정수를 의미한다.) 특히, 그림 〈4-4〉에서 'Recv-6.0'는 자원 타입에 따라 다르게 설정될 수 있는 절차를 의미하며, 나머지 단계들은 공통적으로 적용되는 단계이다.

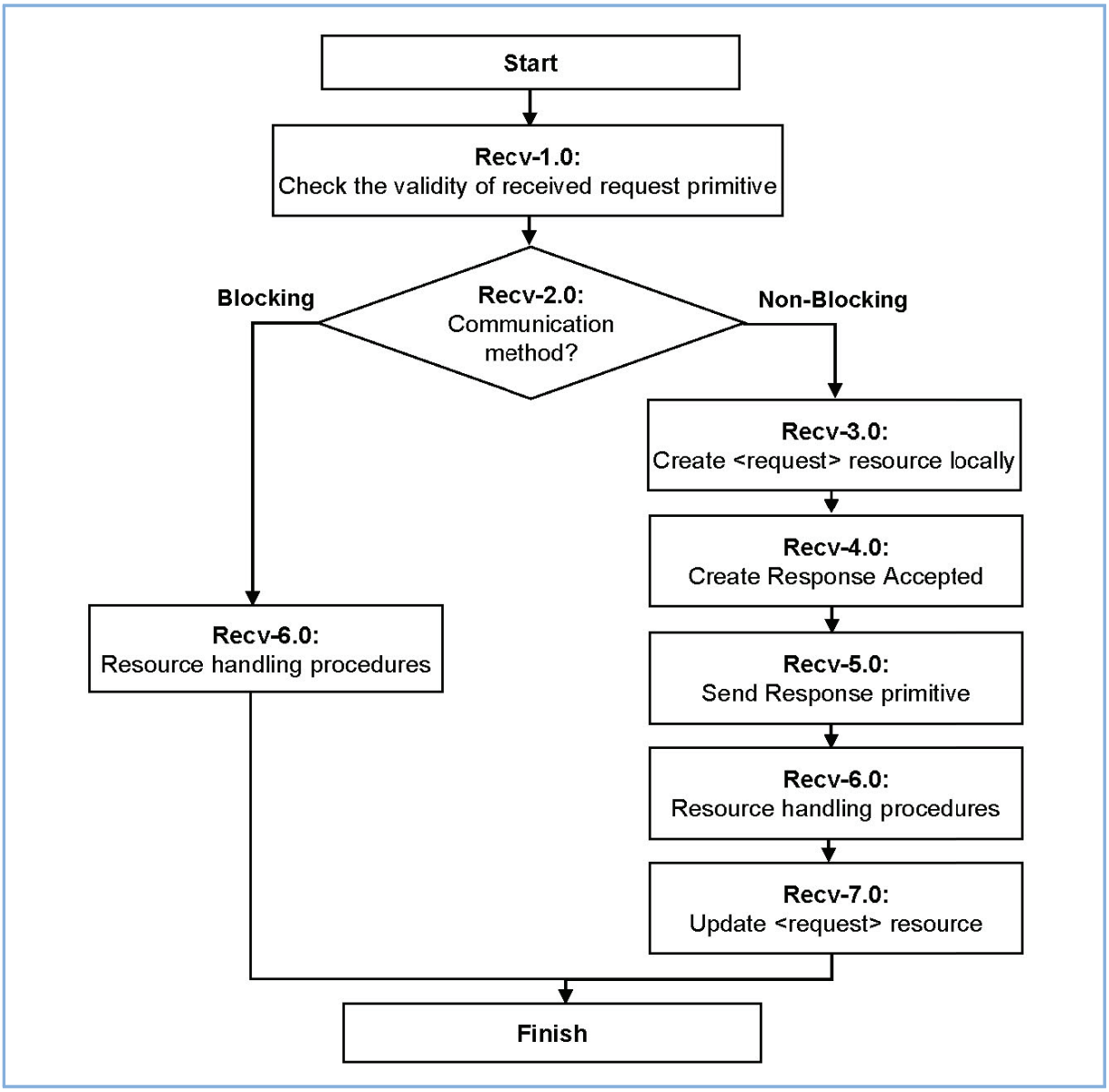



그림 〈4-4〉 수신자 측면에서 포괄적 절차

### Recv-1.0 "Check the validity of received request primitive"

수신자는 수신된 요청 프리미티브의 유효성 검사를 실시한다. 본 단계에서 수신자는 해당 요청이 매핑된 프로토콜 기술 규격서(즉, HTTP/CoAP 및 MQTT 바인딩 기술 규격서)에 적합하게 작성되었는지를 확인한다. 만약, 수신된 요청이 다른 공통 서비스 엔티티로 전달되거나 CDMH 처리가 지원되는 경우에는 추가적으로 "CMDH 메시지 검증 절차"가 수행된다. 또한 본 단계에서 해당 요청 프리미티브가 유효하지 않을 경우에는 해당 요청은 거절된다.

### Recv-2.0 "Communication method?"

수신자는 전송 받은 요청 프리미티브가 블로킹 모드인지 블로킹 모드가 아닌지 'Response Message Type(rt)' 파라미터를 통하여 확인한다. 수신자가 요청 프리미티브의 블로킹 여부를 확인하는 과정은 프리미티브에 따라 추가적으로 〈request〉 자원 타입에 대한 명령이 수행되기 때문이다. 이 때, 해당 요청이 블로킹 모드가 아닌 경우에는 'Recv-3.0' 단계로 이동하고, 블로킹 모드인 경우에는 'Recv-6.0' 단계로 이동한다.

### Recv-3.0 "Create 〈request〉 resource locally"

수신자가 〈request〉 자원을 지원할 수 있고 'Response Message Type(rt)' 파라미터가 블로킹 모드가 아닌 경우, 수신자는 〈request〉 자원을 생성한다. 이 때, 본 단계가 적용되는 경우는 nonBlockingRequestSynch 또는 nonBlockingRequestAsynch이다.

### Recv-4.0 "Create Response Accepted"

블로킹 모드가 아닌 경우에 수신자는 짧은 시간에 응답 프리미티브를 발신자에게 전송할 수 없다. 그러므로, 수신자는 acknowledge를 사용하여 발신자에게 응답 프리미티브를 전송한다.

### Recv-5.0 "Send Response Primitive"

수신자는 응답 프리미티브를 발신자에게 전달한다.

### Recv-6.0 "Resource handling procedure"

발신자로부터 송신된 요청 프리미티브에 있는 자원 타입에 따라 명령이 수행되는 과정이다. 본 단계에 대한 Recv-6.1부터 Recv-6.11로 구성된다. (해당 절차는 3.3.3에서 자세하게 살펴본다.)

Recv-7.0 “Update 〈request〉 resource”

‘Recv-6.0’에 대한 과정이 종료된 후, 수신자는 〈request〉 자원을 갱신한다. 이 때, 갱신되는 일반 속성(Common Attribute) 및 자원에 따른 속성(Resource-specific Attribute)은 다음 표 〈4-4〉와 같다.

표 〈4-4〉 자원에 따른 절차가 종료된 후 갱신되는 〈request〉 자원의 속성

| 속성 이름 | 값 |
|---|---|
| lastModifiedTime | 마지막으로 변경된 날짜 및 시간 |
| stateTag | 모든 변경마다 증가됨 |
| requestStatus | 호스팅 공통 서비스 엔티티(Hosting CSE)는 관련된 non-blocking request 적절한 상태 값으로 설정 |
| operationResult | 호스팅 공통 서비스 엔티티는 원래 요청된 명령의 결과 값으로 설정 |

### 4.3.3. 수신자 측면에서 자원 처리 절차

마지막으로 수신자 측면의 절차 가운데 Recv-6.0 “Resource handling procedure”, 즉 자원 타입에 따른 처리 절차를 자세히 살펴본다. 그림 〈4-5〉는 자원 타입에 따른 처리 절차를 표현한 것으로, 해당 그림에서 각 단계는 ‘Recv-6.X’으로 표현된다. 수신자 측면에서 자원 처리 절차는 자원 타입에 따라 공통적으로 설정될 수 있는 것과 다르게 설정되어야 하는 것으로 구분될 수 있으며, 이는 구체적으로 Recv-6.1부터 Recv-6.11로 구성된다.

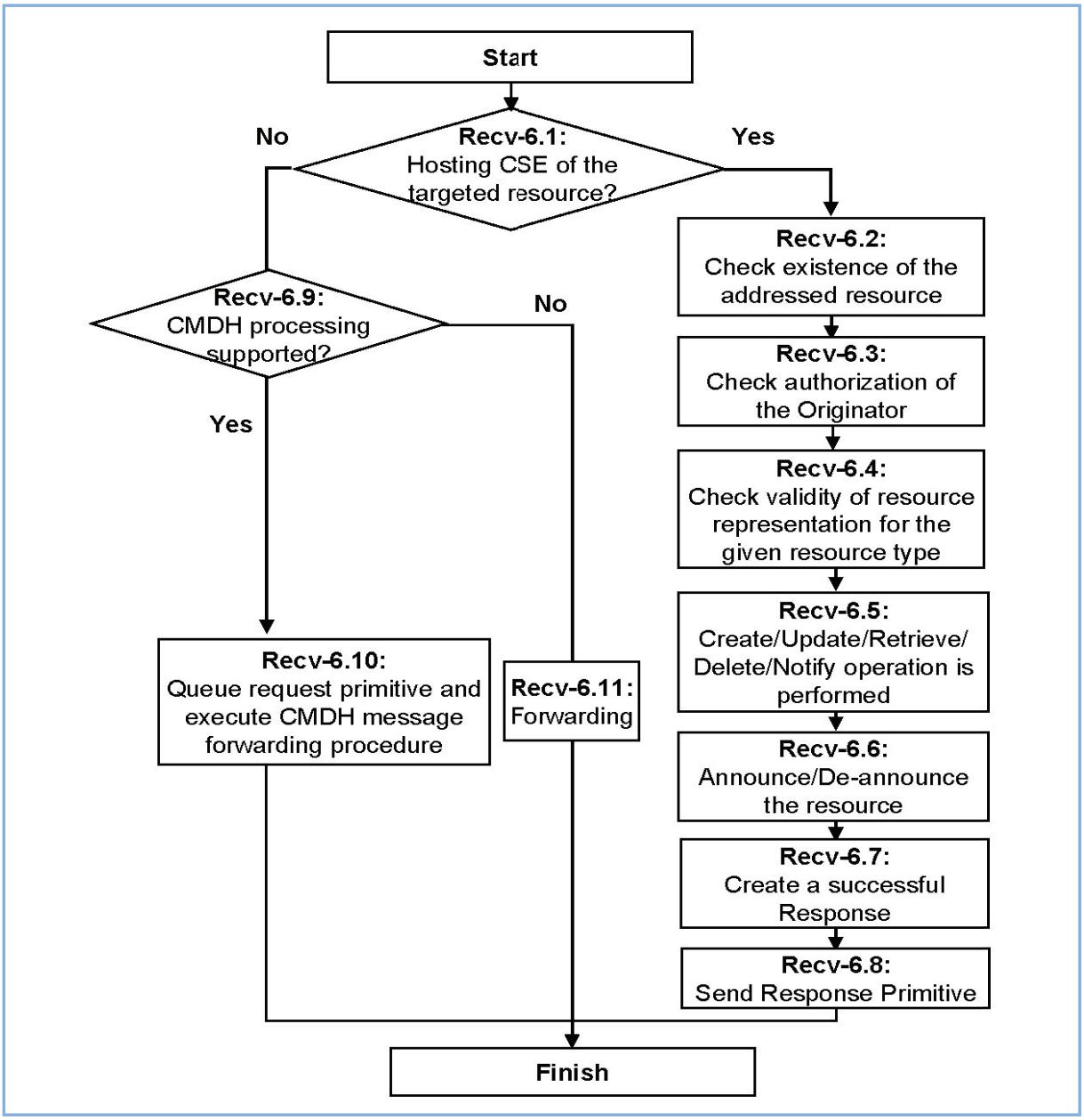



그림 〈4-5〉 자원 타입에 따른 처리 절차

### Recv-6.1 "Hosting CSE of the targeted resource?"

수신자는 전송 받은 요청 프리미티브의 'To(to)' 파라미터를 통하여 해당 공통 서비스 엔티티가 최종 목적지인지 여부를 확인한다. 수신자가 해당 공통 서비스 엔티티의 유형을 확인하는 과정은 최종 목적지와 중계 공통 서비스 엔티티에 따라 동작이 다르기 때문이다. 이 때, 해당 공통 서비스 엔티티가 최종 목적지인 경우에는 'Recv-6.2' 단계로 이동하고, 중계 공통 서비스 엔티티인 경우에는 'Recv-6.9' 단계로 이동한다.

### Recv-6.2 "Check existence of the addressed resource"

수신자는 전송 받은 요청 프리미티브의 'To(to)' 파라미터의 자원이 존재하는지 확인한다. 이 때, 해당 자원이 존재하지 않을 경우 요청 프리미티브는 거절된다.

### Recv-6.3 "Check authorization of the Originator"

수신자는 요청 프리미티브에 대한 권한여부를 확인한다. 이 때, 수신자는 발신자의 권한을 확인하기 위하여 그림 〈4-6〉과 같이 요청된 자원의 accessControlPolicyIDs 속성에 있는 주소를 참고하여 accessControlPolicy 의 privileges 값을 통하여 권한 여부를 확인한다. (그림 〈4-6〉에서 'App_AE'는 해당 자원에 대하여 Retrieve 및 Delete 권한이 있다는 것을 보여준다.)

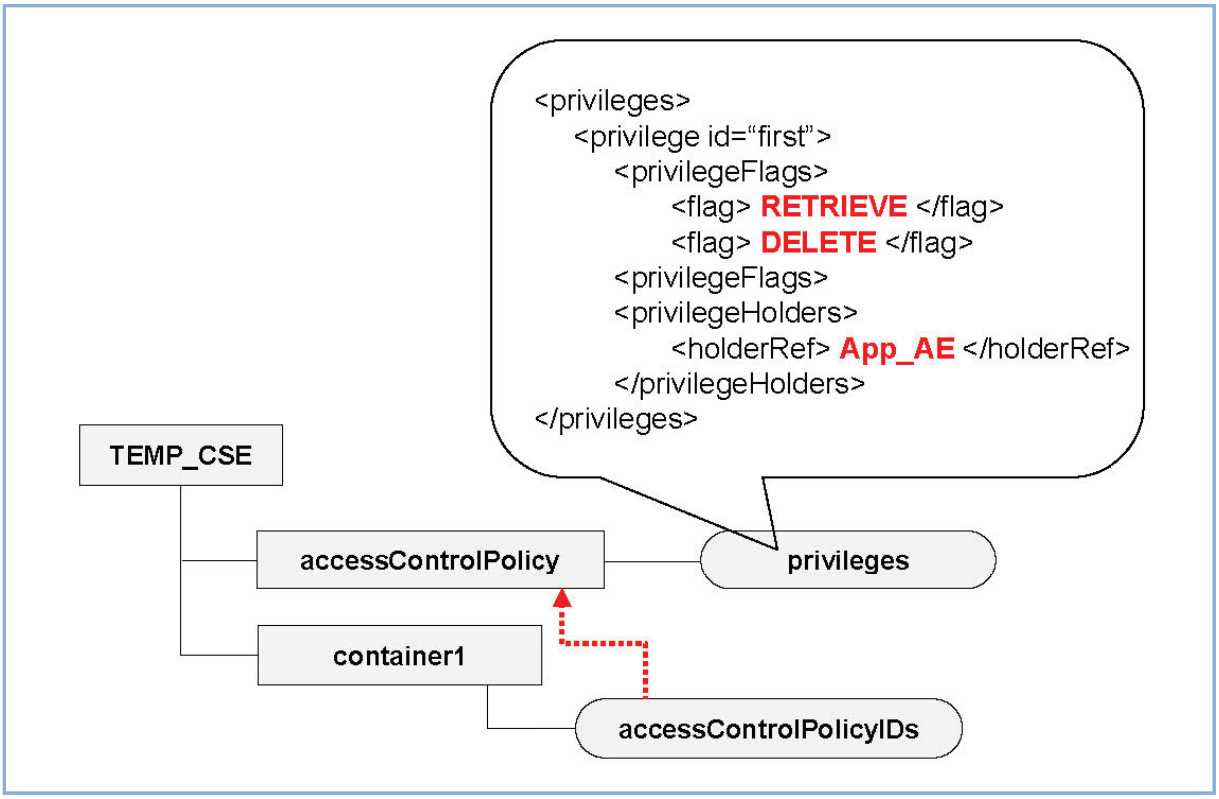



그림 〈4-6〉 발신자의 권한 확인

### Recv-6.4 "Check validity of resource representation for the given resource type"

수신자는 요청 프리미티브에 포함된 자원 타입에 따라 유효성을 검증한다. 본 단계에서 유효성 검증은 생성(Create) 및 갱신(Update)에 따라 다르게 실시되며, Notify (통지) 명령어에 대해서는 본 단계가 적용될 수 없다. 유효성 검증 과정에서 특정 과정을 만족하지 못하면, 다른 속성들은 검증되지 않는다.

### Recv-6.5 “Create/Update/Retrieve/Delete/Notify operation is performed”

수신자는 요청 프리미티브에 포함된 동작(Create / Update / Retrieve / Delete / Notify)을 수행한다.

### Recv-6.6 “Announce/De-announce the resource”

발신자가 요청(Announce) 및 어나운스 해지(De-announce)에 대한 동작을 요청하였을 경우, 수신자는 해당 어나운스 동작을 수행한다. 어나운스와 관련된 동작은 ‘announceTo’ 및 ‘announcedAttribute’ 속성을 사용하여 설정된다.

### Recv-6.7 “Create a successful Response”

수신자는 발신자로부터 전달받은 요청 프리미티브에 대한 동작을 성공적으로 수행한 후, 응답 프리미티브를 생성한다. 생성된 응답 프리미티브에는 생성된 자원의 URI가 포함된다.

### Recv-6.8 “Send Response Primitive”

수신자는 생성된 응답 프리미티브를 발신자에게 전송한다.

### Recv-6.9 “CMDH processing supported?”

수신자는 자신이 CMDH와 관련된 처리를 할 수 있는지 확인한다. 이 때, CMDH와 관련된 처리를 수행할 수 있는 경우에는 ‘Recv-6.10’ 단계로 이동하고, 그렇지 않은 경우에는 ‘Recv-6.11’ 단계로 이동한다.

### Recv-6.10 “Queue request primitive and execute CMDH message forwarding procedure”

수신자가 CMDH와 관련된 처리를 할 수 있는 경우, 수신자는 수신된 요청 프리미티브를 큐(Queue)하고, CMDH 메시지 전달과정에 따라 해당 요청을 처리한다.

### Recv-6.11 “Forwarding”

수신자가 CMDH와 관련된 처리를 할 수 없는 경우에는 요청 메시지를 다음 공통 서비스 엔티티로 전달한다.

## 4.4. HTTP Binding

HTTP(HyperText Transfer Protocol)란 웹(Web) 상에서 클라이언트(Client)와 서버(Server) 사이에 이루어지는 요청 메시지(Request) 및 응답 메시지(Response)로 구성된 통신 프로토콜이다. 예를 들어, 클라이언트인 웹 브라우저(Web Browser)가 HTTP를 통하여 서버로부터 웹 페이지(Web Page)나 동영상 정보를 요청하면 서버는 이 요청에 응답하여 필요한 정보를 해당 사용자에게 전달되고, 전달된 정보가 출력 장치를 통해 사용자에게 나타나는 것이다.

oneM2M Rel-1의 HTTP 바인딩 규격서는 RFC7230[3]에 명시된 HTTP/1.1 프로토콜에 대응하는 바인딩 규격을 제공한다.

### 4.4.1. oneM2M 프리미티브 매핑

단일 oneM2M 프리미티브는 단일 HTTP 메시지와 매핑되며, oneM2M 프리미티브와 HTTP 메시지 간 매핑은 그림 〈4-2〉와 같이 발신자와 수신자의 바인딩 기능에 의해 수행되며 이는 아래의 4가지 경우로 나눌 수 있다.

- ✓ 요청자가 요청 프리미티브를 전송할 때 (요청 프리미티브 → HTTP 요청 메시지)
- ✓ 요청자가 응답 프리미티브를 수신할 때 (HTTP 응답 메시지 → 응답 프리미티브)
- ✓ 수신자가 응답 프리미티브를 전송할 때 (응답 프리미티브 → HTTP 응답 메시지)
- ✓ 수신자가 요청 프리미티브를 수신할 때 (HTTP 요청 메시지 → 요청 프리미티브)

HTTP 바인딩 기술 규격서는 oneM2M 프리미티브 파라미터와 HTTP 메시지의 파라미터/헤더 간의 매핑 방법을 규정하며, 매핑 순서는 HTTP 요청/응답 메시지의 구조를 기준으로 정의되어 있다.

### 4.4.2. oneM2M 프리미티브 파라미터 매핑 규칙

oneM2M 프리미티브는 요청과 응답에 대하여 다양한 파라미터를 정의하고 있다. 이 파리미터들은 크게는 HTTP 메시지의 URI, Query, Header 및 Body에 매핑될 수 있다. 따라서 특정 파라미터를 HTTP 메시지의 어떤 엘레먼트에 매핑할지에 대한 규칙이 필요하다.

표 〈4-5〉는 oneM2M 프리미티브 파라미터를 3가지 그룹으로 분류한다. 이는 요청과 응답 프리미티브에 공통으로 존재하는 파라미터와 한 가지에만 존재하는 그룹이다. 이는 HTTP 요청 메시지가 HTTP 응답 메시지와 구성적으로 Request-Line에 Query String을 추가적으로 포함하는 점을 이용하기 위해서이다.

3) "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", IETF, June 2014

기본적으로 요청 프리미티브에만 존재하는 파라미터는 Query String에 매핑하고, 요청 및 응답에 존재하는 파라미터는 헤더/확장헤더에 그리고 응답에만 존재하는 파라미터는 확장헤더에 매핑한다.

표 〈4-5〉 oneM2M 프리미티브 파라미터 분류

| | 파라미터 | 매핑 대상 HTTP 엘레먼트 |
|---|---|---|
| 요청 및 응답 모두 존재 | Primitive Type | n/a |
| | To | Request-Target |
| | From | From (header) |
| | Request Identifier<br>Originating Timestamp<br>Result Expiration Timestamp<br>Event Category | (new extension headers) |
| | Content | Message-body |
| 요청에만 존재 | Operation | Method (POST/GET/PUT/DELETE) |
| | Resource Type | Content-Type (header) |
| | Name<br>Request Expiration Timestamp<br>Operation Execution Time<br>Response Type<br>Result Persistence<br>Result Content<br>Delivery Aggregation<br>Group Request Identifier<br>Filter Criteria<br>Discovery Result Type | Query-String |
| 응답에만 존재 | Response Status Code | (new extension header) |

규칙1: 요청 및 응답 프리미티브 모두에 존재하는 파라미터는 아래 매핑 규칙을 적용할 수 있다.

- ✓ Message-body는 content 파라미터만 매핑한다.
- ✓ Primitive Type은 요청과 응답 메시지로 매핑되므로 별도로 매핑하지 않는다.
- ✓ To는 Request-Target에 매핑한다.
- ✓ From은 From 헤더에 매핑한다.
- ✓ 나머지 파라미터는 확장 헤더에 매핑한다.

규칙2: 요청 프리미티브에만 존재하는 파라미터는 아래 매핑 규칙을 적용할 수 있다.

✓ Operation은 Method에 매핑한다.

✓ Resource Type은 Content-Type 헤더에 매핑한다.

✓ 나머지 파라미터는 Query-String에 매핑한다. 이 때는 각 파라미터의 짧은 이름을 사용한다.

규칙3: 요청 및 응답 모두에 존재하는 파라미터는 아래 규칙을 적용할 수 있다.

✓ Response Status Code는 확장 헤더에 매핑한다.

✓ 본 절에 기술된 파라미터 매핑 규칙은 CoAP 바인딩 규격에도 동일하게 적용된다.

### 4.4.3. HTTP Request-Line 매핑

Request-Line은 Method, Request-Target 및 HTTP-Version으로 구성된다. 먼저 Method는 아래 표 〈4-6〉과 같이 매핑된다. POST Method를 사용하는 HTTP 요청 메시지는 Resource Type 파라미터 값에 따라 Create 동작과 Notify 동작을 구분하여 매핑한다.

〈표 4-6〉 HTTP Method 매핑

| oneM2M Operation | HTTP Method |
|---|---|
| Create | POST |
| Retrieve | GET |
| Update | PUT or POST (for partial update) |
| Delete | DELETE |
| Notify | POST |

Request-Target은 To 파라미터 값이 Absolute-Path에 매핑되어 프리미티브의 요청 대상을 명시하며, 추가적으로 Query-String에 oneM2M 요청 프리미티브의 다양한 파라미터를 매핑한다. 실제 파라미터 이름은 아래 표 〈4-7〉과 같은 짧은 이름의 필드 이름으로 사용된다.

표 〈4-7〉 Query-String 매핑

| Parameter | Field Name |
|---|---|
| Name | *nm* field |
| Request Expiration Timestamp | *ret* field |
| Operation Execution Time | *oet* field |
| Response Type | *rt* field |
| Result Persistence | *rp* field |
| Result Content | *rc* field |
| Delivery Aggregation | *da* field |
| Group Request Identifier | *gid* field |
| Filter Criteria | *fc* field |
| Discovery Result Type | *drt* field |

HTTP 1.1 버전을 지원하므로 HTTP-Version은 "HTTP/1.1"값이 매핑된다.

### 4.4.4. HTTP Status-Line 매핑

Status-Line은 먼저 HTTP-Version으로 "HTTP/1.1"값이 매핑된다. 이어서 Status-Code 값은 Response Status Code 파라미터에 대응하는 HTTP Status-Code 값이 매핑된다. oneM2M의 응답 상태 코드는 HTTP의 코드보다 더욱 다양한 코드가 정의되어 있으므로 HTTP 코드와 1:1로 매핑되지는 않는다. 해당 코드는 별도의 헤더에 별도로 매핑된다.

아래 표 〈4-8〉은 Response Status Code와 HTTP 응답 코드의 매핑 예시를 나타낸다.

표 〈4-8〉 Status-Code 매핑 (일부)

| oneM2M Response Status Codes | HTTP Status Codes |
|---|---|
| Success | 200 (OK) |
| Accepted | 202 (Accepted) |
| Unsupported resource | 404 (Not Found) |
| Cannot forward, target not reachable | 404 (Not Found) |
| Create error – no privilege | 403 (Forbidden) |

또한 위 〈표 4-8〉과 같은Response Status Code의 정보는HTTP Reason-Phrase에 매핑된다.

### 4.4.5. HTTP 헤더 매핑

HTTP 메시지의 헤더 및 확장헤더는 아래와 같이 oneM2M 프리미티브 파라미터와 매핑된다.

표 〈4-9〉 HTTP 헤더 매핑 (1/2)

| Primitive Parameter | Header Name |
|---|---|
| From | From |
| Resource Type | Content-Type |
| Request Identifier | X-M2M-RI |
| Originating Timestamp | X-M2M-OT |
| Result Expiration Timestamp | X-M2M-RST |
| Event Category (eventCatType 사용 시) | X-M2M-ECT |
| Event Category (eventCatNo 사용 시) | X-M2M-ECN |
| Response Status Code | X-M2M-RSC |

표 〈4-9〉의 프리미티브 파라미터 매핑 외에도 다양한 정보가 HTTP 헤더와 매핑된다. 이는 표 〈4-10〉에 요약되어 있다.

표 〈4-10〉 HTTP 헤더 매핑 (2/2)

| Information | Header Name |
|---|---|
| pointOfAccess | Host |
| oneM2M MIME Type | Accept |
| 생성/획득/갱신 응답의 실제 타겟 자원의 URI | Content-Location |
| Message-body의 길이 (Content 파라미터의 길이) | Content-Length |
| 획득한 자원의 Etag 값 | Etag |
| 생성된 자원의 URI | Location |

pointOfAccess는 수신자 CSE의 IP와 같은 물리적인 주소이며, 수신자 CSE는 Core Protocol 규격에서 정의하는 라우팅 방법에 의해 결정된다.

### 4.4.6. 보안 고려 사항

HTTP 바인딩 관련 보안은 아래와 같은 사항을 고려한다.

HTTP 요청 메시지 인증 방법

- ✓ Registrar CSE에 Credential을 전송할 때 Proxy-Authorization 헤더를 사용한다.
- ✓ Hosting CSE에 Credential을 전송할 때 Authorization 헤더를 사용한다.
- ✓ OAuth 2.0의 Access Token을 통해 Hosting CSE에 Credential을 전송할 때, Bearer 인증 기법을 사용한다.
- ✓ 전송 계층 보안 기법 사용
- ✓ TLS(Transport Layer Security)에 의해 Hop-by-Hop 보안을 보장한다. 현재 Rel-1에서 End-to-End 보안은 보장하지 않는다.

### 4.4.7. 기타

아래 사항은 추후 구체적으로 규격화될 HTTP 바인딩 고려 사항이다.

- ✓ HTTP 메시지의 라우팅 기법
- ✓ Registrar CSE의 HTTP Proxy Server 동작
- ✓ WebSocket 이용한 Notification 전송 기법

## 4.5. 코어 프로토콜과 HTTP 간의 바인딩 예

본 장에서는 앞에서 살펴본 코어 프로토콜 및 HTTP 간의 바인딩 예시를 통하여 프로토콜 기술 규격서를 살펴본다.

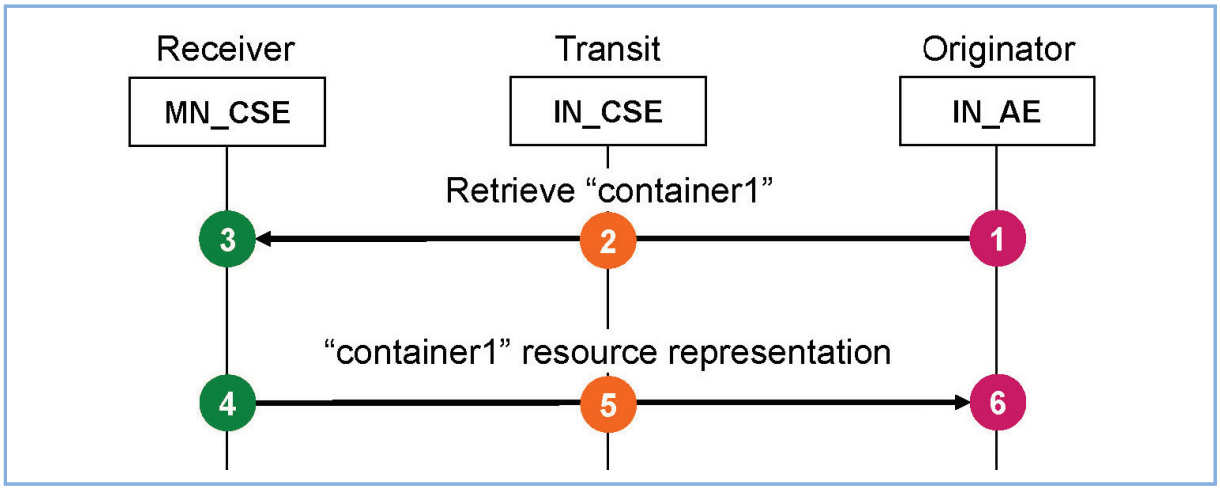



그림 〈4-7〉 코어 프로토콜 및 HTTP 간의 바인딩 예시를 위한 구성

그림 〈4-7〉은 IN_CSE에 등록된 IN_AE가 MN_CSE의 'container1' 자원을 Retrieve하는 예시이다. 상기 예시에서 IN_AE, IN_CSE 및 MN_CSE는 발신자, 중계 엔티티(Transit Entity) 및 수신자를 각각 의미한다. 이 때, 수신자 MN_CSE에는

'container1' 자원이 이미 생성되어 있으며 발신자 IN_AE는 해당 자원에 대한 Retrieve 할 수 있는 권한이 있다. 또한 설명의 편의성을 위하여 본 예제에서 요청 프리미티브는 'Blocking' 모드이며, 모든 엔티티들은 CMDH 처리를 지원하지 않는 것을 가정한다. 상기와 같은 환경에서 발신자, 중계 엔티티 및 수신자의 동작을 살펴보면 각각 다음과 같다.

먼저 그림 〈4-7〉의 ①단계를 위하여 발신자 IN_AE는 수신자 MN_CSE에 있는 'container1' 자원을 Retrieve 하기 위한 oneM2M 요청 프리미티브를 생성하고, 생성된 oneM2M 요청 프리미티브를 HTTP 요청 메시지로 바인딩하여 전송한다. ① 단계는 코어 프로토콜 규격서에서 살펴본 발신자 측면에서 포괄적 절차(그림 〈4-3〉)의 Orig-1.0 ~ Orig-3.0 단계를 의미한다. 이 때, oneM2M 요청 프리미티브는 표 〈4-11〉과 같은 파라미터로 구성된다.

**표 〈4-11〉 발신자에서 oneM2M 요청 프리미티브 예시('container1' Retrieve)**

```
From: IN_CSE/IN_AE
To: MN_CSE/container1
Operation: RETRIEVE
primitiveType = REQUEST
RequestID: 001
responseType = blocking
Content: currentNrOfInstances, currentByteSize, latest
```

발신자에서 oneM2M 요청 프리미티브가 HTTP와 바인딩 되는 경우이므로, 요청 프리미티브는 HTTP 요청 메시지로 표 〈4-12〉와 같이 매핑된다.

**표 〈4-12〉 발신자에서 HTTP 요청 메시지 예시('container1' Retrieve)**

```
GET /MN_CSE/container1 HTTP/1.1
Host: http://m2m.sp1.com
From: IN_CSE/IN_AE
X-M2M-RI: 001
Accept: application/vnd.onem2m-resource-data+xml
Content-Length: 115
〈xml〉
    〈content〉
        〈attributeName〉currentNrOfInstances〈/attributeName〉
        〈attributeName〉currentByteSize〈/attributeName〉
        〈attributeName〉latest〈/attributeName〉
    〈/content〉
〈/xml〉
```

다음으로 중계 엔티티 IN_CSE는 그림 〈4-7〉의 ②단계를 수행한다. 해당 단계에서 중계 엔티티는 기저 네트워크를 통하여 전달받은 HTTP 메시지에 대하여 역바인딩 과정을 수행한 후, 발신자로부터 전송된 oneM2M 요청 프리미티브를 확인 및 처리하는 과정을 수행한다. (모든 엔티티에서 바인딩 및 역바인딩 과정은 동일하게 발생하므로, 이 후 모든 과정에서 바인딩 및 역바인딩 과정은 설명을 생략한다.) ② 단계를 코어 프로토콜 규격서에서 살펴본 수신자 측면에서 포괄적 절차(그림 〈4-4〉) 및 자원 타입에 따른 처리 절차(그림 〈4-5〉)를 통해 살펴보면, 중계 엔티티는 해당 oneM2M 프리미티브의 유효성(Recv-1.0) 및 통신 방법(Recv-2.0)을 확인한다. 이 때, 요청 프리미티브의 통신방법과 관련된 파라미터가 Blocking 모드(responseType = blocking)이므로, 이 후 자원을 처리하는 과정(Recv-6.0)을 수행한다. 또한 중계 엔티티는 목적지를 확인할 수 있는 파라미터(To = MS_CSE/container1)를 통해 해당 엔티티가 최종 목적지가 아닌 것을 판단하고(Recv-6.1), CMDH와 관련된 처리를 해당 엔티티가 지원하는지 확인(Recv-6.9)한다. 본 예제에서 모든 엔티티들은 CMDH를 지원하지 않으므로 해당 oneM2M 프리미티브는 다음 엔티티로 전달(Recv-6.11)된다.

그림 〈4-7〉의 ③단계에서 수신자 MN_CSE는 수신된 oneM2M 프리미티브에 대해 그림 〈4-4〉 및 그림 〈4-5〉의 과정을 상기 ②단계와 동일하게 적용한다. 상기 ②단계와 마찬가지로 MN_CSE에서 통신 방법은 Blocking 모드이므로, 자원을 처리하는 과정(Recv-6.0)이 수행된다. 그러나, ③단계에서는 oneM2M 프리미티브를 처리하는 엔티티가 수신자이므로 해당 엔티티가 최종 목적지 확인(Recv-6.1) 이 후, Recv-6.2부터 Recv-6.6을 수행한다.

수신자는 발신자로부터 전송 받은 oneM2M 요청 프리미티브를 처리한 후, ④단계에서 oneM2M 응답 프리미티브를 생성 및 발신자에게 생성된 oneM2M 응답 프리미티브를 전송한다. 그림 〈4-7〉의 ④단계는 그림 〈4-5〉의 oneM2M 응답 프리미티브 생성(Recv-6.7) 및 생성된 oneM2M 응답 프리미티브 전송(Recv-6.8)에 해당한다. 또한 생성된 oneM2M 응답 프리미티브는 다음 표 〈4-13〉과 같은 파라미터로 구성되며, 해당 oneM2M 응답 프리미티브를 HTTP 바인딩 기술 규격서를 통해 매핑되면 표 〈4-14〉와 같다.

**표 〈4-13〉 수신자에서 oneM2M 응답 프리미티브 예시('container1' Retrieve)**

| |
|---|
| primitiveType = RESPONSE<br>responseCode = Success<br>RequestID: 001<br>currentNrOfInstances = 2<br>currentByteSize = 95<br>latest = MN_CSE/container1/a1bx3 |

표 〈4-14〉 수신자에서 HTTP 응답 메시지 예시('container1' Retrieve)

```
200 OK
X-M2M-RI: 001
Content-Type:application/vnd.onem2m-resource-data+xml
Content-Length: 115
〈xml〉
    〈container〉
        〈currentNrOfInstances〉2〈/currentNrOfInstances〉
        〈currentByteSize〉95〈/currentByteSize〉
        〈latest〉MN_CSE/container1/a1bx3〈/latest〉
    〈/container〉
〈/xml〉
```

그림 〈4-7〉의 ⑤단계를 통해 전달된 oneM2M 응답 프리미티브는 ⑥단계를 통해 발신자 IN_AE에서 처리된다. ⑥단계를 그림 〈4-3〉의 발신자 측면에서 포괄적 절차를 통해 살펴보면, 해당 예제에서 oneM2M 요청 프리미티브가 Block 모드이므로 발신자는 oneM2M 요청 프리미티브를 전송 및 oneM2M 응답 프리미티브를 기다린다. 이 후, ⑥단계에서 발신자는 oneM2M 응답 프리미티브를 처리하고 모든 과정을 종료한다.

마지막으로, 코어 프로토콜 및 HTTP 바인딩 기술 규격서가 적용된 예제를 간략하게 살펴보면 그림 〈4-8〉과 같이 표현될 수 있다.

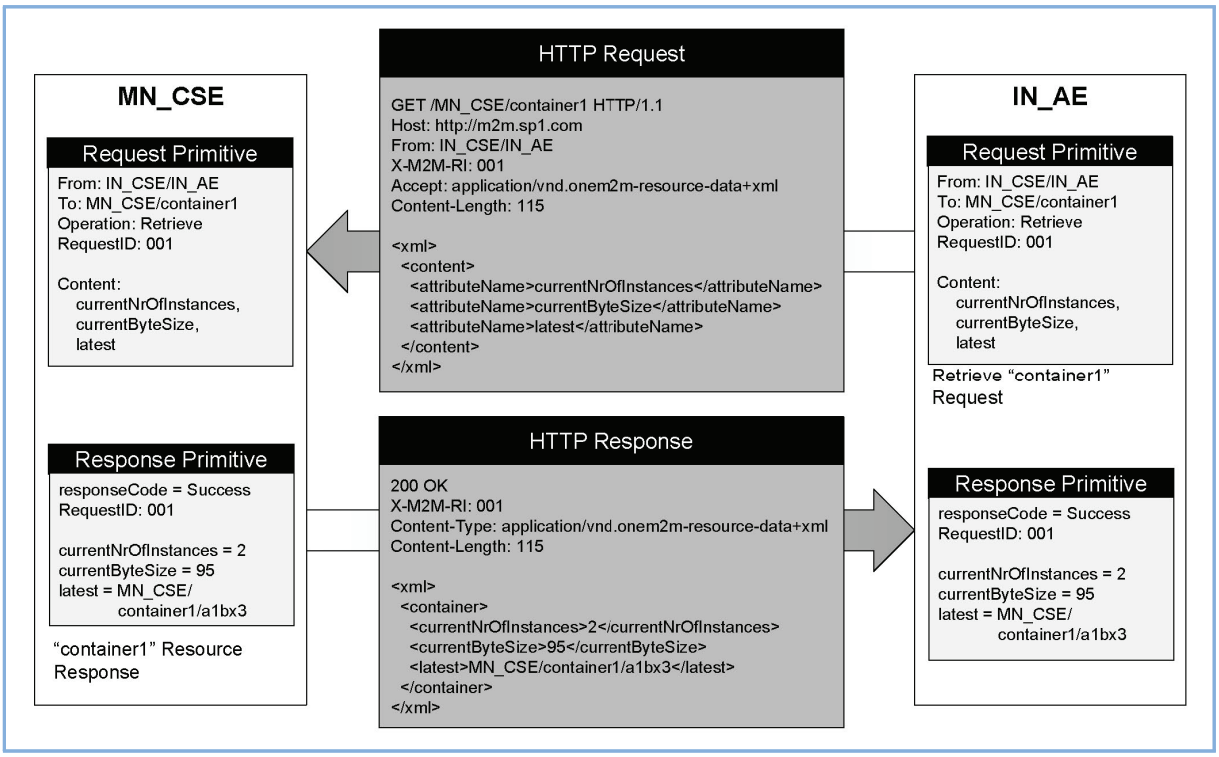



그림 〈4-8〉 oneM2M 프리미티브 및 HTTP 메시지

## 4.6. MQTT 프로토콜 개요

MQTT는 네트워크 지연 및 손실이 심한 전송 환경에서 센서, 검침 기기 등 작은 장치들을 위한 신뢰성 있는 메시지 전달을 가능하게 하기 위해서 IBM에 의해 1999년에 개발된 메시지 프로토콜이다.

MQTT는 Message Queuing Telemetry Transport의 약자로서 원격 장치에 대한 모니터링을 목적으로 데이터 수집을 용이하게 하기 위해서 개발되었다. MQTT 프로토콜의 동작 시에는 아래 그림 〈4-9〉와 같이 MQTT 브로커 또는 MQTT 서버라고 불리는 중계서버를 기반으로 MQTT 클라이언트 기기들과 Publish/Subscribe 관계를 맺고, 이를 바탕으로 경량의 메시지를 교환하는 방식을 취하고 있다.

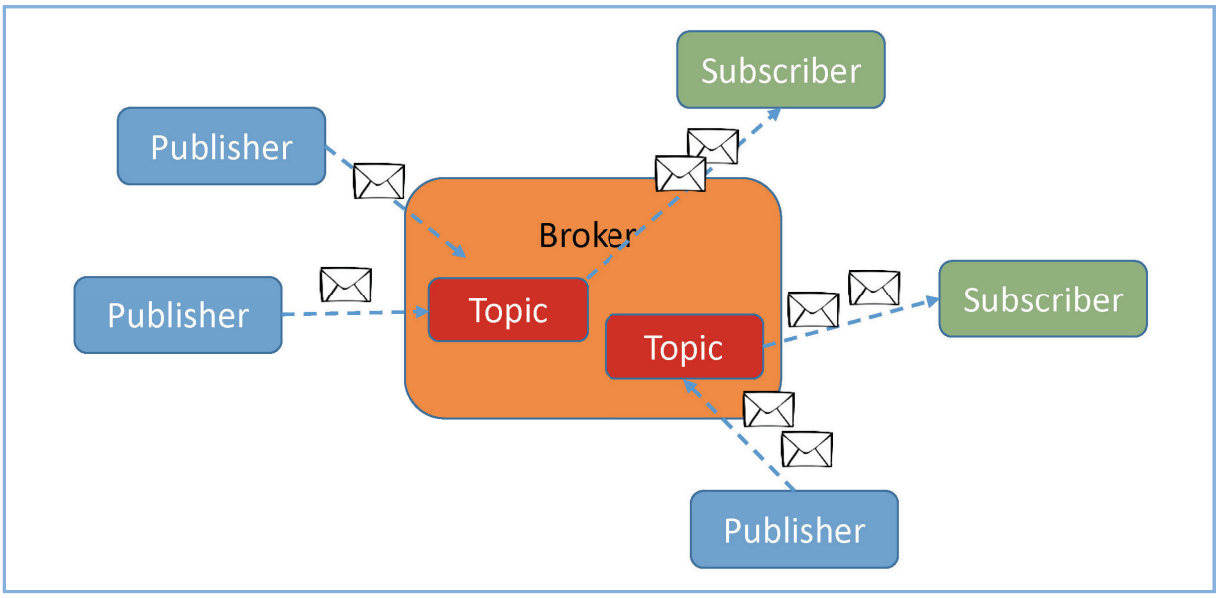



그림 〈4-9〉 MQTT Publish-Subscribe 메시지 교환 방식

### 4.6.1. MQTT 토픽

MQTT에서 다수의 클라이언트들은 MQTT 서버에 연결되어 토픽 이름에 Subscribe하며 클라이언트들은 토픽 이름을 기반으로 보내려는 메시지를 Publish 함으로써 해당 메시지는 MQTT 서버를 거쳐 Publish 된 토픽 이름에 가입한 클라이언트들에게 전달된다. 이때, 다수의 MQTT 클라이언트들이 같은 토픽 이름에 Subscribe할 수 있기 때문에 데이터를 보내려는 임의의 한 MQTT 클라이언트가 한 개의 메시지를 해당 토픽으로 Publish하게되면 이 메시지는 다수의 클라이언트들에게 전달이 될 수 있다.

이와 같이MQTT 메시지는 그것이 Publish 될 때 토픽 이름을 가지고, 이를 통해서 해당 토픽에 가입한 클라이언트들에게 전달될 수 있는 구조이다. 따라서 토픽이 데

이터 전달을 위한 중요한 연결 매개체가 된다. 토픽의 이름은 트리 구조의 계층적인 이름으로 구성될 수 있는 형태를 가지며, 각 계층은 "/"(Slash) 구분 기호로 구별된다. 토픽의 이름은 클라이언트가 MQTT 서버에 접속할 때 Subscribe 하고자 하는 토픽의 이름을 전송 함으로서 MQTT 서버에 해당 정보가 생성된다. 또한 토픽의 이름 컴포넌트로서 "#"(multi-level), "+"(single-level) 와일드카드 캐릭터를 지원하기 때문에, 클라이언트는 한번에 다수의 토픽에 Subscribe 할 수 있고, 다수의 토픽에 Publish 가능하다. 해당 토픽 이름의 예는 다음과 같다.

- ✓ sport/tennis/player1
- ✓ sport/tennis/player1/ranking
- ✓ sport/tennis/# (a multi-level wildcard)
- ✓ sport/tennis/+ (a single-level wildcard)

### 4.6.2. MQTT 메시지 전달 신뢰성

MQTT는 저전력으로 동작되며 데이터 전송 지연 및 손실이 발생하는 네트워크 환경을 고려하고 있다. 따라서 메시지 전달의 신뢰성을 위한 세 가지 QoS레벨을 정의하고 있다. 여기서 QoS는 MQTT 클라이언트에서 Publish 되어 전달된 메시지가 다른 MQTT 클라이언트에게 어떻게 전달될 것인가를 정의하는 것이다.

예를 들어, QoS 레벨0의 경우는 전달되는 메시지에 대한 신뢰성을 보장하지 않는 경우로서 클라이언트는 메시지를 Publish하고 바로 전달이 되었을 것이라고 가정하고 메시지를 삭제한다. 클라이언트와 서버 사이에 메시지 전달에 대한 어떠한 Acknowledgement를 지원하지 않으며 이 경우는 메시지의 유실 가능성이 발생할 수 있다. QoS 레벨1의 경우는 전달되는 메시지에 대한 신뢰성을 보장하는 방식이다. 클라이언트는 MQTT 서버로 메시지를 Publish하고 서버로부터 Acknowledgement를 받기 전까지 메시지를 보관한다. 이는 전송과정 중에 메시지가 손실되는 경우 재전송을 할 수 있게 하기 위함이다. 해당 방식은 메시지가 최소 1번 전달되는 방식으로 중복 전달 가능성은 있다. 마지막으로 QoS 레벨 2는 전달 메시지에 대한 신뢰성을 보장하며 정확하게 메시지가 한번만 상대에게 전달될 수 있는 경우를 보장하는 방식이다. 이를 위해서 두 단계의 Acknowledgement 방식을 지원한다. 물론 QoS 레벨이 높을수록 메시지 전달에 대한 엄밀성, 신뢰성이 증가하지만 데이터 전달 지연은 증가한다.

클라이언트A에서 서버를 거쳐서 클라이언트B로 데이터가 전달되는 상황에서 Publish 및 Subscribe 경우에 대한QoS 레벨에 따라서 데이터 전달이 되는 상황을 살펴보면 다음과 같다. 첫 번째 예로서, 클라이언트 A가 QoS 레벨 2로 메시지를 토픽의 이름으로 Publish하고 클라이언트 B가 QoS 레벨 0로 해당 토픽 이름에 Subscribe하였

다면 MQTT서버에서 클라이언트 B로 메시지를 전달할 때는 QoS 레벨 0로 전달한다. 두 번째 예로서, 클라이언트 B가 QoS 레벨 2로 토픽 이름에 Subscribe 한 상황에서 클라이언트 A가 QoS 레벨 0로 Publish 하였을 경우는 서버에서 클라이언트 B로는 QoS 2가 아닌 QoS 0으로 데이터가 전달된다. 즉, MQTT서버로부터 MQTT 클라이언트 방향으로의 데이터 전달은 클라이언트에서 서버로 Publish 한 QoS 레벨과 동일하거나 작다.

### 4.6.3. MQTT Retained 메시지

MQTT는 Retained 메시지 방식을 지원한다. 해당 방식은 MQTT 서버가 MQTT 클라이언트로부터 Publish된 메시지를 해당 토픽에 가입한 클라이언트들에게 전달한 후 메시지를 서버에 보유하는 것이다. 이 방식을 통해서 메시지가 Publish 되어진 토픽 이름에 새로이 Subscribe 한 MQTT클라이언트들이 서버에 저장된 메시지를 전달 받을 수 있게 된다. Retained 메시지 방식의 유용성은 다음과 같다. 예를 들면 해당 토픽 이름으로 전달되는 데이터의 주기가 긴 경우에, 해당 토픽에 새로이 Subscribe 한 클라이언트들은 가입 후 전달되는 데이터 즉, 업데이트 되는 데이터를 받기까지 시간이 한참 걸릴 수 있지만, Retained 메시지 방식을 통해서 Subscribe와 동시에 즉시 데이터의 업데이트를 받을 수 있다는 장점이 있다.

### 4.6.4. oneM2M Primitive와 MQTT 바인딩 개요

oneM2M의 AE와 CSE간의 또는 CSE 간의 전달되는 oneM2M 메시지를 프리미티브라고 하고 해당 oneM2M 프리미티브들은 실제 전달될 때, 프로토콜로 바인딩 되어 네트워크 계층을 통해서 전달되는 구조를 갖는다. 본 장에서는 두 개의 엔티티 사이에 (AE 또는 CSE) 신뢰성 있는 양방향 메시지 전달을 지원하는 MQTT 프로토콜을 사용한 oneM2M 프리미티브와의 바인딩을 살펴본다.

oneM2M 아키텍쳐 구조에서 정의한 Mca (AE-CSE간의 참조포인트), Mcc (CSE-CSE 간의 참조포인트) 참조 포인트를 통해서 oneM2M 프리미티브가 전달이 되고, 이는 MQTT 프로토콜 바인딩을 통해서 실제 MQTT 프로토콜을 통해서 네트워크 계층을 거쳐서 데이터가 전달된다. 실제 AE 또는 CSE가 MQTT 프로토콜과 바인딩 되기 위해서는 해당 oneM2M 엔티티들은 MQTT 클라이언트 라이브러리와 연동되어야 하며, oneM2M에서 Mca, Mcc참조 포인트에서의 데이터 전달은 실제로는 클라이언트로부터 MQTT 서버를 경유하여 상대 클라이언트로 전달하는 구조를 갖는다.

이와 관련하여 구조적으로 MQTT 바인딩 시나리오로서 oneM2M 엔티티와 MQTT 서버가 직접 연동되어 존재하는 방식과 oneM2M 엔티티와 MQTT 서버가 분리되어 독립적으로 존재하는 방식으로 구분할 수 있으며 해당 시나리오를 살펴보기로 한다.

### 4.6.5. MQTT 바인딩 시나리오 (구조적 관점)

✓ oneM2M 엔티티와 MQTT 서버가 직접 연동되는 방식

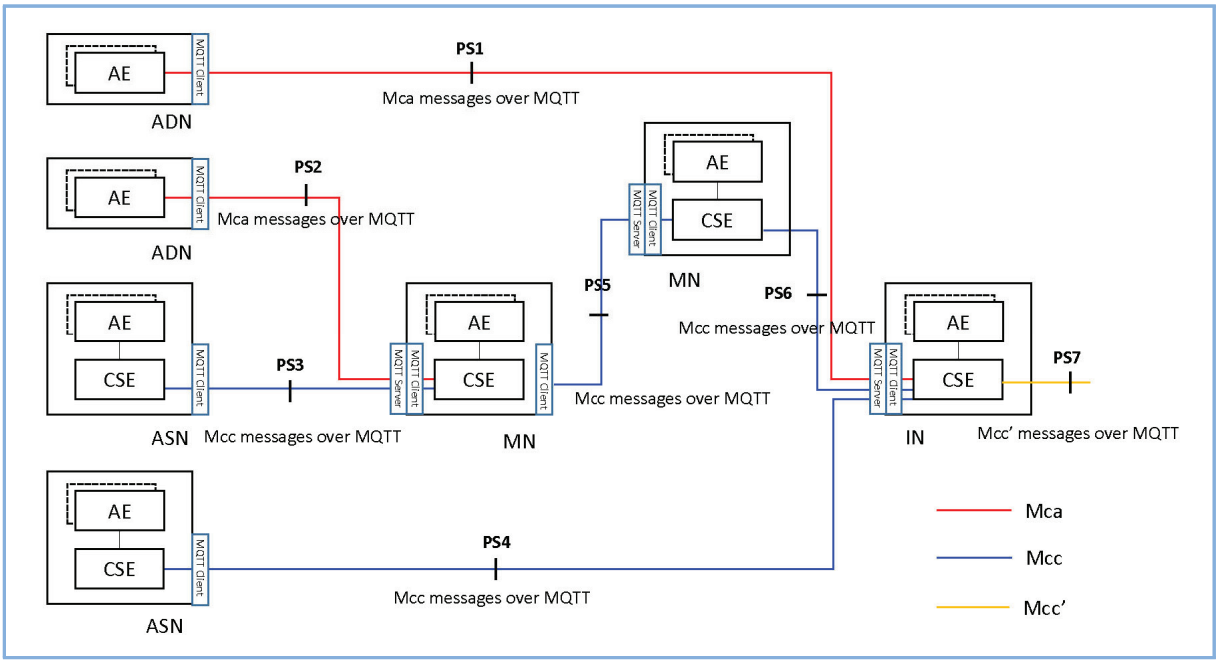



그림 〈4-10〉 MQTT Server Co-located 방식

이 방식은 그림 〈4-10〉와 같이 oneM2M 엔티티에 MQTT클라이언트 및 서버가 직접적으로 연동되어 같이 존재하는 방식으로 MQTT 의 메시지 교환이 각 노드 들간 이뤄진다. 그림과 같은 oneM2M 노드 구성에서 모든 oneM2M 노드들 즉 ADN (Application Dedicated Node), ASN(Application Service Node), MN(Middle Node), IN(Infrastructure Node) 들은 MQTT 클라이언트를 가지고 있으며 MQTT 서버는 MN, IN 에서 노드에 포함되어 제공되는 구조를 갖는다.

해당 oneM2M 엔티티와 MQTT 서버가 직접 연동되는 방식의 시나리오에서 각각의 참조포인트를 통해 전달되는 프로토콜 세그먼트에서 지원하는 oneM2M 참조포인트 및 MQTT 아키텍쳐 구조에서 상호작용 관계는 표 〈4-15〉와 같다.

✓ MQTT 서버가 oneM2M 엔티티로부터 독립적으로 위치하는 방식

이 방식은 그림 〈4-11〉과 같이 MQTT 서버가 oneM2M 엔티티와 분리되어 독립적으로 위치하는 방식으로 MQTT 의 메시지 교환이 각 노드 들간 이뤄진다. 그림과 같은 oneM2M 노드 구성에서 모든 oneM2M 노드들 즉 ADN, ASN, MN, IN 은 MQTT 클라이언트를 포함하고 있으며 그리고 MQTT 서버는 oneM2M 노드와 독립적으로 노드 외부에 존재한다.

해당 MQTT 서버가 oneM2M 엔티티와 독립적으로 위치하는 시나리오에서 각각의 참조포인트를 통해 전달되는 프로토콜 세그먼트에서 지원하는 oneM2M 참조포인트 및 MQTT 아키텍쳐 구조에서 보이는 상호작용 관계는 표 〈4-16〉과 같다.

표 〈4-15〉 프로토콜 세그먼트에 대한 oneM2M - MQTT 구조 관계 (직접연동케이스)

| Protocol Segment | oneM2M Message Transported | MQTT Interaction |
|---|---|---|
| PS1 | Mca (AE of ADN to CSE of IN) | Client in ADN to Server |
| PS2 | Mca (AE of ADN to CSE of MN) | Client in ADN to Server |
| PS3 | Mcc (CSE of ASN to CSE of MN) | Client in ASN to Server |
| PS4 | Mcc (CSE of ASN to CSE of IN) | Client in ASN to Server |
| PS5 | Mcc (CSE of MN to CSE of MN)<br>Mcc (CSE of MN to CSE of ASN)<br>Mca (CSE of MN to AE of ADN) | Client in MN to Server |
| PS6 | Mcc (CSE of MN to CSE of MN) | Client in MN to Server |
| PS7 | Mcc (CSE of IN to CSE of MN)<br>Mcc (CSE of IN to CSE of ASN)<br>Mca (CSE of IN to AE of ADN) | Client in IN to Server |
| PS8 | Mcc' (CSE of IN to CSE of IN) | Client in IN to Server |

표 〈4-16〉 프로토콜 세그먼트에 대한 oneM2M - MQTT 구조 관계 (독립케이스)

| Protocol Segment | oneM2M Message Transported | MQTT Interaction |
|---|---|---|
| PS1 | Mca (AE of ADN to CSE of IN) | Client in ADN to Server in IN |
| PS2 | Mca (AE of ADN to CSE of MN) | Client in ADN to Server in MN |
| PS3 | Mcc (CSE of ASN to CSE of MN) | Client in ASN to Server in MN |
| PS4 | Mcc (CSE of ASN to CSE of IN) | Client in ASN to Server in IN |
| PS5 | Mcc (CSE of MN to CSE of MN) | Client in MN to Server in MN |
| PS6 | Mcc (CSE of MN to CSE of IN) | Client in MN to Server in IN |
| PS7 | Mcc' (CSE of IN to CSE of IN) | Client in IN to Server in IN |

## 4.7. MQTT 프로토콜 바인딩

oneM2M 메시지 프리미티브를 MQTT 프로토콜 바인딩을 통해서 전달하고 자 할 때, oneM2M 메시지 프리미티브를 MQTT 메시지에 어떻게 매핑 할 것인지, MQTT 프로토콜의 어떤 옵션을 이용할 것인지에 대한 것들이 결정 되어야 한다. 본 장에서는 다음에 사항에 대해서 다룬다.

- ✓ oneM2M 엔티티 즉, CSE 또는 AE가 MQTT에 접속하는 방법
- ✓ 요청자가 요청메시지를 MQTT메시지로 구성하고 수신자에게 전달하는 방법
- ✓ 수신자가 요청메시지를 어떻게 수신하고 이에 대한 응답을 어떻게 할 것인지에 대한 방법
- ✓ Mca, Mcc 참조 포인트를 통한 oneM2M 프리미티브 전달을 MQTT 메시지에 어떻게 매핑할 것인가에 대한 방법

### 4.7.1. MQTT 접속

MQTT 클라이언트들이 서로 통신을 하기 위해서 기본적으로 브로커라고 불리는 MQTT 서버에 접속하여야 한다. 앞서 살펴본 것과 같이 oneM2M 엔티티와 MQTT 서버의 위치관계에서, MQTT 서버는 엔티티와 직접 연동되는 구조를 갖을 수도 있고, MQTT 서버가 독립적으로 위치하는 구조를 갖을 수도 있다.

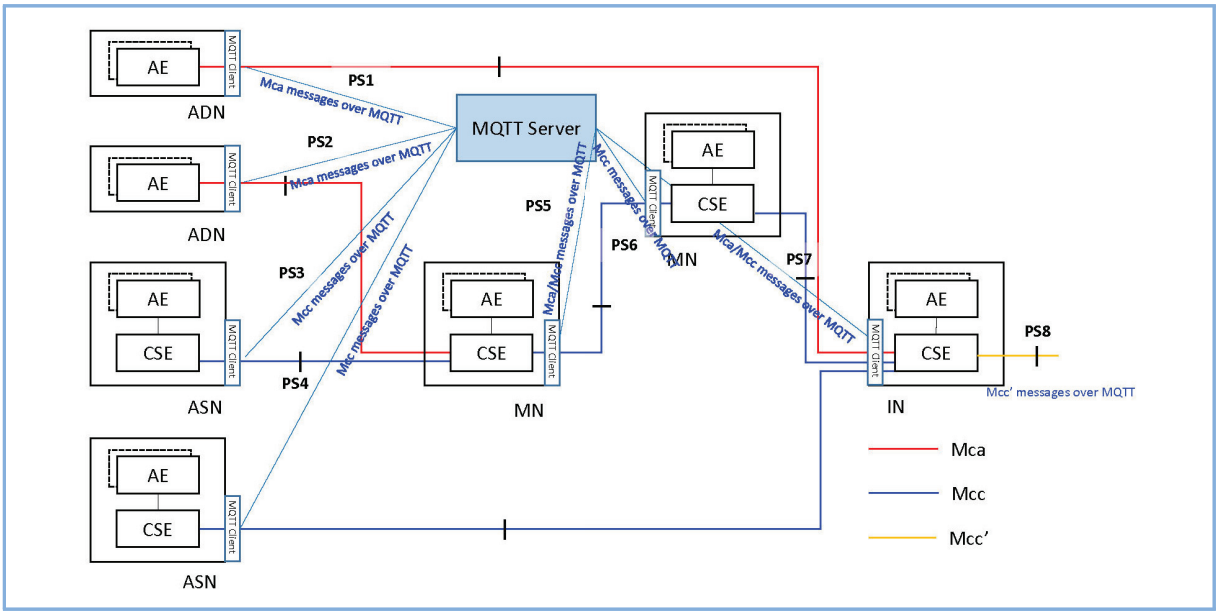



그림 〈4-11〉 MQTT 서버가 독립적으로 위치하는 방식

MQTT 클라이언트는 MQTT 서버의 주소를 획득하고 MQTT CONNECT 메시지로서 서버에 접속을 시도한다. CONNECT 메시지는 ClientId를 포함하는데 해당 ClientId는 MQTT 서버에 접속하는 모든 클라이언트들 사이에 구별될 수 있어야 한다. 이러한 조건은 MQTT 클라이언트가 연동되는 AE의 ID (AE-ID)와 CSE 의 ID (CSE-ID)를 ClientId로 사용함으로써 만족될 수 있다. 또한 MQTT 클라이언트는 접속을 위해서 CONNECT 메시지를 전달할 때, 해당 메시지에 "Clean Session" 플래그를 False로 설정하도록 정하고 있다.

MQTT에서 클라이언트간의 통신은 MQTT서버를 경유하게 되며 클라이언트들간의 직접적인 통신 연결은 존재하지 않는다. 또한 클라이언트가 MQTT서버와 접속을 유지하지 않겠다면 다음과 같은 절차를 통해서 연결을 종료한다. 일단 서버와의 연결을 종료한다. 그리고 "Clean Session" 플래그를 True로 설정하고 서버와 재 연결을 한다. 그리고 다시 연결을 종료한다. 이러한 단계를 통해서 MQTT서버에 클라이언트와 관계된 상태 설정을 남기지 않고 종료를 시키토록 정하고 있다.

### 4.7.2. oneM2M Request/Response 메시지 바인딩

✓ 요청/응답 메시지

MQTT 패킷 구조는 그림 〈4-12〉와 같이 고정 헤더부분, 가변 헤더부분, 페이로드 부분으로 나눌 수 있다. MQTT 는 페이로드에 담길 수 있는 데이터에 대해서 특정한 데이터 모델로 한정하고 있지 않기 때문에 자유로이 데이터를 담을 수 있는 특징이 있다. 따라서 oneM2M Mca, Mcc 참조 포인트를 통해서 전달되는 oneM2M 메시지 프리미티브를 XML 또는 JSON 형식으로 직렬화해서 MQTT 페이로드에 담는다.

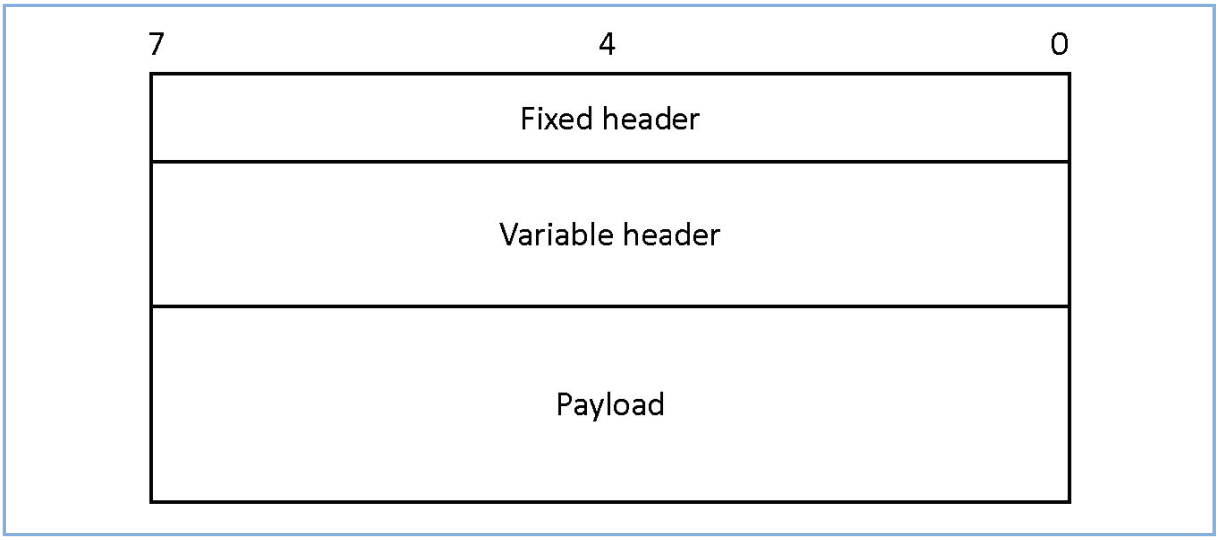



그림 〈4-12〉 MQTT 헤더 구조

또한 MQTT 패킷 구조에서 고정 헤더는 2바이트로 크기를 가지고 패킷 타입, 플래그, 패킷 길이를 담을 수 있는 구조를 가진다. 여기에서 패킷 타입 필드는 14개의 MQTT 컨트롤 패킷 타입을 정의하기 위해서 사용되며 4비트를 가지고 16개의 표현이 가능하다. oneM2M의 Reqeust/Response 메시지는 MQTT 프로토콜 바인딩을 통해서 네트워크 레이어로 전송될 때, MQTT 패킷 타입에서 Publish로서 표현되어진다. 즉 표 〈4-17〉에서 보이듯이 Publish에 대한 값 3이 패킷 타입에 쓰여진다고 이해할 수 있다.

표 〈4-17〉 MQTT Packet Type 필드 값

| | | |
|---|---|---|
| Reserved | 0 | Reserved for future use |
| CONNECT | 1 | Client request to connect to server |
| CONNACK | 2 | Connect acknowledgement |
| PUBLISH | 3 | Publish message |
| PUBACK | 4 | Publish message acknowledgement |
| PUBREC | 5 | Publish received (QoS=2) |
| PUBREL | 6 | Publish release (QoS=2) |
| PUBCOMP | 7 | Publish complete (QoS=2) |
| SUBSCRIBE | 8 | Client subscribe request |
| SUBACK | 9 | Subscribe acknowledgement |
| UNSUBSCRIBE | 10 | Client unsubscribe request |
| UNSUBACK | 11 | Unsubscribe acknowledgement |
| PINGREQ | 12 | Ping request |
| PINGRESP | 13 | Ping response |
| DISCONNECT | 14 | Client disconnection request |
| Reserved | 15 | Reserved for future use |

그림 〈4-13〉에서 보듯이 Publish 메시지에 대한 플래그는 4비트로 표현되며, 여기서 2비트의 값으로 QoS 레벨이 정의된다. oneM2M 프리미티브의 MQTT 바인딩 시에 전송에 대한 신뢰성 및 지연성을 고려하여 QoS 레벨 1로 설정하도록 하는 것을 고려하고 있다.

✓ oneM2M Request메시지 보내기

oneM2M Request메시지는 MQTT의 Publish 메시지를 통하여 전송 되며 Publish 메시지의 가변 헤더에는 토픽의 이름이 들어간다. oneM2M 메시지 프리미티브의 MQTT 바인딩 시에, oneM2M Request 메시지에 대한 MQTT 토픽 이름은 다음과 같이 설정한다.

/oneM2M/req/〈to〉

- "oneM2M"은 oneM2M을 위한 메시지임을 확인하는 용도
- "req"는 Request메시지임을 나타내는 용도
- "<to>"는 타겟 리소스에 대한 URI

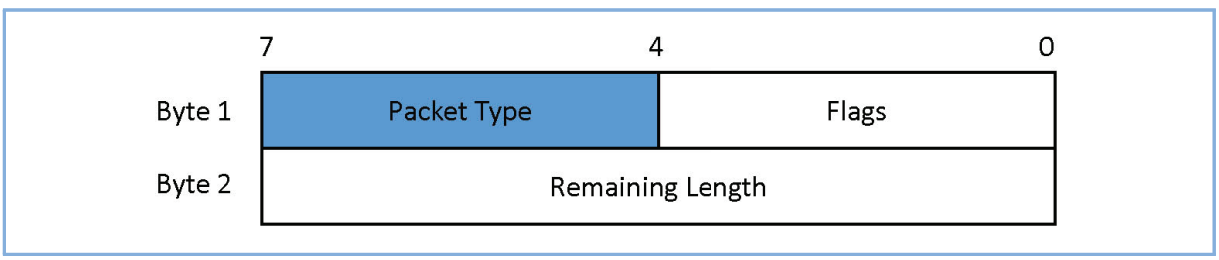



그림 〈4-13〉 MQTT 고정 헤더 구조

✓ oneM2M Request 메시지의 수신 및 응답과정

수신 클라이언트는 oneM2M Request 메시지에 대한 MQTT 토픽 이름으로 subscribe 함으로써 MQTT를 통해서 전달되는 oneM2M Request 메시지를 받을 수 있다. oneM2M Request 메시지는 타겟 리소스에 대한 CRUD 오퍼레이션을 담고 있는데, CREATE 오퍼레이션을 제외한 RETRIVE, UPDATE, DELETE 오퍼레이션 관련해서는 해당 타겟 리소스가 존재하기 때문에 관련한 이름으로 해당 토픽에 Subscribe 되어 있으면 MQTT를 통해서 전달되는 oneM2M Request 메시지를 받을 수 있다. 그러나 CREATE 오퍼레이션에 대한 타겟 리소스는 수신 MQTT클라이언트에서 해당 토픽 이름을 미리 알고 Subscribe 할 수 없으므로 토픽 이름으로 멀티레벨 와일드 캐릭터를 (예를 들어 "/oneM2M/req/#") 사용하여 해당 이름으로 Subscribe 함으로써 해당 oneM2M Request 메시지를 받을 수 있다.

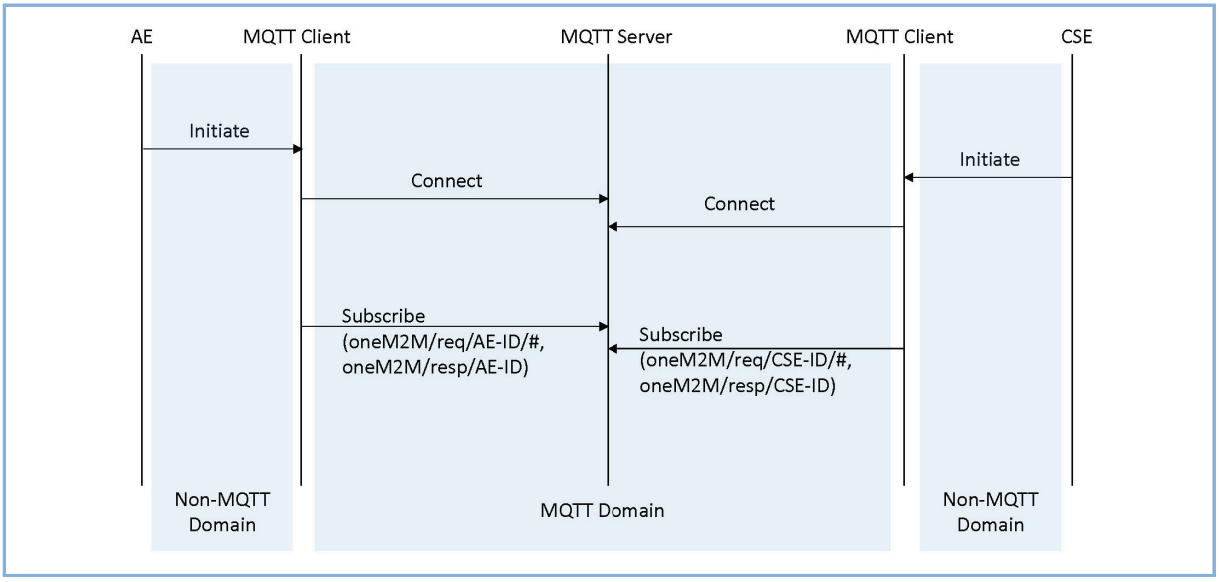



그림 〈4-14〉 MQTT 바인딩을 위한 Subscribe 과정

MQTT 수신 클라이언트는 oneM2M Request메시지를 받으면 해당 메시지에 포함된 메시지에 대한 유효성을 알 수 있는 종료 타임스탬프를 체크를 하고 해당 시간이 지나면 폐기를 한다. 그렇지 않다면, 즉 유효하다면 MQTT PUBLISH 메시지 페이로드에 포함된 oneM2M Request 메시지를 확인 후 관련된 처리를 진행한다. 그리고 oneM2M Response 메시지를 아래의 MQTT 토픽 이름으로 Publish 한다.

/oneM2M/resp/〈to〉

- "oneM2M"은 oneM2M에서 사용되는 메시지임을 구분하기 위한 용도
- "resp"는 응답메시지임을 나타내는 용도
- "<to>"는 요청자에 대한 URI (즉, CSE-ID, AE-ID)

단, oneM2M Reqeust 메시지를 보낸 MQTT 클라이언트가 해당 oneM2M Response 메시지를 받기 위해서 해당 MQTT 토픽 이름에 Subscribe되어 있어야 하며, MQTT Publish 메시지의 페이로드에는 oneM2M Response 메시지가 담기게 된다.

✓ Request/Response 메시지 흐름

MQTT 클라이언트들은 MQTT 서버에 토픽 이름으로 Subscribe 함으로써, 향후 해당 토픽 이름으로 Publish 되는 메시지를 받을 수 있다. 따라서 oneM2M Request/Response메시지에 대한 MQTT 바인딩을 통한 메시지 전달이 가능하기 위해서는 그림 〈4-14〉와 같이 oneM2M 엔티티인 AE 또는 CSE는 MQTT 바인딩 절차를 위해서 AE-ID 또는 CSE-ID 정보를 MQTT 클라이언트에게 전달한다. 그리고 해당 MQTT 클라이언트는 MQTT 서버에 접속을 하며 MQTT CONNECT 메시지에 ClientId로써 AE-ID, CSE-ID를 사용한다.

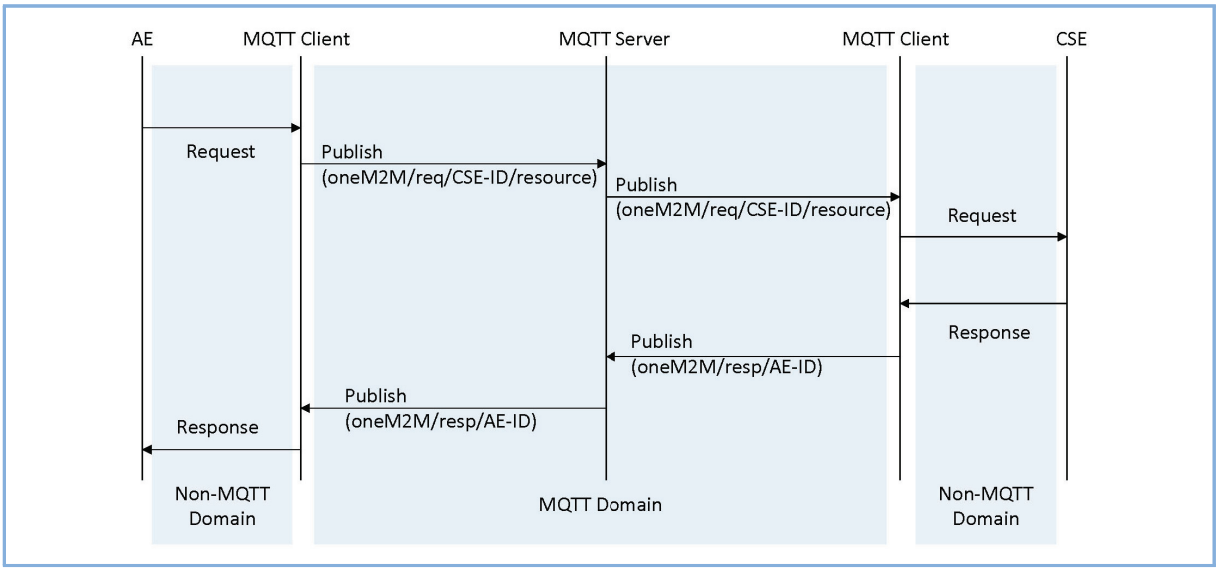



그림 〈4-15〉 MQTT 바인딩을 통한 Req/Resp 메시지 전달 과정

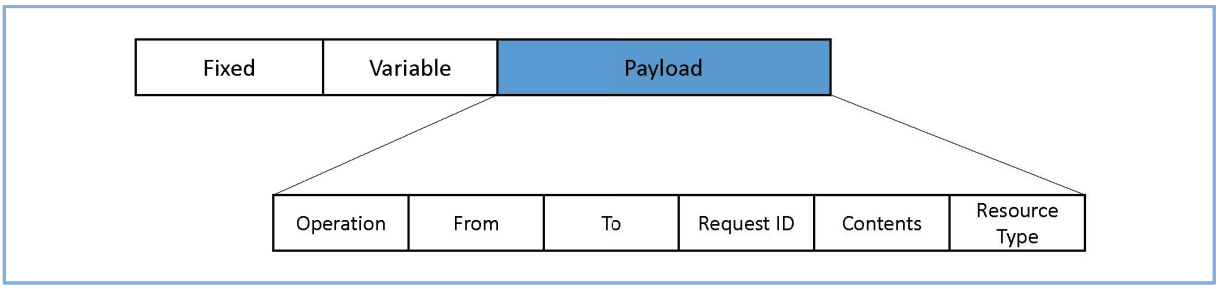



그림 〈4-16〉 MQTT 바인딩을 통한 Request 메시지 예

해당 MQTT 클라이언트가 성공적으로 MQTT 서버와 연결되면 각 클라이언트는 서버에 Subscribe를 하는데 해당 토픽 이름으로 AE-ID 또는 CSE-ID 뒤에 "#" 와일드 캐릭터를 사용하여 이름을 구성하고 해당 토픽 이름에 Subscribe를 수행한다.

즉, "oneM2M/req/ID/#"의 경우, "oneM2M/req/ID"와 관련된 토픽에 Subscribe 한다는 것을 의미한다. 이러한 설정을 통해서 MQTT 클라이언트는 해당 AE-ID, CSE-ID를 상위 컴포넌트로 가지고 하위에 어떠한 리소스 이름의 컴포넌트에 대해서도 대응되는 MQTT 메시지를 받을 수 있는 환경이 만들어 지게 된다.

와일드 캐릭터 "#"을 사용한 토픽 이름에 Subscribe를 함으로써, oneM2M/req/ID/resource 와 같은 이름으로 Publish 되어진 메시지에 대해서도 타겟 리소스에 대한 이름에 직접적으로 Subscribe 없이 상위 컴포넌트 매칭 과정을 통해서 해당 메시지를 MQTT 클라이언트는 수신할 수 있다.

✓ oneM2M 메시지와 MQTT 메시지 Mapping

### 1. oneM2M Request 메시지와 MQTT 메시지 Mapping

MQTT 메시지 구성은 그림 〈4-16〉 과 같이 고정 헤더, 가변 헤더, 페이로드로 구성되며, oneM2M Request 메시지는 MQTT 페이로드 부분에 담길 수 있다.

예로서 AE와 CSE사이에 Mca 레퍼런스 포인트를 통한 그림 〈4-15〉의 oneM2M Request/Response 메시지의 흐름에서 AE의 MQTT 클라이언트 에서는 MQTT 서버로 전달하고자 하는 CSE-ID가 포함된 "oneM2M/req/CSE-ID"또는 "oneM2M/req/CSE-ID/resource 토픽 이름으로 MQTT 메시지를 Publish를 한다. 또한 해당 MQTT Publish 메시지의 페이로드에는 다음의 정보가 포함된다: {"Operation (op)", "From (fr)", "To (to)", "Request Identifer (ri)", "Content (cn)", "Resource Type (ty)"} 및 다른 옵션 파라미터 MQTT 서버의 경우 MQTT Publish 메시지를 받으면 해당 토픽이름으로 가입한 MQTT 클라이언트에게 MQTT 메시지를 전달하며, MQTT 클라이언트는 전달받은 메시지를 해당CSE 또는 AE로 전달하게 된다.

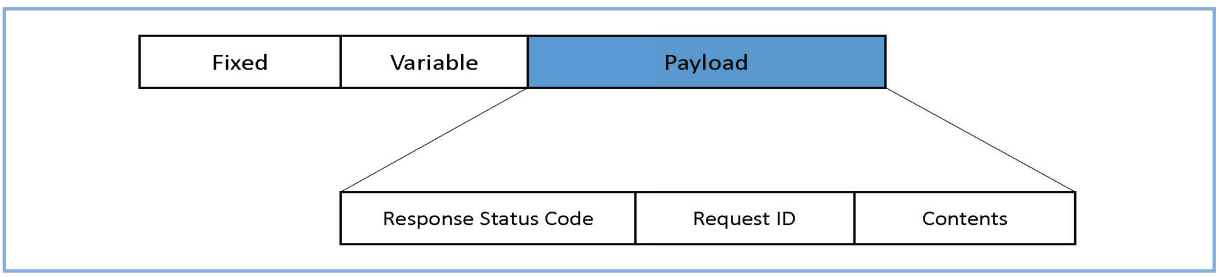



그림 〈4-17〉 MQTT 바인딩을 통한 Response 메시지 예

### 2 oneM2M Response 메시지와 MQTT 메시지 Mapping

oneM2M Request메시지와 동일하게 oneM2M Response메시지는 MQTT 페이로드 부분에 담기게 된다. 해당 페이로드에는 oneM2M Request에 대한 성공, 실패를 알려주는 응답코드, 요청ID, 그리고 추가 정보 등이 담기며 oneM2M Response메시지가 MQTT 프로토콜을 통해서 전달되어지기 위해서 응답을 위한 MQTT 메시지는 oneM2M Request 메시지의 "fr" 필드를 참조하여 CSE-ID 또는 AE-ID를 토픽 이름으로 MQTT 메시지를 Publish한다. 해당 예에서는 "oneM2M/resp/AE-ID" 토픽 이름에 MQTT 메시지를 Publish하여 응답 메시지를 전송한다. 또한 MQTT Publish 메시지의 페이로드에 포함되는 정보는 그림 〈4-17〉과 같이 {"Response Status Code (rsc)", "Request Identifier (ri)", "Content (cn)"} 와 다른 옵션 파라미터들을 포함한다. AE로부터 CSE로 메시지가 전달되는 절차와 동일하게 CSE로부터 AE로의 응답 메시지가 전달되는 절차도 MQTT 서버를 경유하여 메시지가 전달된다.

## 4.8. CoAP 개요

사물인터넷 환경에서 연결되는 디바이스들은 스마트폰, PC와 같은 하드웨어 성능이 좋은 디바이스도 존재하지만, 센서와 액츄에이터와 같은 저 용량의 메모리와 낮은 연산처리 성능을 가진 디바이스까지 다양하게 존재한다. CoAP은 Constrained Application Protocol 의 약자로서 전송지연과 패킷손실률이 높은 네트워크 환경에서 저 사양의 하드웨어로 동작되는 센서 디바이스의 RESTful 웹서비스를 지원하기 위한 경량 프로토콜이다.

CoAP은 인터넷 표준화 단체인 IETF Core WG에서 HTTP 웹 프로토콜에 대응되는 경량형 웹 프로토콜로서 개발되었으며 다음과 같은 특징을 갖는다. CoAP 프로토콜은 HTTP와 같은 URI 를 통해서 리소스에 대한 CRUD 오퍼레이션을 지원하며 하위 프로토콜 스택으로 IPv6을 지원하는 6LowPAN과 트랜스포트 계층으로는 UDP을 고려하고 있다. 또한 CoAP은 작은 메시지 크기 및 쉬운 인코딩, 디코딩을 위해 바이너리 인코딩 방식을 지원한다는 특징을 가지고 있으며 CoAP의 메시지 헤더 크기는 약 10~20바이트이고 HTTP와 비교하여 메시지 크기는 약 1/10 정도이기 때문에 HTTP와 비교하여 메시지 단편화 가능성이 낮아지기 때문에 패킷손실률이 높은 네트워크 환경에서 유리하는 특징을 갖는다.

CoAP에서 메시지 전송 방식은 확인형(CON), 비확인형(NON), 승인(ACK), 리셋(RST)의 네 가지를 지원하며 요청/응답 방식의 메시지 교환을 한다. 또한 메시지 트랜잭션마다 각각 토큰 ID를 두어 각 트랜잭션을 구분한다. 추가적으로 현재 IETF Core WG에서는 큰사이즈의 데이터 전달을 위한 블록단위 전달방식과 CoAP 서버의 리소스 변화를 통보 받을 수 있는 Observe 기능을 표준화하고 있다.

## 4.9. CoAP 바인딩 고려사항

oneM2M 메시지 프리미티브를 CoAP 프로토콜과 바인딩을 할 때, CoAP의 메시지 형식 및 전달 방식을 이해하는 것이 필요하다. 또한 CoAP의 고유한 기능에 대한 고려요소를 파악하여야 한다. 본 장에서는 CoAP의 메시지 형식, 전달 방식, 캐싱 및 블록단위 전송 기능을 살펴본다.

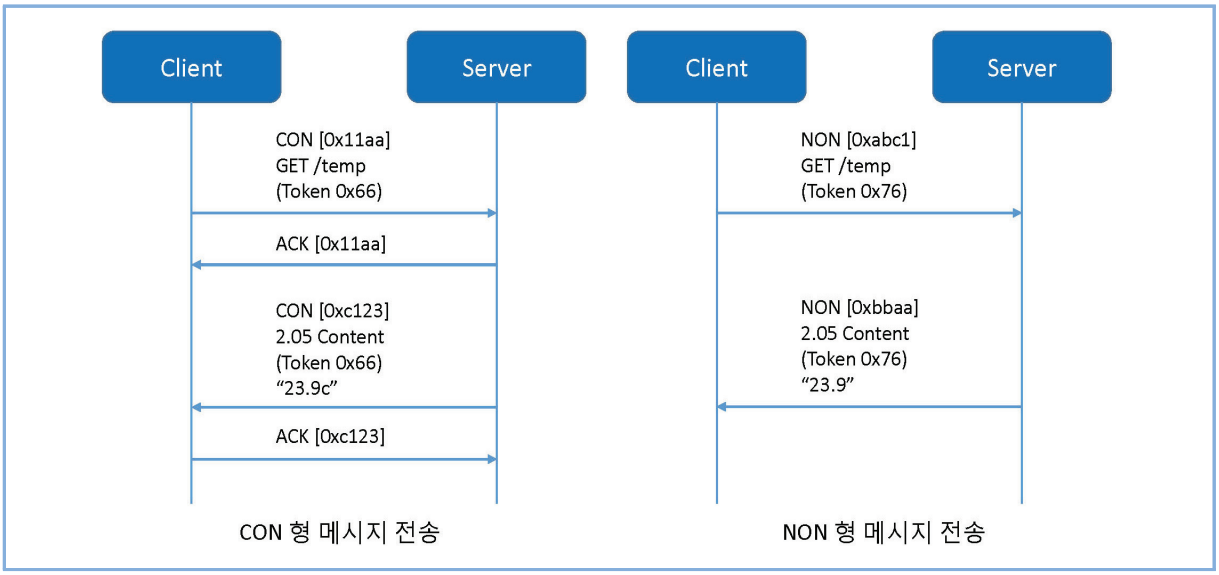



그림 〈4-18〉 CoAP 메시지 타입별 메시지 전송 방식

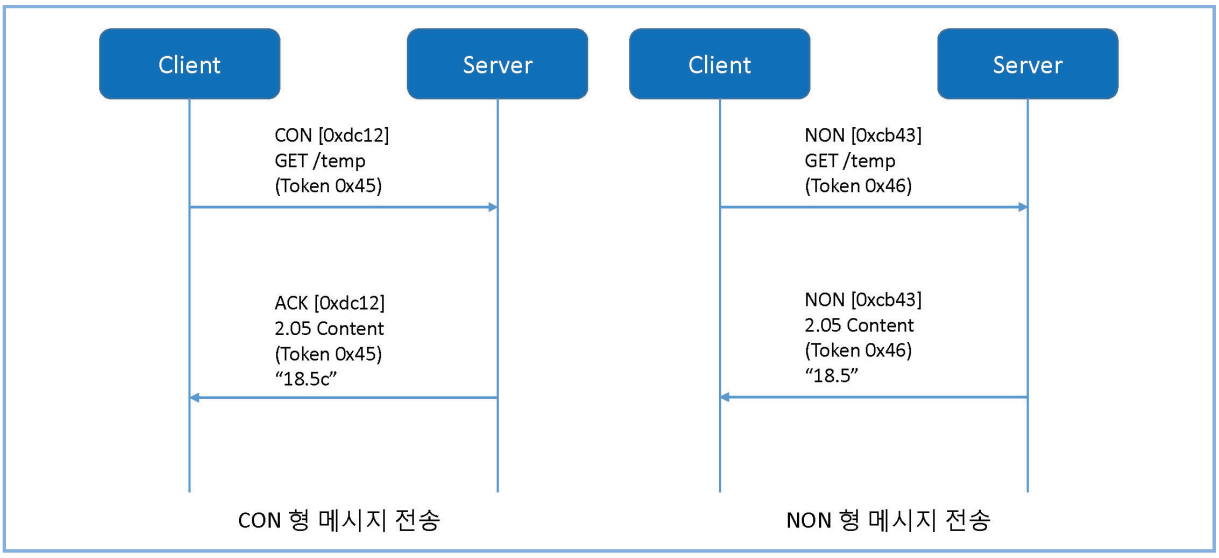



그림 〈4-19〉 CoAP에서 Piggy-back 형태의 응답

### 4.9.1. CoAP 메시지 타입

CoAP 메시지 헤더에는 CoAP 메시지 타입을 정의하는 2비트가 있고 해당 필드에 CON (0), NON (1), ACK (2), RST (3)의 메시지 타입을 위한 0~3의 비트 값을 설정할 수 있다. CoAP은 HTTP와 동일하게 GET, POST, PUT, DELETE 메소드를 지원하며 두 개의 호스트간의 요청/응답 메시지의 데이터 교환이 발생하고 응답 메시지에는 요청메시지에 대한 성공과 실패여부에 대해서 HTTP 호환성을 갖는 응답 코드를 지원한다. 그림 〈4-18〉에서 보듯이 CON형 메시지 전송에서는 신뢰성을 보장하기 위해서 요청메시지에 대한 ACK를 응답으로 받을 수 있으며, NON형 메시지 전송은 비신뢰성 메시지 전송방식으로 ACK가 전달되지 않는다. 또한 CoAP 요청에 따른 응답은 토큰이라는 필드를 통해서 서로 짝을 이루며, 메시지 ID는 하나의 트랜잭션을 구분하는데 사용된다. 이밖에 그림 〈4-19〉와 같이 CoAP에서는 요청 메시지에 대해서 즉시 응답이 가능한 경우에 실제 데이터를 Piggy-back 형태로 응답하는 방식도 지원하고 있다.

### 4.9.2. CoAP 메시지 형식

기본적으로 CoAP 메시지는 OSI 레이어에서 어플리케이션 계층에 속하며 해당 CoAP 메시지는 OSI레이어의 트랜스포트 계층에서 UDP 데이터그램의 페이로드 부분에 들어간다. 그림 〈4-20〉과 같이 CoAP 메시지는 바이너리 포맷으로 인코딩 되는데 4바이트의 고정헤더를 포함하고 있다. 이어서 0~8 바이트 길이의 토큰이 위치하며 이후 옵션이 위치하고 옵션 이후의 메시지 형식은 CoAP의 페이로드가 위치한다.

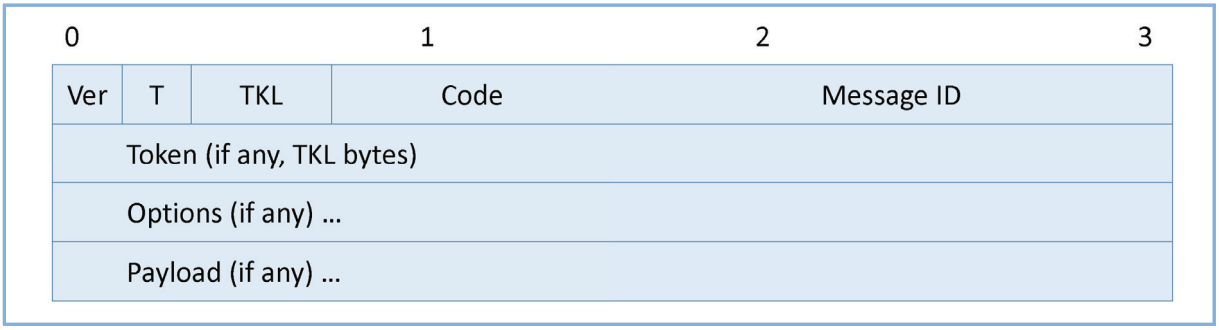



그림 〈4-20〉 CoAP 메시지 형식

- Version (Ver): 2 비트의 값을 갖고, CoAP 버전 넘버를 의미한다. 현재 표준은 01 binary로 설정되어 있다.
- Type (T): 2비트의 값을 갖고, 메시지 타입을 의미한다. 0은 확인형, 1은 비확인형, 2는 승인, 3은 리셋을 나타낸다.
- Token Length (TKL): 4비트의 값을 갖고, 토큰 필드의 길이를 나타낸다. 단위는 바이트이며 0에서 8까지의 값을 사용할 수 있다. 9 에서 15까지의 값은 예약되어 있다.

- Code: 8비트의 값을 갖고, 3비트는 클래스를, 5비트는 자세한 내용을 의미한다. c.dd와 같은 형태로 표현되는데 c는 0~7 값을 갖고 dd는 00에서 31까지의 값을 갖는다. 클래스는 0은 요청, 2는 성공적인 응답, 4는 클라이언트 에러 응답, 5는 서버 에러 응답이다. 요청의 경우, 0.01은 GET, 0.02은 POST, 0.03은 PUT, 0.04는 DELETE를 나타낸다.
- Message ID: 16비트의 값을 갖고, 중복 확인 및 CON 또는 NON메시지에 대한 짝으로서 ACK, RST 메시지에 사용된다.

### 4.9.3. CoAP 캐싱 기능

CoAP은 캐싱 기능을 지원함으로써 요청/응답 메시지가 네트워크를 경유하는 지연 및 대역폭을 감소 시키는 효과를 줄 수 있다. 해당 캐싱 기능은 주로 프록시 서버에서 지원이 되고 프록시 서버를 경유하는 응답 메시지에 대해서 응답코드를 기반으로 캐싱이 가능한 리소스에 대해서 프록시 서버가 해당 리소스를 캐싱한다. 따라서 만약 프록시 서버로 해당 리소스에 대한 요청 메시지가 경유하는 경우에 대해서 프록시 서버에서 캐싱되어있는 리소스라면 origin server에 요청 없이 응답을 줄 수 있다. 캐싱 기능은 freshness와 validity를 사용해서 지원이 된다.

✓ Freshness: 요청 메시지에 대해서 캐싱된 리소스가 "fresh"라면 해당 리소스에 대해서 origin server에 요청하지 않아도 되므로 네트워크 지연을 줄일 수 있다. origin server는 Max-Age CoAP 옵션을 사용하여, 이를 통해서 CoAP 응답 메시지에 포함된 리소스에 대한 expiration time을 명시 한다. 시간이 Max-Age 값을 넘어간다면, 해당 캐싱된 리소스 정보는 유효하지 않음을 나타내는 것이다.

- Max-Age 옵션은 디폴트 값으로 60초로 설정되며, Max-Age 옵션이 없는 응답 메시지의 경우에 자동적으로 디폴트 값으로 Max-Age가 설정된 것으로 인지된다.
- Origin server가 Max-Age 값을 0으로 설정하게 되면 캐싱을 하지 않는다는 것을 의미하는 것이다.
- 프록시 서버는 캐싱하고 있는 리소스를 갱신하기 위해서 origin server에 재요청을 수행할 수 있다.

✓ Validity: CoAP 클라이언트(프록시)는 캐싱된 리소스에 대한 유효성을 확인하고 갱신하기 위하여 origin server에게 리소스 전체에 대한 질의를 하는 것이 아닌 ETag를 사용하여 조건부 질의를 할 수 있다. ETag를 사용하면 origin server에서 해당 리소스가 갱신이 되었는지의 여부를 확인할 수 있다.

- CoAP 클라이언트에서 캐싱된 리소스가 여전히 유효하다면 서버는 Max-Age 옵션과 함께 응답코드 2.03-Valid을 통해서 응답하고 이를 통해서 클라이언트는 해당 리소스에 대한 freshness값을 업데이트 한다.
- CoAP 클라이언트의 캐싱된 응답이 유효하지 않다면 서버는 업데이트된 리소스 정보와 함께 응답코드 2.05-Content를 통해서 응답을 준다. 해당 메시지를 받은 클라이언트는 캐싱된 리소스에 대해서 응답 받은 리소스로 업데이트를 수행한다.

### 4.9.4. 블록단위 전송

CoAP의 기본 메시지는 센서 값, 액츄에이터 상태 값 등 작은 사이즈의 데이터를 다룰 수 있게 되어 있지만, 블록단위 전송 옵션을 통해서 큰 사이즈의 데이터를 전달할 때 블록 단위로 나누어서 전송할 수 있는 유용한 옵션이다. 해당 기능은 oneM2M 바인딩 시에 펌웨어/소프트웨어 업데이트 등 큰 사이즈의 데이터 전송이 필요할 때 유용성이 있다. 블록 단위 전송 기능은 CoAP 메시지 어플리케이션 단에서 큰 사이즈의 페이로드를 fragmentation해서 전송할 수 있도록 지원하는 옵션이다.


그림 〈4-21〉 CoAP 블록단위 전송 GET 예제

블록 옵션 1은 요청 메시지에 대한 블록단위 전송 시 사용되며, 블록 옵션 2는 응답 메시지에 페이로드를 보내는 경우에 사용된다. 그림 〈4-21〉은 블록 옵션 2의 예제인데, 콜론 앞의 숫자를 통해서 블록옵션 1인지 블록옵션 2인지를 나타낸다. 콜론 다음의 숫자는 블록 번호, "/"(Slash), 다음은 M(more) 비트, 두 번째 "/"(Slash) 다음의 숫자는 블록 사이즈를 나타낸다.

## 4.10. oneM2M 메시지 Primitive와 CoAP 바인딩

본 장에서는 Mca, Mcc 참조 포인트를 통한 oneM2M 요청/응답 메시지 전송 시 해당 메시지 플로우를 CoAP의 메시지 플로우에 연동하는 방법과 oneM2M 프리미티브를 CoAP 메시지에 매핑하는 방법에 대해서 살펴본다.

### 4.10.1. oneM2M 요청 메시지를 CoAP 요청 메시지에 Mapping

oneM2M 요청 메시지를 CoAP 요청 메시지로의 매핑 시에 우선, onM2M 메시지 프리미티브의 오퍼레이션에 대해서 HTTP 프로토콜과 마찬가지로 CoAP 메소드로 매핑하는 부분이 필요하다. 앞서 살펴본것과 같이 CoAP의 요청 메시지 형식에는 4바이트 고정 헤더에서 8비트 길이의 코드를 포함하고 있으며 해당 코드 필드를 통해서 요청 메소드를 나타낼 수 있다. 추가적으로 oneM2M 요청 메시지가 갖는 프리미티브 파라미터의 값은 CoAP의 페이로드에 들어간다.

표 〈4-18〉 oneM2M 오퍼레이션과 CoAP 메소드 Mapping

| oneM2M operation | CoAP Method |
|---|---|
| CREATE | POST |
| RETRIEVE | GET |
| UPDATE | PUT |
| DELETE | DELETE |
| NOTIFY | POST |

표 〈4-18〉에서 보듯이 oneM2M CRUD 오퍼레이션이 CoAP 메소드로 매핑될 수 있으며, RETRIEVE, UPDATE, DELTE에 대해서는 매핑되는 CoAP 메소드가 1대1로 매칭된다. 하지만 CREATE, NOTIFY에 대해서는 두 개의 메소드 모두 CoAP의 POST로 매핑 되는데, 이를 CoAP에서 구분하기 위해 CREATE 의 경우에는 oneM2M 프리미티브로서 Resource Type (ty) 파라미터가 존재하므로 따라서 Resource Type 파라미터가 존재하면 CREATE 오퍼레이션, 존재하지 않으면 NOTIFY오퍼레이션으로 구분한다. 또한 CoAP 메시지 헤더에서 지원하는 내용 중 oneM2M 프리미티브와 매핑을 위해서 활용할 수 있는 Code와 옵션의 정보는 제한적이기 때문에, oneM2M의 프리미티브와 매핑이 어려운 부분은 CoAP 페이로드에 담긴다.

### 4.10.2. CoAP Content-Format Negotiation 옵션

CoAP의 Accept 옵션은 요청 메시지를 보낸 CoAP 클라이언트에서 수신 시 지원할 수 있는 Content-Format을 표시하는데 사용한다. CoAP 요청 클라이언트로부터 요청메시지를 받은 CoAP 서버(Hosting CSE)가 해당 Content-Format 옵션에 명시된 Content-Format으로 보낼 수 있다면, Hosting CSE는 요청 메시지에 Accept 옵션이 있을 때, CoAP 수신자(Hosting CSE)가 지원할 수 있는 Content-Format 을 응답 메시지에 담아서 응답할 수 있다. 만약 Hosting CSE가 요청 메시지의 Accept 옵션에 명시된 Content-Format을 지원할 수 없다면 "4.06-Not Acceptable"을 명시하여 응답하게 된다.

### 4.10.3. 토큰

CoAP에서 토큰은 멀티 Hop에 대한 추적이 아니라, Hop-by-hop의 CoAP 요청 메시지와 CoAP 응답 메시지의 매칭을 위해서 사용된다. 따라서 oneM2M 메시지 프리미티브 파라미터 중 response type이 nonBlockingRequestSync 또는 nonBlockingRequestAsync인 경우에 oneM2M 메시지 프리미티브 파라미터 중 Request Identifier를 CoAP의 토큰 값에 매핑 할 수 있다.

### 4.10.4. URI 옵션

oneM2M 메시지 프리미티브에서 "to" 파라미터에 명시된 정보들이 CoAP 메시지 헤더에서 옵션의 Uri-Path로 매핑되며, Uri-Host, Uri-Port 부분은 메시지를 전달하는 상대에 대한 IP-address와 Port 가 명시된 〈remoteCSE〉의 pointOfAccess 속성값을 참조하여 매핑 될 수 있다.

### 4.10.5. 쿼리스트링

oneM2M 메시지 프리미티브의 다양한 요청 메시지 파라미터들이 CoAP의 쿼리스트링 Uri-Query에 매핑 될 수 있다. 예를 들어서 Delivery Aggregation (da), Filter Criteria (fc), Event Category (ec), Request Expiration Time (rqet), Result Expiration Time (rset), Response Type (rt) 등이 가능하다.

### 4.10.6. oneM2M 응답 메시지를 CoAP 응답 메시지에 Mapping

CoAP의 응답 메시지는 성공 리턴 또는 실패 리턴의 형태를 갖는다. 또한 CoAP 요청 메시지에 대한 처리 결과의 상세 정보를 갖는다. 이 밖에 만약 응답 메시지에 oneM2M 리소스 관련 정보가 포함된다면 이는 CoAP 응답 메시지 페이로드 부분에 담길 수 있다. CoAP 응답 메시지가 갖는 상태코드 값은 아래 표 〈4-19〉 및 〈4-20〉과 같다.

표 〈4-19〉 CoAP 응답 메시지의 성공결과 리턴 코드

| Status Code | Status Code of CoAP |
|---|---|
| STATUS_CREATED | 2.01 Created |
| STATUS_DELETED | 2.02 Deleted |
| STATUS_CHANGED | 2.04 Changed |
| STATUS_CONTENT | 2.05 Content |
| STATUS_ACCEPTED | ACK |

표 〈4-20〉 CoAP 응답 메시지의 실패 결과 리턴 코드

| Numeric Code | Description | Status Code of CoAP | Description |
|---|---|---|---|
| 2000 | Location info not authorized | 4.01 | Unauthorized |
| 2001 | Unsupported resource | 5.01 | Not Implemented |
| 2002 | Unsupported attribute | 5.01 | Not Implemented |
| 2003 | Target not reachable | 5.05 | Proxying Not Supported |
| 2004 | Cannot forward, other reason | 5.00 | Internal Server Error |
| 2005 | No privilege | 4.01 | Unauthorized |
| 2006 | Already exists | 4.12 | Precondition Failed |
| 2007 | Create error – missing mandatory arameter | 4.02 | Bad Option |
| 2009 | Does not exist | 4.04 | Not Found |
| 2023 | Create mgmtObj – memory shortage | 4.13 | Request Entity Too Large |
| Numeric Code | Description | Status Code of CoAP | Description |
| 2028 | External Object not found | 4.04 | Not Found |
| 2031 | Cancel execInstance – not cancellable | 4.03 | Forbidden |
| 2032 | Cancel execInstance – already complete | 4.00 | Bad Request |
| 2033 | Delete execInstance – not cancellable | 4.03 | Forbidden |
| 2034 | Delete execInstance – already complete | 4.00 | Bad Request |
| 2035 | Retrieve CSEBase – format error | 4.15 | Unsupported Content-Format |
| 2036 | CMDH rules – non compliant | 4.00 | Bad Request |
| 2037 | Target is not subscribable | 4.05 | Method Not Allowed |
| 2038 | Cannot initiate subscription verification | 5.03 | Service Unavailable |
| 2039 | Subscription verification failed – Originator | 5.03 | Service Unavailable |

### 4.10.7. 추가 정보

CoAP 메시지 헤더에는 8비트 길이의 코드 필드를 포함하며, CoAP 요청 메시지인 경우에는 해당 코드 필드는 요청 메소드를 나타내고, CoAP 응답 메시지인 경우에는 해당 코드 필드는 응답 결과 코드를 나타낸다. 코드 필드가 oneM2M 프리미티브를 위한 모든 응답 결과 코드를 포함할 수 없다면 추가적인 정보는 CoAP 메시지 페이로드에 넣는 방법을 고려한다.

## 4.11. CSE의 리소스 접근방식

OneM2M의 AE 와 CSE, CSE와 CSE간에 메시지는 요청/응답 방식으로 이뤄지고 해당 메시지 플로우가 CoAP의 메시지 플로우와 매핑 하는 과정이 필요하다. CoAP의 메시지 플로우도 HTTP와 같은 방식의 요청/응답을 지원하고 있다. 본 장에서는 oneM2M의 메시지 플로우를 CoAP의 메시지 플로우와 매핑 하는 과정을 살펴본다.

### 4.11.1. 블록킹 액세스

oneM2M의 Response Type (rt) 파라미터가 "blockingRequest"로 설정되었을 때, CoAP으로 매핑시에 Confirmable 메소드로 매핑 되며, 요청메시지에 대한 처리가 성공적이면 응답메시지로 Acknowledge가 전달되며 추가적인 데이터는 CoAP Acknowledge 메시지에 piggy-back되어 적절한 성공 응답 코드가 해당 메시지에 실린다.

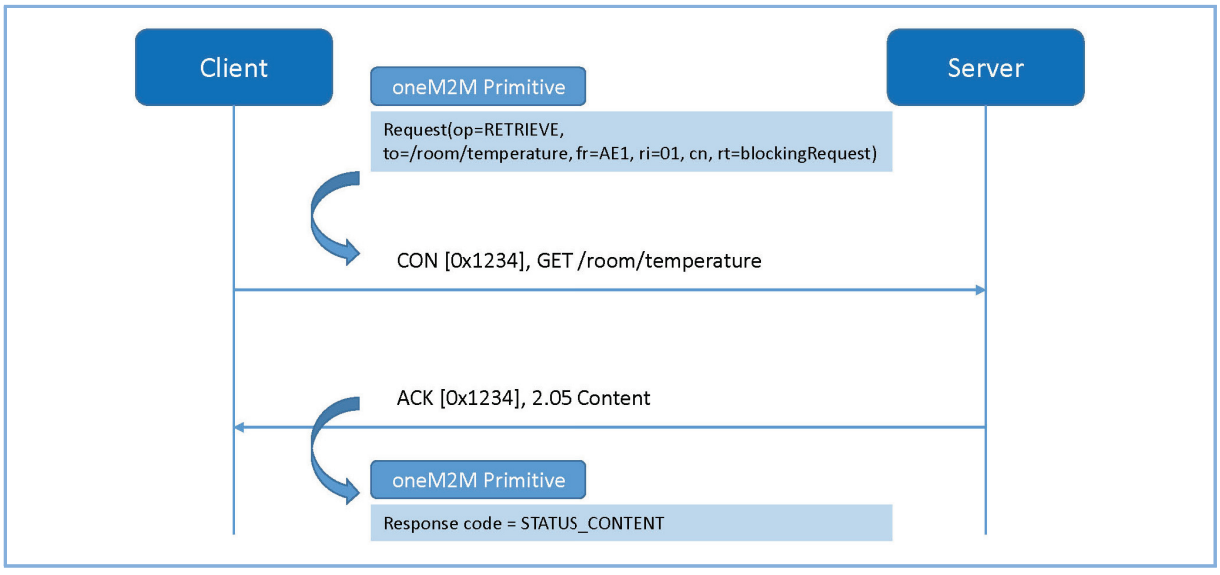



그림 〈4-22〉 블로킹 액세스에 대한 CoAP Mapping

그림 〈4-22〉와 같이 Blocking oneM2M 요청 메시지의 경우에 RETRIEVE 요청의

경우, 해당 CoAP의 GET 요청메시지에 매핑 되고, 해당 CoAP 응답 메시지에 대해서 2.05 성공 메시지는 oneM2M 메시지의 응답코드 STATUS_CONTENT에 매핑 된다.

### 4.11.2. 논블록킹 Asynchronous 액세스

oneM2M의 Response Type (rt) 파라미터가 "nonBlockingRequestAsynch"로 설정되었을 때, CoAP 클라이언트는 Confirmable Method로 매핑 하면서 요청메시지에 토큰값을 oneM2M 메시지 프리미티브의 Request Identifier (ri) 값으로 설정한다. 요청메시지를 받은 CoAP 서버는 요청메시지를 잘 수신했다는 의미를 담은 Acknowledge를 전달한다. 요청메시지를 받은 CoAP 서버는 처리가 끝난 후 적절한 응답 코드와 추가적인 데이터가 piggy-back되어 포함된 Confirmable 메시지를 전송한다. 해당 CoAP 응답 메시지에는 CoAP 요청 메시지에 포함된 토큰값과 동일한 값으로 토큰값을 설정하여 전달한다. 해당 메시지를 받은 수신자는 Acknowledge를 전달하여 메시지 전달을 마치게 된다.

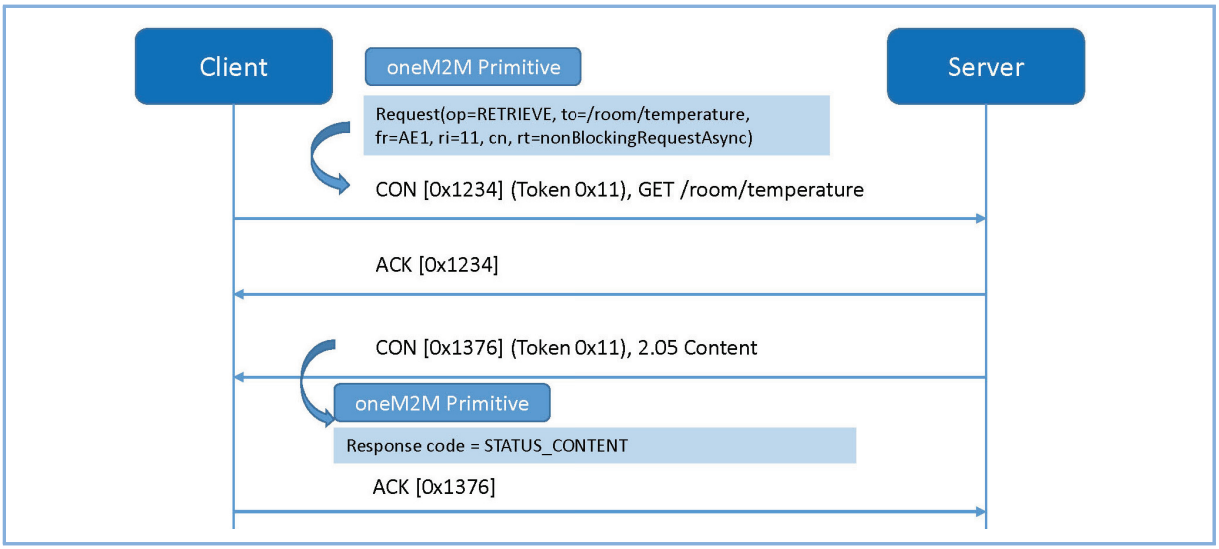



그림 〈4-23〉 논블로킹 Asynchronous액세스에 대한 CoAP Mapping

그림 〈4-23〉와 같이 nonBlocking Asynchronous oneM2M 요청 메시지의 경우에 RETRIEVE 요청의 경우, 해당 CoAP의 GET 요청메시지에 매핑 되고, Request Identifier (ri) 값을 이용하여 CoAP 메시지의 토큰값에 매핑 한다. CoAP 요청 메시지에 대한 Acknowledge 수신 후, 서버에서 요청메시지에 대한 처리를 마치고 나면 클라이언트에게 서버는 CoAP 요청 메시지에 추가적인 데이터를 piggy-back 하여 2.05 성공 메시지를 보낸다. 해당 2.05 Content 성공은 oneM2M 메시지의 응답코드 STATUS_ CONTENT에 매핑 된다.

### 4.11.3. 논블록킹 Synchronous 액세스

oneM2M의 Request Type (rt) 파라미터가 "nonBlockingRequestSync"로 설정된다면 CoAP 클라이언트는 Confirmable Method로 매핑 하면서 요청메시지에 토큰값을 oneM2M 메시지 Primitive의 Request Identifier (ri) 값으로 설정한다. 해당 요청 메시지를 받은 CoAP 서버는 Acknowledge를 요청자에게 전송한다. CoAP 서버는 요청메시지에 대해 응답 메시지를 응답 코드와 함께 CoAP 요청 클라이언트에게 전달한다. 그리고 이때 Request Reference를 같이 전송함으로써, 차후 요청자가 레퍼런스를 통해 처리된 결과를 요청할 수 있도록 한다. 이후 CoAP 요청 클라이언트는 CoAP 서버로부터 받은 레퍼런스를 통해서 리소스 처리 결과에 대해서 접근하며 요청 클라이언트로부터 해당 레퍼런스를 통해서 접근 요청받은 CoAP 서버는 리소스의 현재 처리 상태를 응답한다. 해당 플로우에 대해서는 그림 〈4-24〉를 참조할 수 있다.

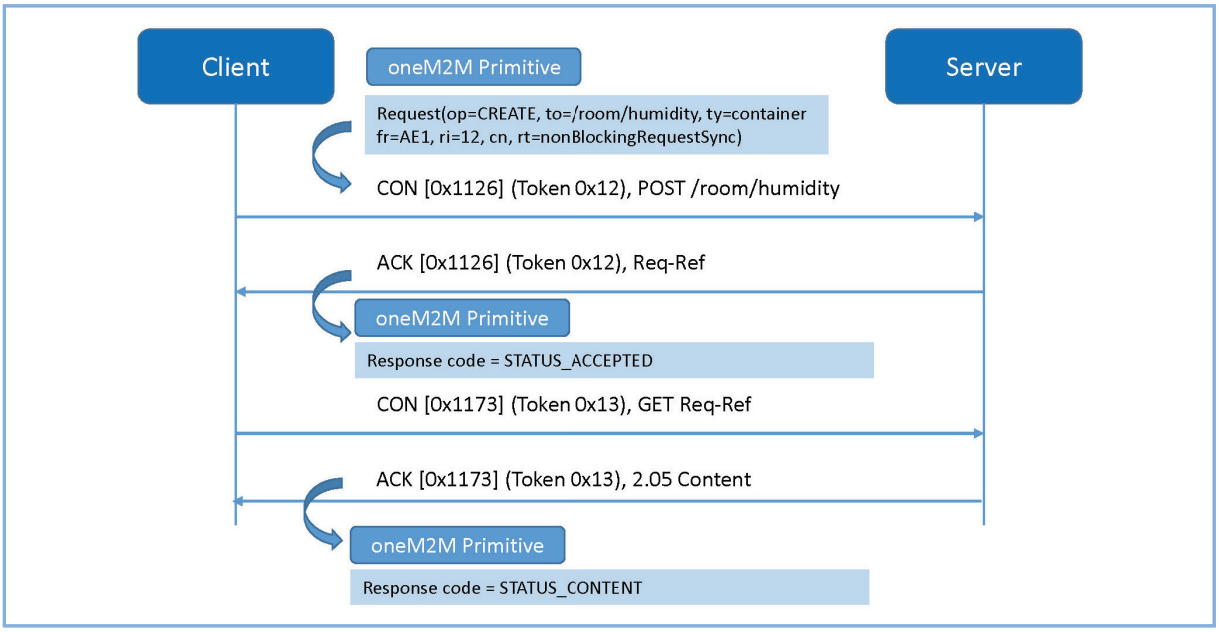



그림 〈4-24〉 논블로킹 Synchronous액세스에 대한 CoAP Mapping

## 4.12. 보안 고려사항

CoAP 자체는 인증, 허가와 관련된 보안을 제공하지 않고 있다. 하지만 CoAP은 HTTP의 경량화 버전으로 볼 수 있고 HTTP 의 보안 세션을 위한 TLS (Transport Layer Security) 방식을 UDP 기반 방식으로 적용한 DTLS (Datagram TLS)을 사용하여 보안 메커니즘을 제공하고 있다. DTLS를 적용하면 CoAP 서버 및 클라이언트간 인증, 데이터 기밀성과 무결성 서비스를 제공할 수 있다. 따라서 현재 CoAP 메시지 페이로드에 포함된 oneM2M 메시지 프리미티브는 DTLS을 통한 Hop-by-Hop 방식으로 보안이 고려된다.

## 4.13. 요약

본 장에서는 oneM2M에서 개발된 프로토콜 문서 및 구체적인 예시를 살펴보았다. 본 장에서 살펴본 코어 프로토콜 기술 규격서는 아키텍처 기술 규격서에서 정의하는 oneM2M 기능적 구조에 대하여, 개발자들이 실제 구현을 할 때 어떤 식으로 개발이 진행되어야 하는지 정의한 문서이다. 또한 CoAP, HTTP 및 MQTT 바인딩 기술 규격서는 oneM2M이 전송 계층 프로토콜을 사용하여 통신이 이루어지는 상황을 고려하여, 해당 전송 계층 프로토콜의 메시지와 oneM2M 프리미티브 간의 매핑을 정의하였다.

# 5. 보안 및 프라이버시

관련 oneM2M 표준 문서

TS-0003: Security Solutions

## 5.1. 보안 아키텍처 (Security Architecture)

본 절에서는 oneM2M 규격에서 고려되고 있는 보안 아키텍처를 설명한다. 그림 〈5-1〉과 같이 보안 아키텍쳐 모델은 세 가지 계층, 즉 Security Functions layer, Security Environment Abstraction Layer, Secure Environment layer로 구성된다.

먼저, Security Functions layer는 참조점(reference point)인 Mca와 Mcc를 통해 이루어지는 보안기능 집합이라고 있다. 보안기능 집합은 6개의 보안기능 카테고리로 나뉘며, 식별과 인증(identification and authentication), 권한부여(authorization), ID관리(identity management), 보안 연계(security association), 민감 데이터 처리(sensitive data handling), 보안관리(security administration) 등으로 구성되어 있다.

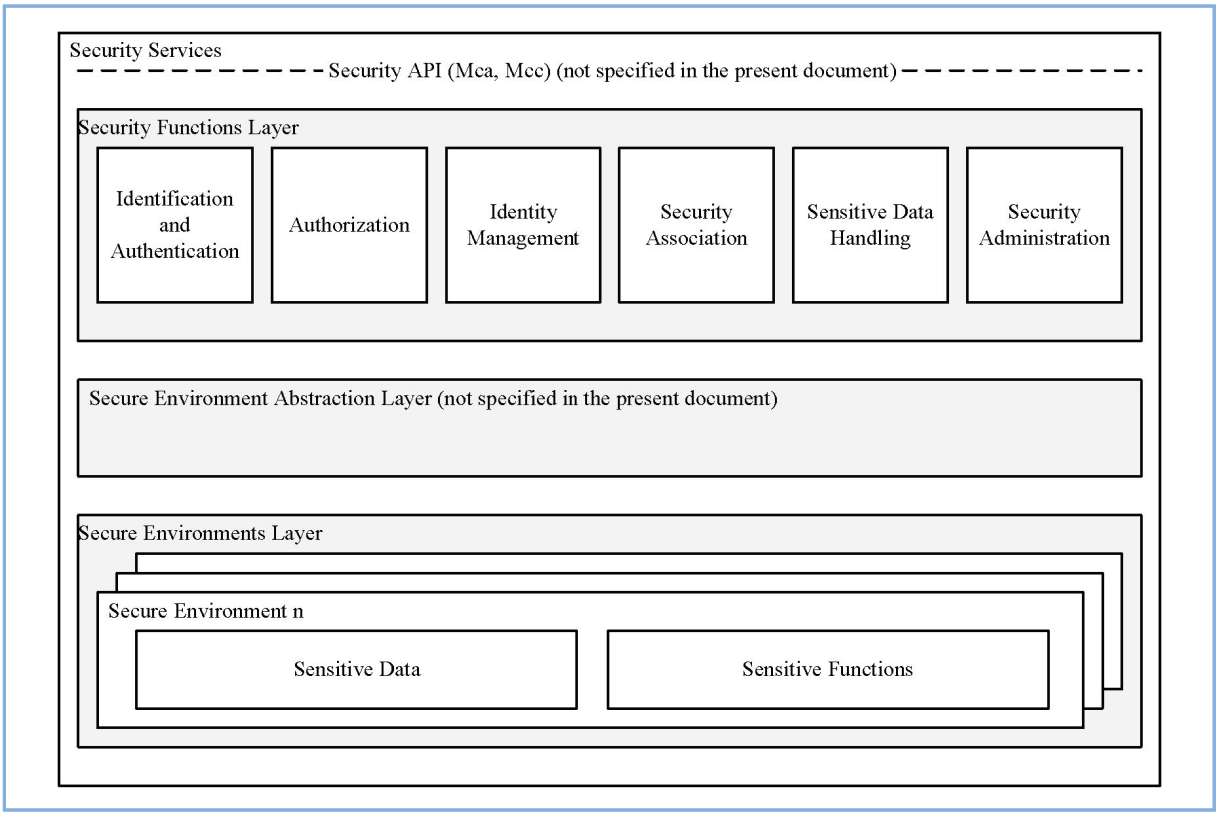



그림 〈5-1〉 보안 아키텍처의 상위레벨 개요

다음 계층인 Security Environment Abstraction Layer에서는 규격에서 명시되어 있지 않지만, 다양한 보안 기능이 제공된다. 예를 들어 보안 기능으로 Protection Level이 있는데 Level 0은 무방어(no protection)라고 해서 방어를 하지 않는 것이고, Level 1은 수동 공격(passive attack)을 방어하는 것인데 한마디로 정보를 빼가는 목적으로 공격하는 것을 차단하는 것이다. 수동 공격의 예로는 도청, 수신정보 수취 등이 있다. Level 2는 원격 공격(remote attack)을 보호하는 것인데 한마디로 정보를 보거나 불법적으로 훔치기 위한 목적으로 하는 공격 및 바이러스나 악성코드를 설치하는 것을 방어하며 지역 소프트웨어 공격(local SW attack)으로부터 보호하는 레벨이다. Level 3은 지역 하드웨어 공격(local HW attack)으로부터 사용자의 컴퓨터를 보호하는 것이다.

세 번째 계층은 여러 개의 Secure Environment를 포함하는데, Secure Environment은 Sensitive Data와 Sensitive Function을 포함하고 있다. 이 중 Sensitive Data는 SE capability, security keys, local credentials, security policies, identity information, subscription information기능을 포함하고 있다. Sensitive Function기능 에서는 data encryption과 data decryption 기능을 포함하고 있다. data encryption에서는 안전한 저장을 할 수 있게 하는 암호화 하는 기능을 지원하고, data decryption에서는 이 암호화된 데이터를 부트스트래핑 하기 위한 방법을 지원한다.

보안 아키텍처를 설계한 원리를 알기 위해서는 일단 CSE(Common Service Entity)를 알아야 하는데, 이것은 M2M Service layer에 있는 미들웨어라고 해서 다른 디바이스의 CSE 혹은 AE(Application Entity)로 바로 연결하여 통신할 수 있는 소프트웨어이다. CSE는 CSF(Common Service Functions)로 이루어져 있는데 이것은 간단히 말하면 CSE의 기능들이다. 그 기능들은 총 13가지로 나뉘며 각 기능마다 요구하는 기능들이 다르다. CSE는 호스트가 되어 필요에 따라 다른 디바이스와 통신하여 모듈을 구성할 수 있는데, 현재 그림의 아키텍처는 모듈식 설계를 제공하는 여러 보안 구성 요소와 하위 구성 요소로 분할되어 있다. 위의 디자인으로 서로 다른 노드와 객체의 아키텍처의 매핑이 가능하다. 매핑을 하기 위해서는 다른 디바이스와 연결이 되어야 하는데, 이 때 사용되는 것이 Security Function Layer에서 언급한 Mcc, Mca이다. Mcc는 CSE의 맨 앞글자를 딴 c를 사용한다. 고로 Mcc라는 것은 서로 다른 디바이스의 CSE와 CSE, 즉 미들웨어끼리의 연결이라는 뜻이고, Mca의 a는 AE의 a를 따온 것으로서, CSE와 AE, 즉 미들웨어와 어플리케이션 사이의 연결을 말한다. 이런 CSE의 보안문제를 해결하기 위해 CSE의 보안환경은 Secure 위 그림의 두 번째 계층인 Environment Abstraction Layer를 통해 접근(access)하는 모든 중요한 자원을 CSE가 직접 가지고 있도록 하였다. 또한 각 기업마다 요구하는 기능들이 다르기 때문에, 위의 그림의 구조는 각 기업이 요구하는 기능들을 충족시키다 보면 구성요소가 조금씩 달라질 수 있다는 것을 염두 해두어야 한다.

이제 Security Functions layer에 있는 여섯 가지 보안기능 카테고리를 설명한다. 먼저 식별(identification)과 인증(authentication)에 대해서 다루겠다. 식별은 인증에 제공된 식별자(ID)가 유효한지 검사 하는 과정으로 식별하는 과정은 인증의 목적에 따라 달라질 수 있다. 인증은 식별과정에서 제공되는 식별자가 자격 증명(trustworthy credential)과 관련이 있는지를 검증하는 과정을 말한다. 예를 들어 어떤 사람이 해외여행을 갈 때 비자나 여권을 가지고 가는데 이러한 것들을 식별자라 하고 공항에서 이것을 검사하는 것을 인증이라고 할 수 있다. 식별 및 인증기능은 CSE 및 AE의 상호 인증을 담당하고 있고 그 인증을 하는데 필요한 것이 식별이다. 식별 과정은 인증의 목적에 따라 수행을 하게 되는데 예를 들어 리소스 접근(resource access)의 경우 인증기능은 AE또는 CSE가 로컬 CSE에 등록 되었는지 확인하기 위해 식별을 요구할 수 있다. AE 또는 CSE 등록의 경우 인증기능은 AE 또는 CSE가 제공 신분이 맞는지 확인하는 식별을 요구 할 수 있다. 인증은 식별 단계에서 제공 한 ID가 신뢰할 수 잇는 권한 정보와 관련되어 있는지 확인하는 처리과정이며, 인증을 처리할 때 어떤 상호 인증 메커니즘을 사용하느냐에 의존적이다. 예를 들어 인증서 기반으로 한 인증 메커니즘을 사용하는 경우, 인증 기능은 디지털 서명을 검증하는 인증을 요구 할 수 있다. 대칭키 기반 인증 메커니즘을 사용하는 경우엔 인증기능은 메시지 인증코드(MAC)을 확인하기 위해 인증을 요구 할 수 있다.

그 다음으로 권한 부여(Authorization) 기능은 접근 통제 정책 및 할당 된 역할에 따라 인증 된 기관에게 서비스 및 데이터 접근을 허가하는 역할을 하며, 인증기능의 인증절차에서 다수의 접근 제어정책을 평가할 필요가 있다. 예를 들면, 인증 평가 과정(ACL, RBAC)은 M2M 서비스 역할이 인증된 객체에 가입하고, 보호 자원과 관련된 접근 통제 정책에 지정 서비스 가입 자원을 활용 한다. 인증 평가 과정은 시간 또는 지리적 위치 등의 상황에 맞는 특성을 고려할 필요가 있는데 발신자의 CSE 또는 AE 및 호스팅 CSE 사이의 인증 보다 상호 인증이 앞서 수행되어야 한다.

세 번째 기능인 식별자 관리(identity management)는 한마디로 ID를 관리하는 것이다. 이것의 기능은 식별자들이 Secure Environment에 저장된 상태라면, OneM2M Identities 또는 Iidentifiers를 필요로 하는 독립적인 개체에 이것들을 제공한다. 그림에서 OneM2M 구조 내에 정의된 oneM2M 식별자는 AE 또는 CSE에 접근하기 쉬운 Sensetive Data로 다뤄질 것이고, 증명에 관련된 어떠한 역할도 부여받지 않고 독립적으로 사용 될 것이다.

네번째 기능인 보안 연계(security association)은 한마디로 통신 링크에 연결된 두 노드 사이의 논리적인 관계이다. 이것의 기능은 안전한 세션 설정과 사물 보안을 통해 안전한 연결을 하는 것이다.

다섯째 기능인 민간 데이터 처리(sensitive data handling) 서비스는 Application Layer에게 세 가지의 기능이 있는 특정 Sensitive Function을 제공한다. 첫째로 안전하게 저장을 할 수 있는 기능을 제공하고, 둘째로 암호화 기능을 지원(data decrtption)하며, 마지막으로 최초의 열쇠를 부트스트래핑 하기 위한 방법(data decrtption)을 지원한다.

여섯번째 기능인 보안관리(security administration) 서비스는 Sensitive Function, Resources 그리고 Attributes를 관리하는 기능을 제공한다. 이것은 Secure Environment를 통해 제공되는 Resources 관리도 포함한다. 보안관리는 보안환경(독립적인 하드웨어 모듈, 신뢰할 수 있는 통합 실행환경 또는 소프트웨어 보호)의 독립적인 유형이다.

## 5.2. 자원의 권한부여(Authorization) 기술

oneM2M 표준에서 권고되는 권한부여(Authorization) 절차는 가용 자원(Resource)과 서비스에 대한 사용자 혹은 사물들의 접근을 제어하기 위한 기술(access control)을 다룬다. oneM2M 시스템 구조에서 서비스는 기본 구성 요소인 공통 서비스 개체(CSE: common service entity)와 응용 개체 (AE: application entity)에 의해 관리된다. oneM2M에서 표준화 하고 있는 권한부여 절차는 다음의 두 가지를 기본 요건으로 하고 있다.

- ✓ 자원에 대한 접근 요청 메시지의 발신자가 인증 기능 (Authentication Function)에 식별되어야 함
- ✓ 접근 제어 절차의 두 주체인 요청 메시지 발신자와 수신자가 상호 인증되어야 함

자원 요청 메시지를 통해 접근이 요청되는 자원은 메시지와 연관된 accessControl PolicyID 애트리뷰트를 갖는다. 이 애트리뷰트에는 요청하는 자원에 적용될 수 있는 〈accessControlProlicy〉 자원들을 식별하는 값들의 목록이 포함된다. 〈accessControlProlicy〉 자원에 대한 구조 및 정보는 “Resource Type accessControlPolicy (TS-0001)”에 정의되어 있다. 각각의 〈accessControlPolicy〉 자원은 권한(privileges)과 자가권한 (self-Privileges) 속성을 갖게 되는데 각 속성은 접근 제어 규칙에 명시되어 있는 정보로 구성되어 있다. 여기서 권한 속성은 하나의 자원을 참조하는 다른 자원에 적용되는 규칙들의 집합을 나타내며, 자가권한 속성은 〈accesscontrolPolicy〉 자원 자체에 적용되는 접근 제어규칙 집합을 나타낸다. 그림 〈5-2〉는 자원의 인스턴스와 〈accessControlPolicy〉 인스턴스와의 관계를 나타낸다. 그림에서 표기된 ACP는 〈accessControlPolicy〉자원의 인스턴스를 나타낸다. 또한 그림 〈5-2〉에서 지정된 접근 제어 방식은 속성 기반 접근 제어(ABAC: attribute based access control)의 개념을 따른다. ACP에 대한 자원접근 요청 시, ACP의 자가권한 속성 정보에 대한 검토가 이루어진다. 그 외의 다른 모든 유형의 자원 인스턴스에 대한 접근 요청은 접근하고자 하는 목표 자원(target resource)과

관련된 ACP 집합(set)의 권한 속성정보에 대한 검토가 이루어진다. 예로, ACP 자원에 대해 자원접근 요청이 온 경우, 최소 한 개의 자가권한 속성의 값이 "허가(Permit)"라고 판단되면, 자원접근 요청에 접근 권한이 부여된다. 그 외의 다른 자원 유형에 대한 자원 접근 요청인 경우, 최소 한 개의 권한 속성이 "허가"라고 판단될 경우에만 자원에 대한 접근권한이 부여된다.

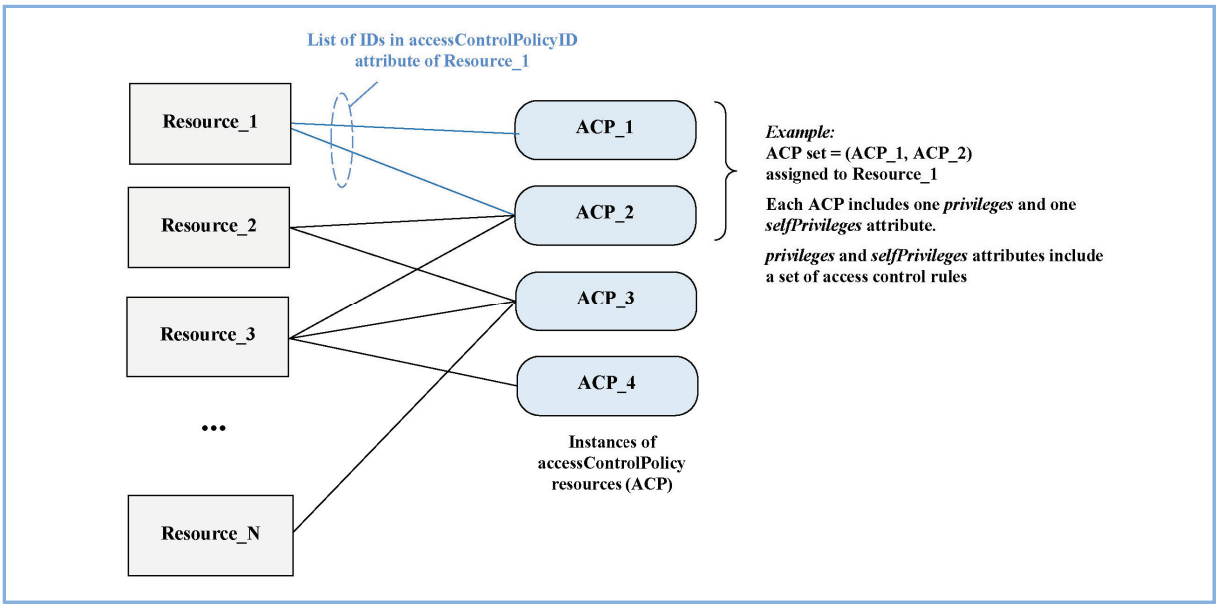



그림 〈5-2〉 자원 인스턴스와 접근 제어 정책 간 관계도

권한 속성과 자가권한 속성은 ACP에 정의되어 있으며 이는 접근권한 요청자가 접근하고자 하는 자원에 접근하여 수행하는 작업(검색, 갱신, 생성 및 삭제)에 대한 접근 허용 결정을 내린다. 뿐만 아니라 특정 상황에 따른 각종 제약 사항(접근시간에 대한 제약, 권한 요청자의 IP에 대한 제약, 위치에 대한 제약 등)에 대해 판단하고 결정한다. ACP에 정의된 정책은 권한부여 논리 구조를 사용한 접근 제어 매커니즘에 의해 적용되며, 자원 접근권한 요청 메시지에 포함된 파라미터, 문맥 정보, 접근을 제어하는 정책 정보를 조합하여 접근권한 부여 결정을 내리게 된다. 다음 그림 5.3은 접근결정 알고리즘 동작 과정을 나타낸다. 그림에 명시된 접근결정 알고리즘은 권한 속성과 자가권한 속성에 포함된 각각의 접근제어규칙을 통해 얻어진 결과의 일부분을 결합하는 형식을 가진다. 접근 결정 알고리즘 중의 결과를 결합하는 방법으로는 접근제어규칙과 ACP에 대한 결합 알고리즘인 "허가-재지정(Permit- overrides)" 알고리즘을 사용하며 다음 두 가지 결과로 판단 될 수 있다.

- ✓ 단일ACP의 속성이나 자가속성에 포함된 단 하나의 접근제어규칙 결정이 "허가(Permit)"라 판단되면 접근 결과는 "허가"로 판단
- ✓ 상기의 경우가 아닐 경우 접근 결정의 결과는 "(거부)Deny"로 판단

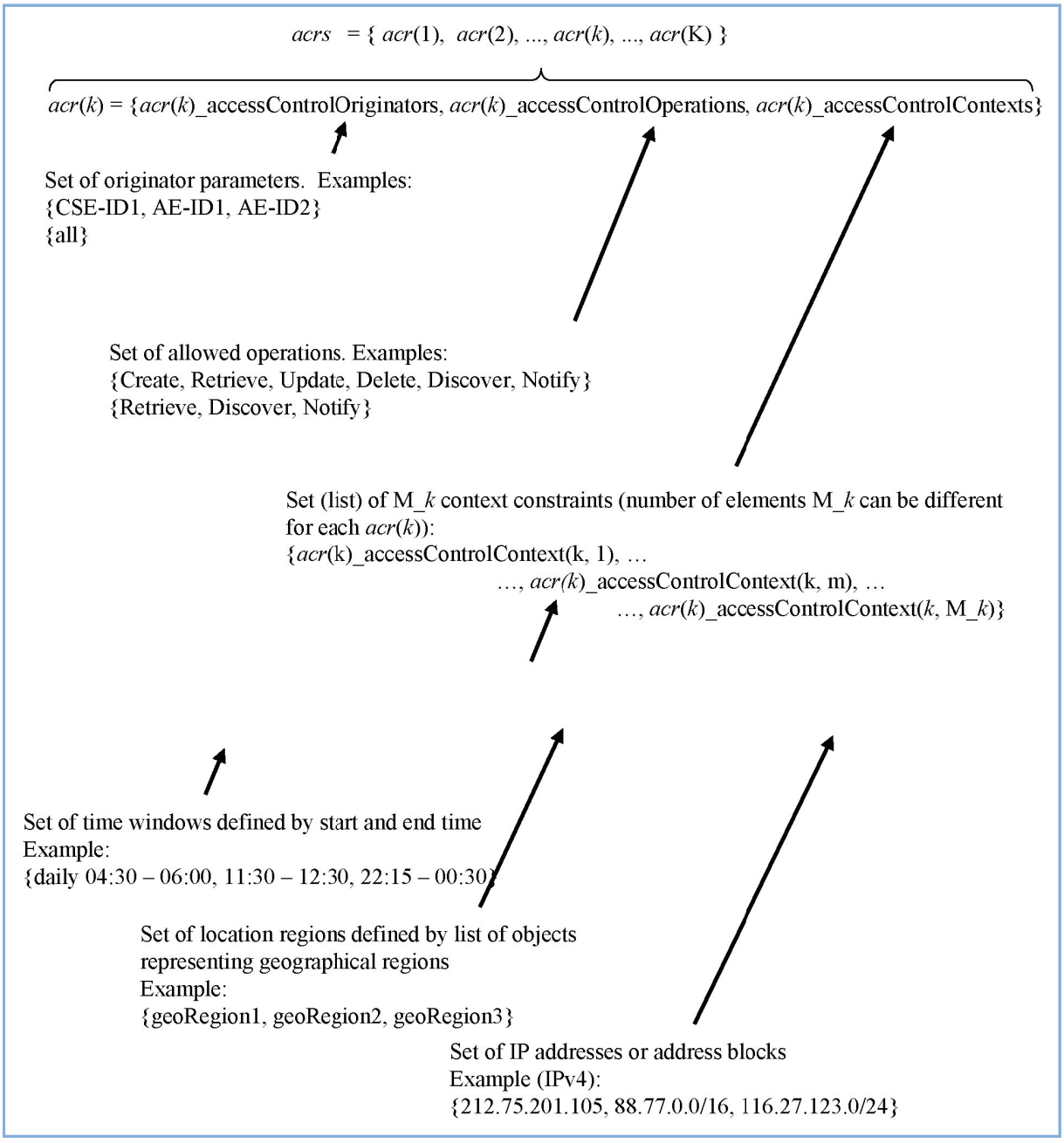



그림 〈5-3〉 접근 결정 판단 알고리즘

## 5.3. 보안 프레임워크

oneM2M 표준 기술은 M2M 응용에서 발생 할 수 있는 다양한 시나리오에 대응하기 위해 시스템 내부의 보안 설정(Security establishment)을 위한 방안과 구성(provisioning) 방법들을 제공한다.

대칭키 기반 보안 프레임워크(Security Framework)에서, 두 통신 주체 간(즉, 각 쌍의 엔티티)에 수행되는 상호인증을 위해 공유된 대칭키가 사전 설정 되어야 한다.

이 대칭키는 디바이스의 제조, 배포 시 사전에 구성(Pre-provisioning) 되거나 SA (Security Association) 단계에서 RSPF(Remote Security Provisioning Framework)를 통해 설정될 수 있다. RSPF를 기반으로 키를 공유하는 두 엔티티는 CSE/AE와 MAF(M2M Authentication Function) 또는 한 쌍의 CSE/AE가 될 수 있다. RSPF 종류로는 사전 구현 대칭키 기반 RSPF, 인증서 기반 RSPF, GBA(Generic Bootstrapping Architecture)기반 RSPF가 있다. 사전에 설정되는 대칭키는 '설정된 대칭키 기반 SAEF (Security Association Establishment Framework)' 또는 'MAF 기반 대칭키 SAEF'에 적용될 수 있다. 설정된 대칭키 기반 SAEF에서는 엔티티 간 SA 생성을 위해 두 엔티티 간 공유된 대칭키 'Kpsa'와 이에 대응하는 대칭키 식별 값 'KpsaId'를 사용한다. 이 경우 Kpsa와 KpsaId는 사전에 설정되거나 원격 설정이 가능하다. 이 밖에 MAF기반 SAEF에서는 마스터 크리덴셜 'Km'과 이에 상응하는 마스터 크리덴셜 식별 값'KmId'를 사용한다. 먼저 사전에 키가 구성 설정되는 대칭키 기반 RSPF에 대해 좀 더 자세히 설명한다. 사전에 설정된 대칭키 기반 RSPF가 수행되기 위해 부트스트랩 크리덴셜 설정 단계에서 대칭키가 사용되며, 이 키는 'Kpm'으로 표기 된다. Kpm은 적절하게 보호되지 않으면 보안 위험을 초래할 수 있으므로 안전한 장소에 보관되어야 한다.

그림 〈5-4〉는 대칭키에 해당하는 'Kpm' 기반 RSPF 동작 흐름을 나타내며, 부트스트랩 크리덴셜 설정, 부트스트랩 명령 구성, 부트스트랩 보안 핸드쉐이크, 등록 키 생성, 보안 협약 핸드쉐이크를 위한 통합 과정들로 구성된다. 대칭키 Kpm과 이에 상응하는 식별 값 KpId는 등록자 (Enrolee)와 MEF(M2M Enrolment Function)에 사전에 설정되어 있다. 부트스트랩 명령 구성단계에서 Enrolee와 MEF에는 원격 설정 방식에 필요한 정보들이 구성되며 초기 원격 구성을 위해 등록자에는 다음의 항목들이 구성된다.

- ✓ 등록 대상의 식별(Identity): 등록자가 설정 되기 위한 대상을 식별함
- ✓ MEF의 인자: MEF는 KpmId 또는 M2M 등록 기능 URI를 이용하여 식별될 수 있음

MEF는 등록자가 등록 목표(Enrolment Target)로의 접근을 가능하게 하기 위한 방안으로 등록자 원격 설정을 수행할 수 있다. 이를 위해 다음 항목들이 구성된다.

- ✓ 등록 대상의 식별(등록자가 설정 되기 위한 등록 대상을 식별)
- ✓ 등록자의 할당된 CSE-ID 또는 AE-ID(등록자ID), 등록 대상의 요청을 받을 때, M2M 등록 기능은 등록 대상에 대한 등록자 Km, Kpsa가 엔티티 식별에 제공
- ✓ 등록자는 KpmId 를 이용하여 M2M 등록 기능에 식별될 수 있음

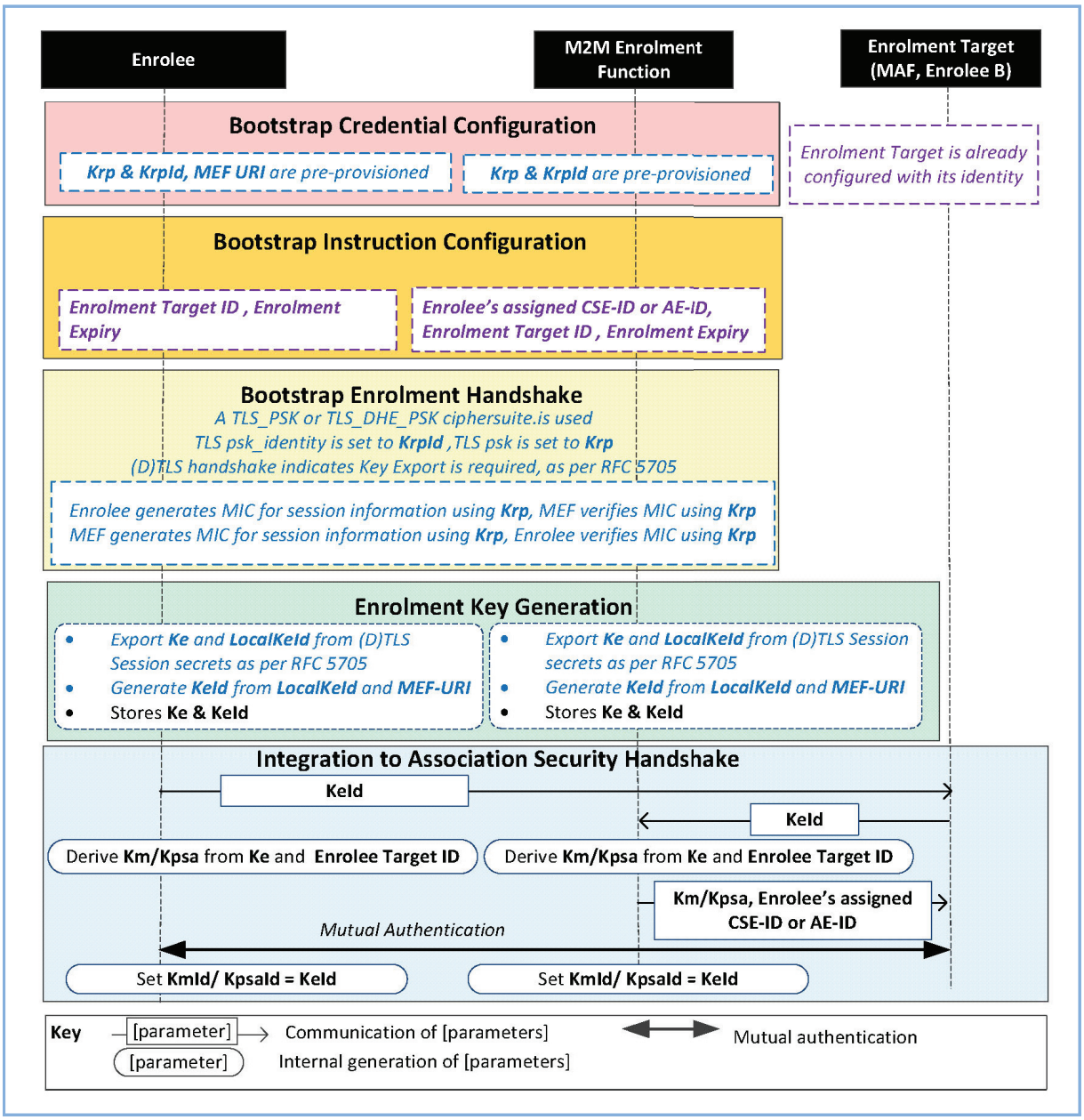



그림 〈5-4〉 Kpm 기반 RSPF 구조도

이후, 부트스트랩 과정에서 등록자와 MEF는 안전한 보안 세션을 개설하기 위해 (D)TLS-PSK 핸드쉐이크 과정을 수행한다. "psk_iden-tity" 파라미터에는 사전 설정/구성되는 등록자의 대칭 키 식별 값인 'KpmId'가 설정된다. "psk" 파라미터는 사전 설정된 대칭키 등록자의 키인 'Kpm'가 설정된다. 등록키 생성과정은 다음과 같다.

- ✓ 등록키 'ke'와 RelativeKeId는 RFC 5705에 정의된 TLS Key Export 기능을 이용하여 (D)TLS 세션 비밀값으로 부터 생성
- ✓ 등록 키 식별자 'KeId'는 RelativeKeId와 M2M 등록 기능의 FQDN으로 부터 생성
- ✓ 등록자와 M2M 등록 기능은 등록 키 'Ke'와 등록 키 식별 값 'KeId'저장

✓ 등록 키 생성을 위한 사전 프로비저닝 대칭 등록자 키 기반 RSPF는 등록 키 생성을 위한 인증서 기반 RSPF와 동일함

대칭키 기반 RSPF와는 달리, 인증서 기반 RSPF의 부트스트랩 크리덴셜로는 인증서가 사용된다. 아래 그림 〈5-5〉는 인증서 기반 RSPF 구조를 나타낸다. 그림에서 부트스트랩 크리덴셜 설정단계에는 등록자와 MEF가 공개키 인증서를 이용하여 상호 인증을 수행하며, 부트스트랩 크리덴셜인 인증서는 각 엔티티에 사전 설정 된다. MEF의 아이덴티티와 등록 대상은 사전에 설정되었다고 가정한다. 이후, 부트스트랩 명령 구성 단계에서 등록자와 MEF에는 원격 구성 권한 부여를 위해 필요한 정보가 설정되며, 등록자에 초기 원격 구성을 처리하기 위해 다음의 정보들이 설정된다.

✓ MEF URI: (D)TLS 교환 라우팅의 목적으로 원격 구성을 용이하게 함
✓ MEF 인증서 인증에 필요한 정보
✓ 등록 대상의 식별

이와 더불어 MEF에는 다음과 같은 정보들이 설정된다.

✓ 등록자의 인증서 인증에 필요한 정보
✓ 인증 대상 식별 : 등록자 인증서 정보를 이용한 인증
✓ 등록자에 할당된 CSE-ID 또는 AE-ID (등록자 ID)

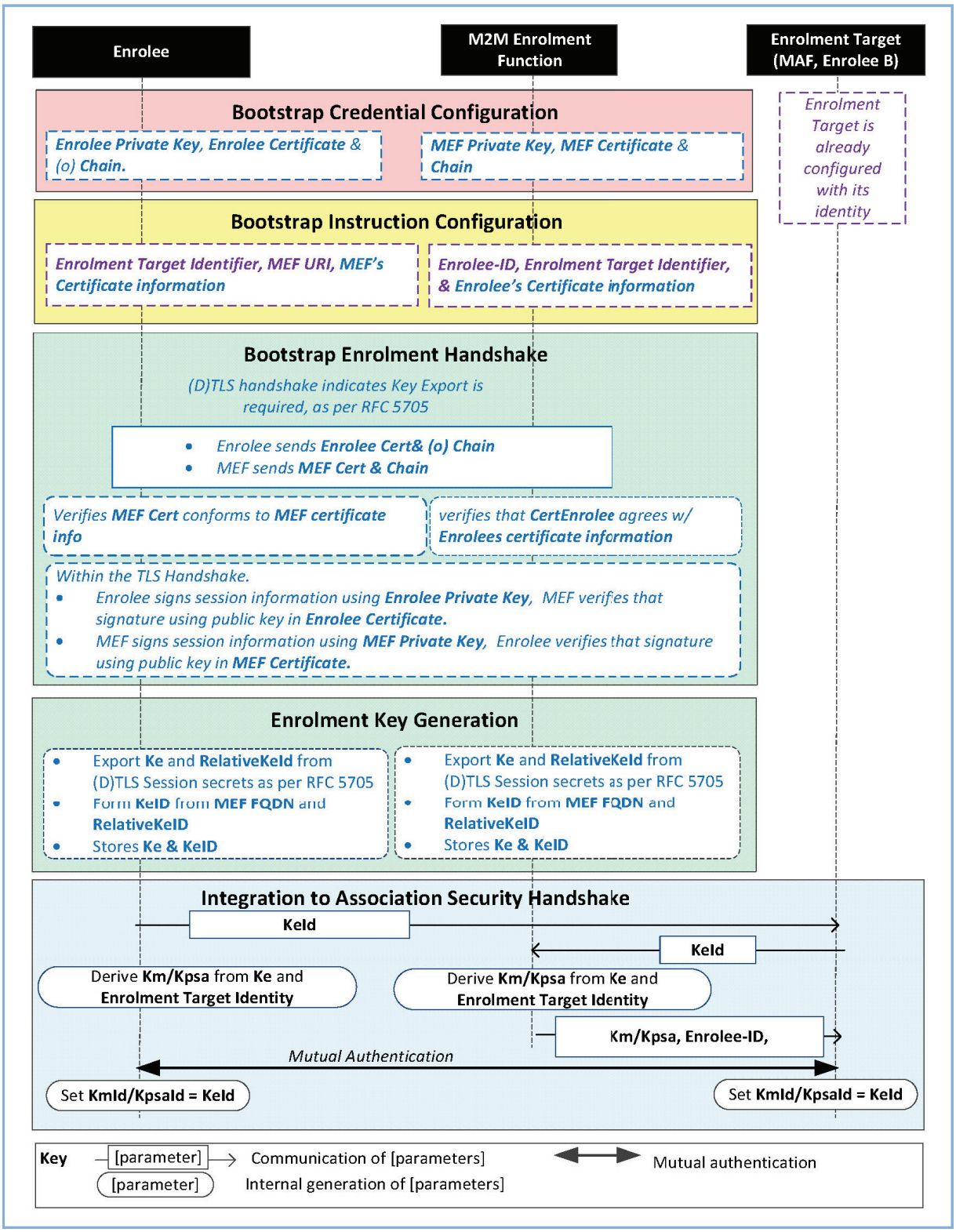



그림 〈5-5〉 인증서 기반 RSRF 구조도

부트스트랩 보안 핸드쉐이크 과정에서 등록자와 MEF는 안전한 보안세션 생성을 위해 (D)TLS 핸드쉐이크 과정을 수행한다. 각 엔티티(등록자, MEF)는 상대 엔티티의 인증서

를 확인하고 등록자와 MEF는 유효한 인증서를 이용하여 상대방을 인증한다. 등록 키 생성 과정은 프리-프로비저닝 대칭키 RSPF에서 등록 키 생성에 대해 기술 한 것과 동일하다.

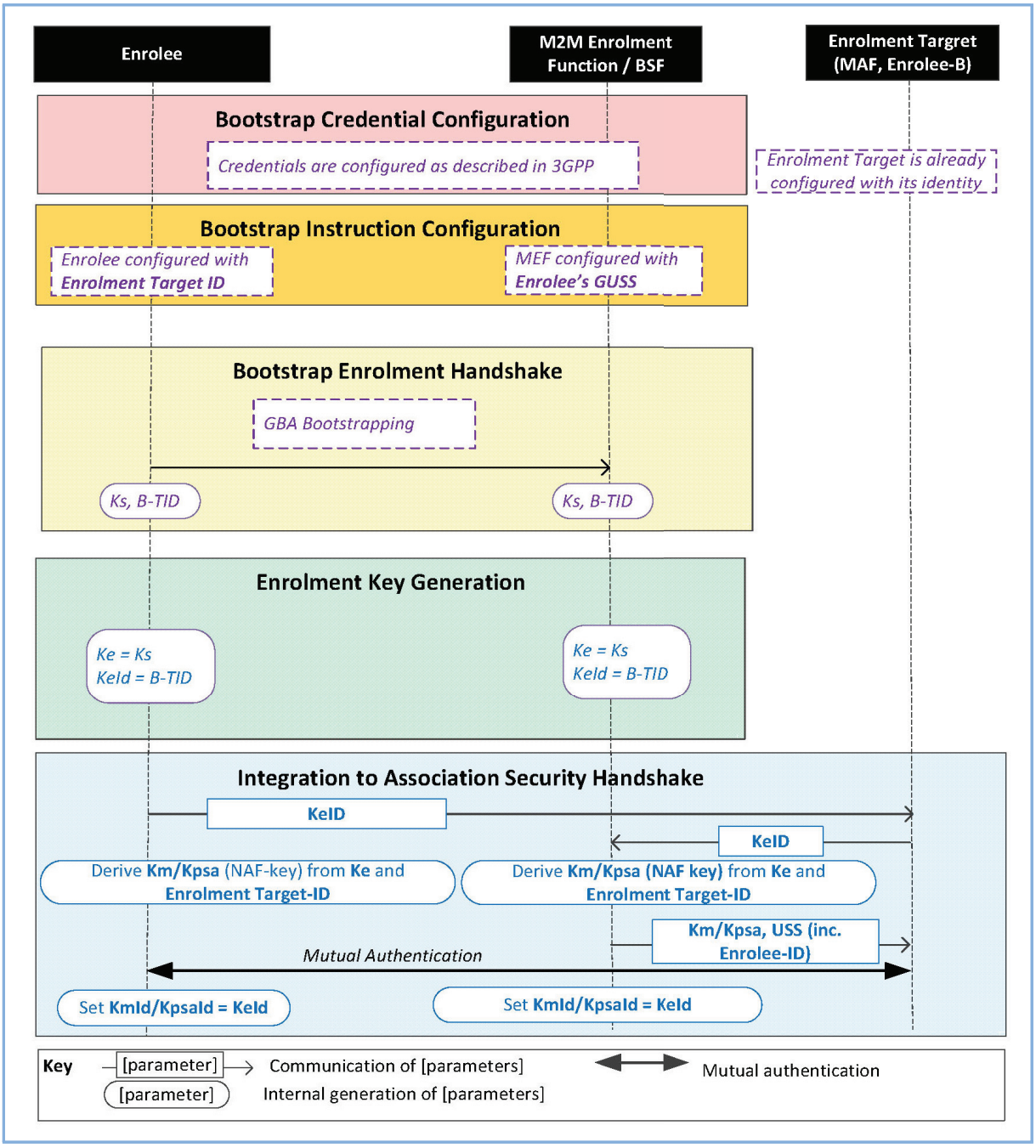



그림 〈5-6〉 GBA기반 RSPF 구조도

마지막으로, 그림 〈5-6〉은 GBA(Generic Bootstrapping Architecture)기반 RSPF 구조를 나타낸다. 어플리케이션 서비스/미들 노드와 등록 대상, M2M 어플리케이션 서비스/미들 노드는 마스터 크리덴셜 'Km'을 공유하거나 구성된 보안 연결 키에 해당

하는 'Kpsa'를 공유하기 위해 성공적인 GBA 부트스트래핑을 수행하여야 한다. 또한 이를 통해 NAF 키 (Ks_(ext/int)_NAF)를 유도 해야 한다. NAF 키는 마스터 크리덴셜 'Km' 또는 구성되는 보안 연결 키인 'Kpsa'를 의미한다.

부트스트랩 크리덴셜 설정단계에서는 등록자와 MEF를 위한 크리덴셜이 설정되고 MEF는 BSF(Bootstrap Server Function)의 역할을 한다. 등록자와 MEF간 상호인증을 수행하기 위한 크리덴셜은 특정 UNSP이다. 부트스트랩 명령 구성 단계에서 등록자와 MEF, MAF에는 권한 부여와 원격 구성에 필요한 정보들이 구성된다. 부트스트랩 등록 핸드쉐이크 단계에서 핸드쉐이크는 등록자와 MEF간 공유된 키인 'Ks'의 수립(Establishment)이 가능하도록 한다. 만약 부트스트랩 된 키인 Ks가 이미 공유되었고, 또한 유효하다면 부트스트랩 등록 핸드쉐이크 과정은 불필요하다. 등록 키 생성 단계는 기존 GBA 부트스트랩 된 키 Ks를 가질 수 있다. 등록 키 생성 단계에서 등록 키 'Ke'는 부트스트랩 등록 핸드쉐이크 단계에서 생성된 GBA 부트스트랩 키 'Ks'와 동일해야 한다. 또한 등록 키 식별자 'Ke-ID'는 부트스트랩 등록 핸드쉐이크 과정 중에 생성된 부트스트래핑 트랜잭션 식별자 'B-TID'과 동일해야 한다.

보안 핸드쉐이크 연관(association)을 위한 통합 단계에서 등록자와 등록 대상은 마스터 크리덴셜 'Km' 또는 구성된 보안 커넥션 키 'Kpsa'를 설정해야 한다. 이 경우 등록 대상은 NAF 역할을 수행한다. 따라서 등록자와 등록 대상은 NAF-specific 키를 생성해야 한다. 등록 대상은 MEF/BSF로 부터 등록자의 사용자 보안 설정 (User security settings) 정보를 전송 받는다. 이때 두 경우에 따라 각 각 키의 설정은 달라진다.

- ✓ GBA_ME인 경우, NAF-specific 키는 Ks_NAF
- ✓ GBA_U인 경우, NAF-specific 키는 Ks_int_NAF와 Ks_ext_NAF
- ✓ 마스터 크리덴셜 또는 구성 보안 연결 키 'Kpsa'는 NAF-specific key 이어야 한다.
- ✓ GBA_ME인 경우, Km/Kpsa=Ks_NAF
- ✓ GBA_U인 경우 만약 HTTP Client가 UICC에 있는 경우 Km/Kpsa = Ks_int_NAF 이고, 그렇지 않으면 Km/Kpsa = Ks_ext_NAF 임

등록자와 등록 대상은 마스터 크리덴셜 식별값 'Km-Id' 또는 'KeID'의 값에 부여된 구성 보안 연결 키의 식별값 'Kpsa-Id'를 설정해야 한다. 등록자와 등록 대상은 마스터 크리덴셜 'Km' 또는 구성 보안 연결 키인 'Kpsa'를 사전에 공유된 키(pre shared Key)로 이용하여 (D)TLS-PSK 핸드쉐이크를 수행해야 한다. 만약 UICC가 원격 보안 프로비저닝이 지원되는 안전한 환경에서 사용되는 경우, GBA-U와 Kc=KS_int_NAF는 인증과 키 교환을 위해 사용 되어야 한다.

## 5.4. 설립 프레임워크 (SAEF)

oneM2M 표준에서 SA(Security Association)는 Mcc, Mcc' 또는 Mca의 참조 점에서 단일 홉으로 제한된다. oneM2M 시스템은 각 엔티티에서 분배된 대칭키를 갖는 "provisioned symmetric key" SA 생성 방안과 인증서 기반 SA 생성 방안, 그리고 MAF (M2M Authentication Function) 기반 SA 생성 방안을 제공한다. 각각의 SAEF는 크리덴셜 설정 단계, 연관 설정(Association Configuration) 단계, 연관 보안 핸드쉐이크(Association Security Handshake) 단계로 구분된다.

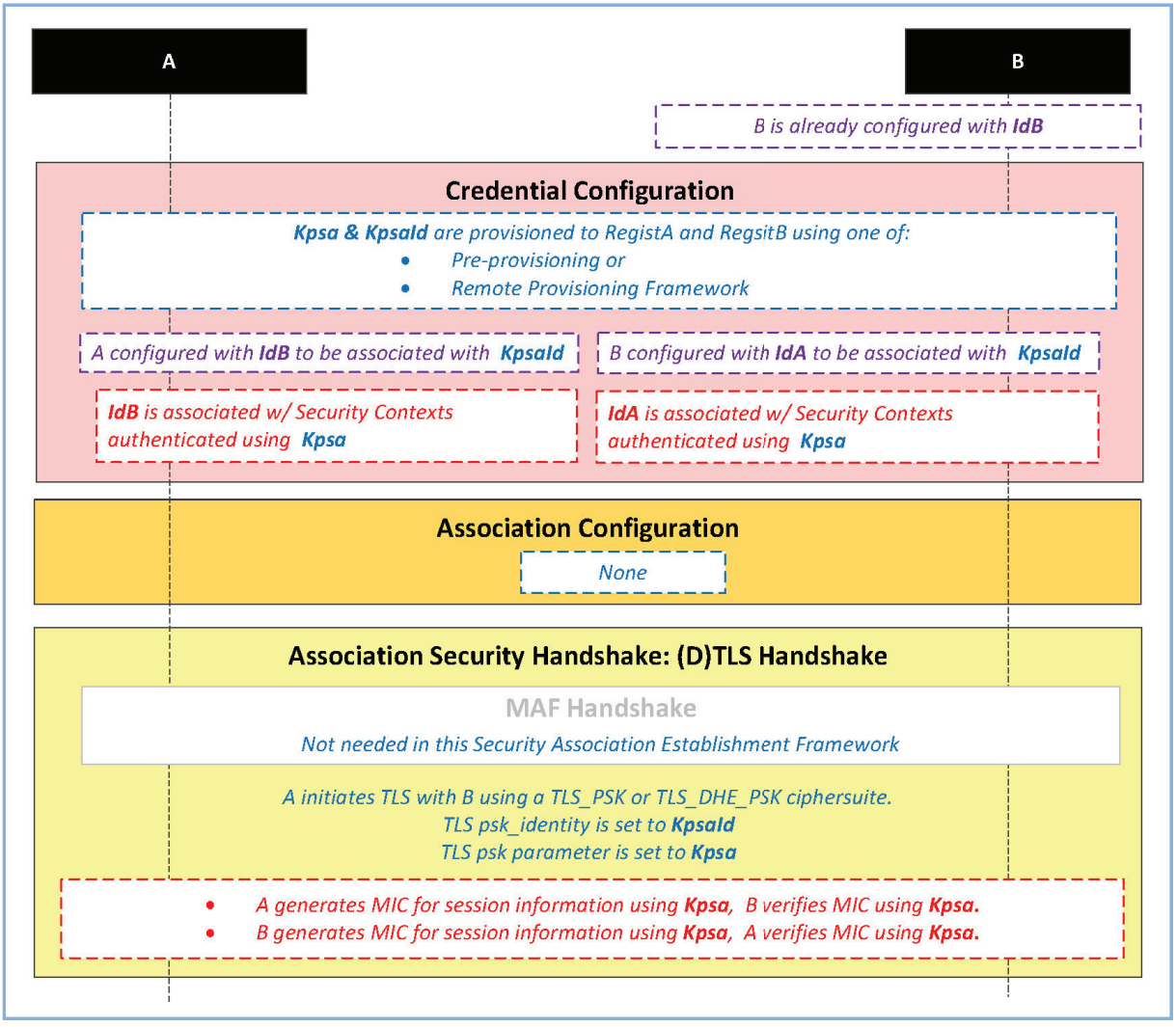



그림 〈5-7〉 프로비저닝 키 기반 SAEF 구조도

첫 번째로, 대칭키 기반 SAEF는 공통 서비스 요소(CSE)간 상호 인증 또는 CSE와 어플리케이션 요소(AE) 간 상호인증을 수행 한다. 이때 각 엔티티에 분배된 대칭키가 상호인증 과정의 크리덴셜로 사용된다. 이 키를 구성된 보안 연결 키라고 하며 'Kpsa'로

표기한다. 각 엔티티는 설치/공급(Provisioning)된 보안 연결 키를 이용한 핸드쉐이크 메시지의 MAC(Message Authentication Code)로 서로를 인증한다. 보안 커넥션 키 Kpsa는 안전하게 보관되지 않을 경우 보안 상 문제가 발생 할 수 있으므로 안전한 환경에 저장되어야 한다.

그림 〈5-7〉은 SAEF에서 구성된 키를 사용하는 상호 인증 과정에 대해 나타낸다. 아래 그림에서 "Entity A"와 "Entity B"는 각각의 CSE 또는 CSE와 AE로 대응될 수 있다. 또는 반대로 AE와 CE가 각각의 Entity A, Entity B로 대응될 수 있다.

위 그림의 자격증명 구성(credential configuration) 단계에서는 구성된 보안 연결 키 'Kpsa'와 이에 해당하는 키 식별자 'KpsaId'는 사전에 설정되어 있거나 원격 구성되어 있다. 협약 구성 (Association configuration) 단계에서 각 엔티티에는 상호 인증과 식별에 필요한 정보들이 설정된다. 마지막 협약 보안 핸드 쉐이크(Association Security Handshake) 단계에서는 안전한 세션을 개설하기 위해 (D)TLS-PSK 핸드쉐이크 과정이 수행된다.[4] 수행과정에서 "psk_identity" 파라미터에는 보안 연결에 필요한 식별자 값인 KpsaId가 설정되며, "psk" 파라미터에는 구성 보안 연결 키인 Kpsa가 설정된다.

그림 〈5-8〉은 인증서 기반 SAEF의 구조를 나타낸다. 그림에서 "Entity A"와 "Entity B"는 CES 또는 CSE 와 AE로 각각 대응될 수 있다. 자격증명 구성(Credential Configuration) 단계에서 각각의 엔티티를 위한 개인키와 인증서는 사전에 구성 되어 있다. SA 구성 단계에서 엔티티 A와 엔티티 B에는 보안 핸드쉐이크 수행 중 필요한 인증 정보, 식별 정보가 설정된다. 엔티티 A는 보안 핸드쉐이크를 개시하도록 명령받고, 해당 명령에는 엔티티 B의 인증서 정보와 식별값(IdB)이 포함된다. 엔티티 B에는 엔티티 A가 엔티티 B와 안전하게 핸드쉐이크를 수행할 수 있는 권한 정보가 구성된다. 엔티티 A의 식별값 'IdA'는 엔티티 B가 엔티티 A를 인증하기 위한 값으로 사용 된다. 협약 보안 핸드쉐이크 과정에서 각 엔티티는 상대 엔티티의 인증서를 확인하고, TLS 1.2 RFC 5246[5]과 DTLS 1.2 RFC 6347[6]의 규격을 따르는 유효한 인증서를 이용하여 인증한다.

4) IETF RFC 4279 "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)"
5) IETF RFC 5246 "The Transport Layer Security (TLS) Protocol, Version 1.2"
6) IETF RFC 6347 "Datagram Transport Layer Security Version 1.2"

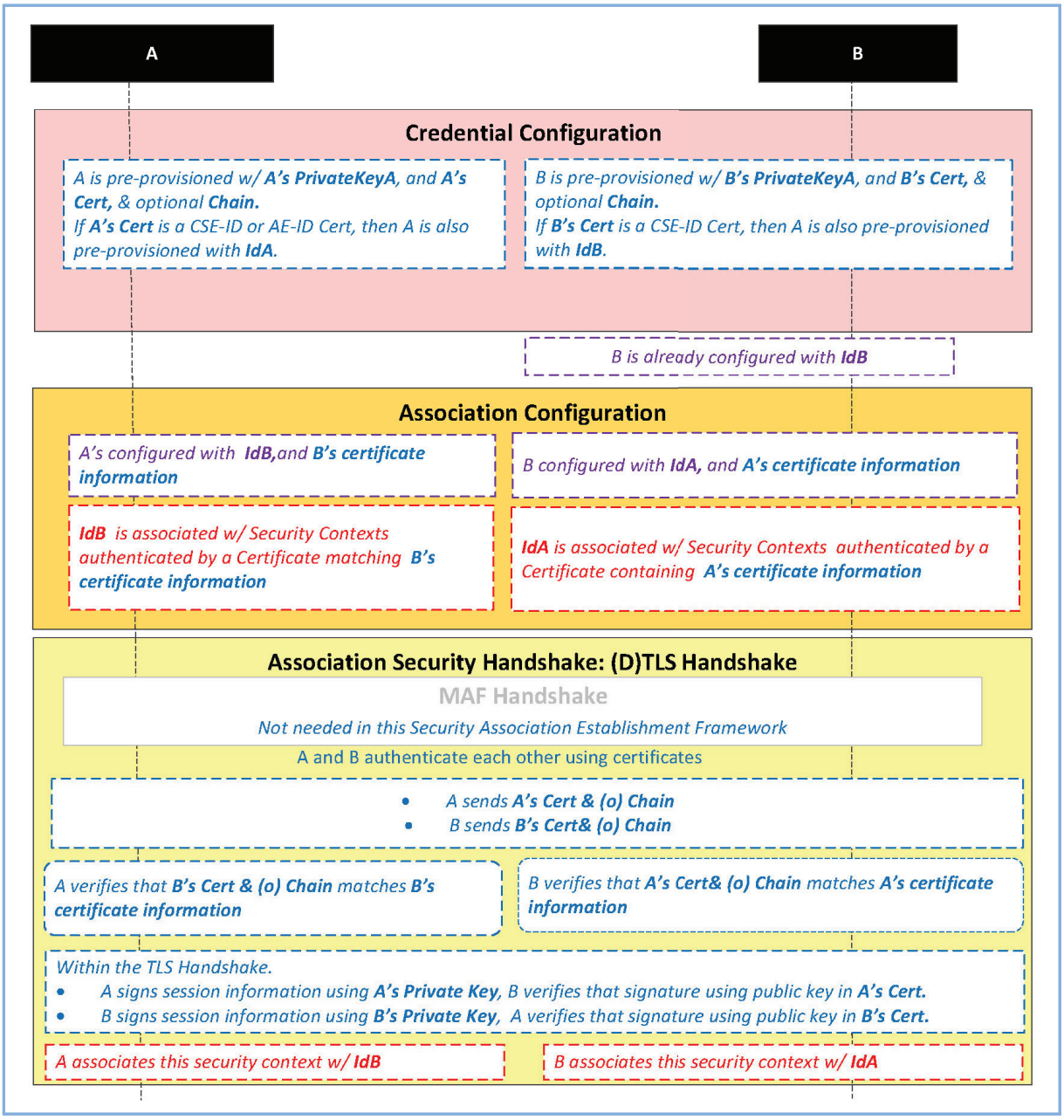



그림 〈5-8〉 인증서기반 보안 SAEF 구조도

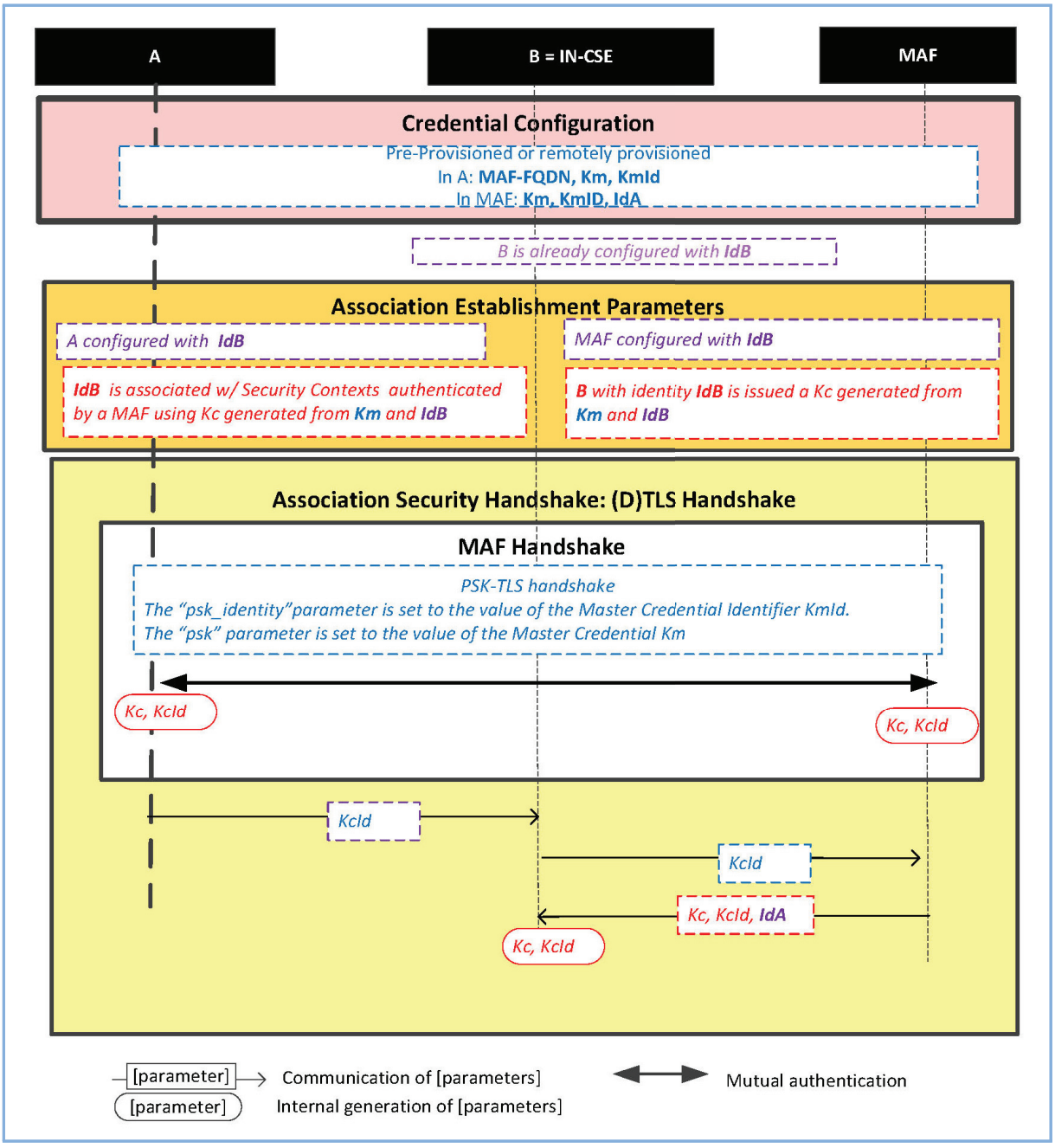



그림 〈5-9〉 MAF기반 SAEF구조도

마지막으로 그림 〈5-9〉와 같이 MAF기반 SAEF는 엔티티 B가 인프라 노드인 경우 사용될 수 있다. 자격증명구성 단계에서 마스터 크리덴셜 'Km'과 이에 대응하는 마스터 크리덴셜 식별값 'KmId'는 엔티티 A와 MAF에 프리-프로비저닝 되어 있거나 RSPF를 이용하여 원격 구성 되어 있다. 협약 구성단계에서 엔티티 A, 엔티티 B에는 협약 보안 핸드쉐이크 과정에서 인증과 식별에 사용되는 정보가 설정된다. 이 과정에서 엔티티A는 엔티티 B의 식별값 'IdB'를 알아야 하며, 엔티티 B는 엔티티 A의 식별값 'IdA'를 알아야 한다. 또한 MAF 는 엔티티 B의 식별값 'IdB'를 알아야 하며, IdB는 IN

식별 값에 해당한다. 협약 보안 핸드쉐이크 과정 중에서 MAF 기반 SAEF의 핸드쉐이크는 M2M 보안 커넥션 키 'Kc'의 수립(Establishment)을 가능하게 한다. "psk_identity" 파라미터는 마스터 크리덴셜 식별자 값, KmId로 설정되고, "psk" 파라미터는 마스터 크리덴셜 값 Km으로 설정된다. 엔티티 A는 엔티티 B(인프라 노드)에 KcId 값을 전송한다. 엔티티 B는 MAF에서 M2M 보안 커넥션 키인 Kc를 검색 할 수 있다.

# 6. 사물들을 제어하고 관리하는 방법

### 관련 oneM2M 표준 문서

TR-0006: Study of Management Capability Enablement
TS-0005: Management Enablement (OMA)
TS-0006: Management Enablement (BBF)

표준에서는 물리적인 디바이스보다는 논리적이 엔티티인 노드를 기준으로 기술되어 있으나, 노드는 설치시 노드의 종류에 따라 ADN, ASN는 M2M 디바이스, MN는 M2M 게이트웨이, IN는 M2M 서버로 물리적으로 매핑되게 된다. oneM2M에서는 이러한 노드들이 매핑된 물리적 디바이스를 관리하고자 디바이스 관리 기술을 기술하고 있다. oneM2M에서는 M2M 디바이스 관리 기술을 직접 개발하기 보다는, 널리 상용화되어 있고 기술 완성도가 높은 기존 디바이스 관리 기술들을 사용하기로 하고, 이러한 기존 디바이스 관리 기술과의 연동에 중점을 맞추어 기술하였다. 이렇게 선정된 기존 디바이스 관리 기술들은 OMA DM 1.3[7], 2.0[8],OMA Lightweight M2M[9], BBF TR-069[10]과 같다.

이에 oneM2M에서 정의된 디바이스 관리 기술과의 연동 기법을 살펴보기 이전에 기존의 디바이스 관리 기술에 대해서 살펴보고, 기존의 디바이스 관리 기술과 oneM2M과의 연동 방법, oneM2M에서 지원하는 디바이스 관리 기술에 대해서 소개한다.

## 6.1. 기존 디바이스 관리 기술 소개

### 6.1.1. OMA DM

OMA(Open Mobile Alliance)의 DM(Device Management) 기술은 이동통신사에서 휴대폰을 원격 제어/관리하고자 하는 목적에서 개발되었다. 2002년 6월부터 표준 개발이 진행되어 현재까지 10년 이상의 디바이스 관리 기술이 축적되어 있다. OMA DM 기술은 2012년 기준으로 14억 단말에 탑재되어 있을 만큼 디바이스 관리 분야에서 널리 쓰이고 있다. 1.X대 버전은 1.1, 1.2, 1.3으로 진행되었으며, 버전 1.3까지 하위 호환성(backward compatibility)를 제공하고 있다. 가장 최근에 개발된 버전 2.0은

7) OMA Device Management Protocol Version 1.3, http://openmobilealliance.org/
8) OMA Device Management Protocol Version 2.0, http://openmobilealliance.org/
9) OMA Lightweight Machine to Machine Version 1.0, http://openmobilealliance.org/
10) BBF TR-069, CPE WAN Management Protocol Issue 1 Amendment 4, http://www.broadband-forum.org

1.3으로 이어지며 호환성을 위해 발생한 복잡도와 쓰이지 않는 기술을 삭제, 프로토콜을 간소화하여 개발되었다. 그림 〈6-1〉 와 같이 OMA DM 서버는 디바이스 내에 존재하는 DM 클라이언트와 바로 연결될 수도 있으며, DM 게이트웨이를 통해 연결 될 수 있다.

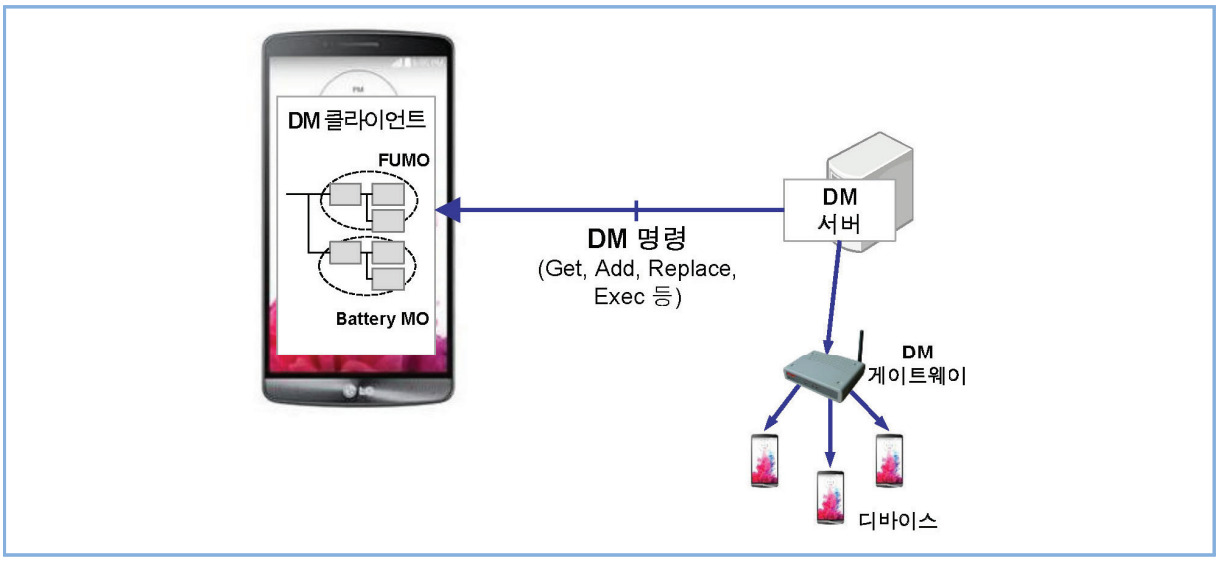



그림 〈6-1〉 OMA DM 디바이스 관리 기술 구조

DM 서버에서 DM 클라이언트에 DM 트리의 자원들을 DM 명령를 통해 조작함으로써 디바이스 관리 서비스가 가능하다. DM 트리에는 MO(Management Object) 단위로 자원들이 존재하며, 각 MO는 다른 디바이스 관리 기능을 제공한다. OMA DM에서 정의한 대표적인 MO는 표 〈6-1〉와 같다.

표 〈6-1〉 OMA DM 대표 MO 기능 설명

| MO | 기능 |
|---|---|
| stdMO | DM 서버 계정 정보, 디바이스 기본 정보 제공 |
| SCOMO | 소프트웨어 설치, 제거 및 관리 |
| FUMO | 펌웨어 업데이트 |
| DCMO | USB, 카메라 등 주변 디바이스 관리 |
| DiagMon | 모니터링(베터리 레벨, App 사용 정보, RF 시그널 상태) 및 진단 |
| GwMO | 게이트웨이 특화 기능 (Fanout 등) |

OMA DM은 디바이스 관리 프로토콜과 MO는 독립적으로 버전을 할당하여 개발하였다. 이를 통해 디바이스 관리 프로토콜이 변경되더라도 기존의 MO에 영향을 주지 않으며, MO가 변경되더라도 디바이스 관리 프로토콜에는 영향을 주지 않아 신규 MO 또는 신규 프로토콜이 정의되더라도 적용이 용이하다. 원래 휴대폰을 관리하고자 하는 목적에서

개발되었으나, 최근에는 휴대폰에 한정되지 않고, 가전 디바이스, M2M 디바이스를 관리하기 위해 게이트웨이 MO, 센서 측정 관련 MO를 정의하는 그 영역을 확대하고 있다.

### 6.1.2. BBF TR-069

BBF(Broadband Forum)의 TR-069(Technical Report-069)는 ACS(Auto Configuration Server)와 CPE(Customer-Premises Equipment)간의 프로토콜인 CWMP(CPE WAN Management Protocol)를 정의하고 있다. 쉽게 설명하면, CPE는 유선으로 연결된 디바이스 (예: 전화, 셋탑박스, 홈 게이트웨이), ACS는 CPE 디바이스를 관리하는 서버, CWMP는 유선 상에서의 디바이스 관리 프로토콜로 생각하면 된다.(그림 〈6-2〉) OMA DM이 무선상의 휴대폰 디바이스 관리에 초점을 맞추었다면, BBF TR-069의 경우 유선상의 디바이스 관리에 초점이 맞추어 개발되었다. 2004년 5월에 버전 1.0이 개발되었으며, 현재 버전 1.4까지 개발되어 있다. CPE Proxier는 Proxied 디바이스와 ACS간을 연결하는 게이트웨이로써, CPE Proxier와 ACS간에는 CWMP가 사용되며, CPE Proxier와 Proxied 디바이스 사이에는 지그비, 블루투스 등의 PAN(Personal Area Network) 프로토콜을 사용할 수 있다.

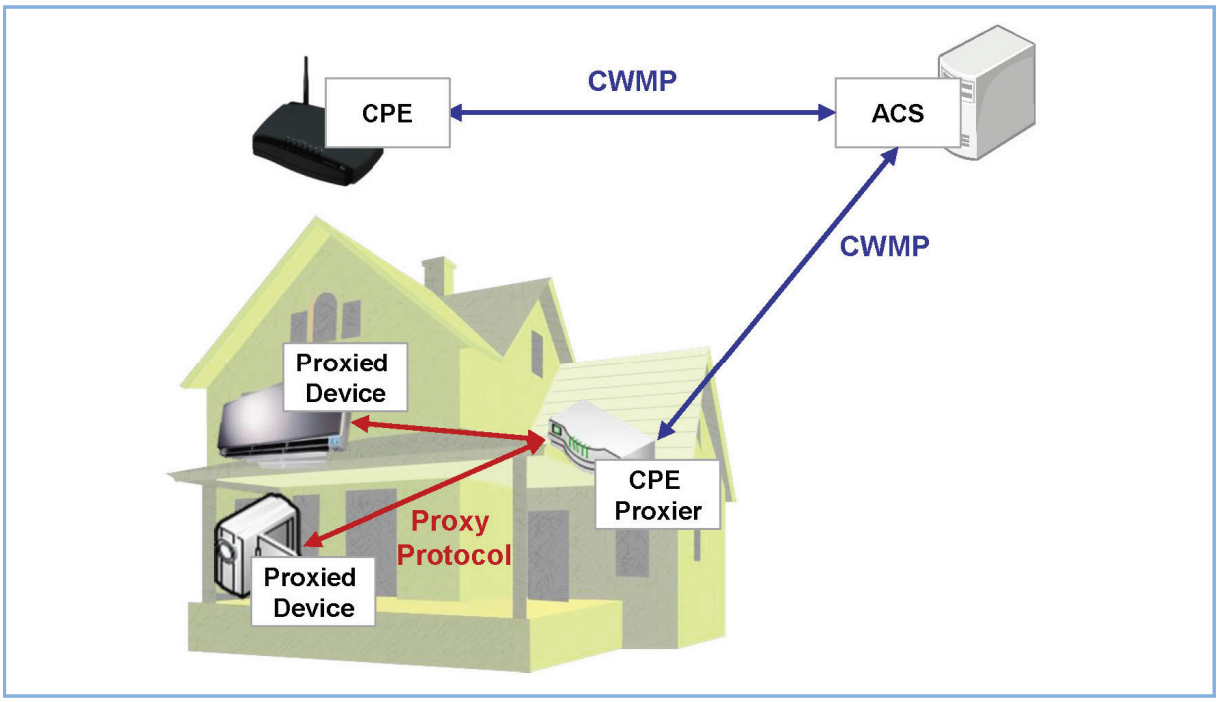



그림 〈6-2〉 BBF TR-069 장치 관리 기술 구조

CWMP는 SOAP/HTTP 기반의 프로토콜로써, RPC(Remote Procedure Call)을 통해서 디바이스 관리 기능을 수행한다. 이에 BBF TR-069에서는 RPC Method와 Method별 Argument들을 정의하고 있다. RPC Method는 ACS에서 CPE로의 Method, CPE

에서 ACS로의 Method로 구분되어 있다. Method를 모두 설명할 수는 없어 간략히 표 〈6-2〉에 몇몇 Method만을 설명한다.

표 〈6-2〉 BBF TR-069 Method 설명

| Method | 기능 |
|---|---|
| GetRPCMethods | CPE, ACS에서 지원하는 RPC Method 검색 |
| SetParameterValues | ACS에서 CPE의 파라미터를 설정 |
| GetParameterValues | ACS에서 CPE의 파라미터를 획득 |
| Download | CPE에서 파일을 다운받도록 함 |
| Reboot | CPE를 리부팅 함 |
| FactoryReset | CPE를 공장 초기화 함 |
| Inform | CPE는 ACS와 세션을 맺으면 Inform 전송 |

### 6.1.3. OMA Lightweight M2M

OMA(Open Mobile Alliance)의 Lightweight M2M(이하 LWM2M) 기술은 가장 최근에 개발된 디바이스 관리 기술로써, M2M 디바이스 관리를 목적으로 개발되었다. 2011년부터 개발을 시작하여 2013년 12월 버전 1.0을 배포하였다. 자원 제약적인 M2M 디바이스(예: 8-16비트 MCU, 수십 KB RAM, 수백 KB 플래시 메모리)를 관리하기 위한 목적으로 개발되었기에 프로토콜이 아주 단순하게 구성되어 있다.

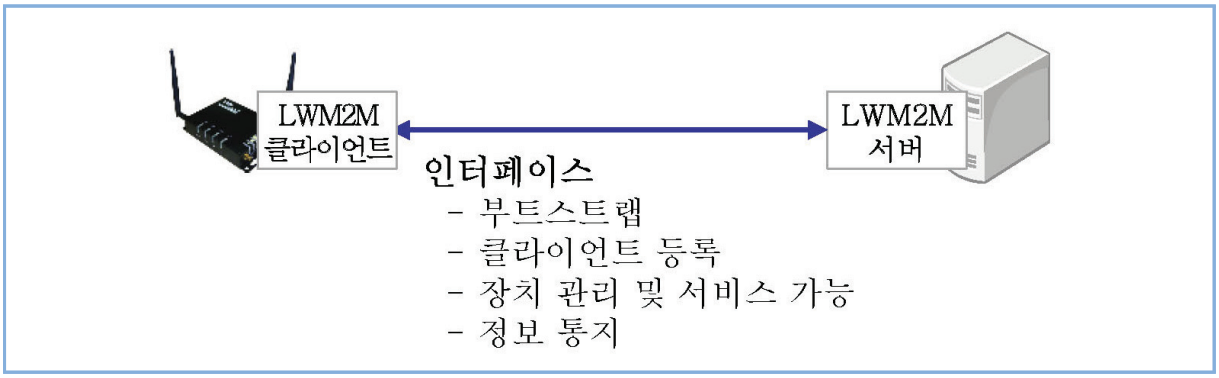



그림 〈6-3〉 OMA Ligthweight M2M 장치 관리 기술 구조

총 4개의 인터페이스를 통해 LWM2M 서버와 LWM2M 클라이언트는 통신을 할 수 있다. 부트스트랩 인터페이스를 통해 LWM2M 클라이언트가 LWM2M 서버와 통신을 할 수 있는 설정을 하고, 디바이스에 파워가 들어오면 LWM2M 클라이언트는 LWM2M 서버에 클라이언트 등록을 수행한다. 등록 이후에는 LWM2M 서버는 디바이스 관리 및

서비스 가능 인터페이스를 통해 LWM2M 클라이언트에 디바이스 관리 및 M2M 서비스를 수행할 수 있으며, 정보 통지 인터페이스를 통해 LWM2M 클라이언트는 변경이 발생한 이벤트를 LWM2M 서버에 전송할 수 있다.

CoAP을 바인딩 프로토콜로 사용하고 있으며, CoAP을 UDP 또는 SMS를 통해 전송 가능하다. 바이너리 TLV(Tag, Length, Value)와 JSON으로 자원을 표현하여 페이로드(Payload)의 부하를 줄였으며, 트리구조의 레벨로 자원 구조가 표현된다. LWM2M에서는 오브젝트(Object)라는 개념으로 자원을 구조화하였는데, 대표적인 오브젝트는 표 〈6-3〉과 같다.

표 〈6-3〉 OMA Lightweight M2M 대표 오브젝트 기능 설명

| Object | 기능 |
|---|---|
| Server | LWM2M 서버 계정 정보 |
| Access Control | 서버 별 자원의 접근 가능 명령 정의 |
| Device | 디바이스 설정 정보, 공장 초기화, 리부팅 |
| Connectivity | 디바이스 네트워크 연결 정보 |
| Location | 디바이스 위치 정보 |
| Firmware Update | 펌웨어 업데이트 |
| Software | 소프트웨어 설치/제거 |

## 6.2. oneM2M 연동 방법

oneM2M에서는 〈mgmtObj〉와 〈mgmtCmd〉 자원 타입을 통해서 기존 디바이스 관리 기술과의 연동이 가능하다. 기존 디바이스 관리 기술과의 연동에 있어, oneM2M에서 가장 고려한 디자인 원칙은 어떠한 기존 디바이스 관리 기술을 사용하였는지 oneM2M 서비스 계층에서 모르더라도 디바이스 관리가 가능하도록 하는 것이었다. 즉, oneM2M 서비스 계층의 어플리케이션 엔티티(AE)는 OMA DM/BBF TR-069/OMA Lightweight M2M의 기술을 전혀 알지 못하더라도 디바이스 관리가 가능하다. 이를 위해 그림 〈6-4〉와 같이 IN-CSE는 Mca 레퍼런스 포인트를 통해 디바이스 관리 기술과 무관한 Uniform한 인터페이스를 제공하고, IN-CSE는 어플리케이션 엔티티는 기존 디바이스 기술과 독립적인/무관한 지네릭(Generic)한 명령을 전달 가능하다. 명령을 받은 IN-CSE는 명령의 타깃 노드가 지원하는 디바이스 관리 기술로 해당 명령을 변형하여 전달한다.

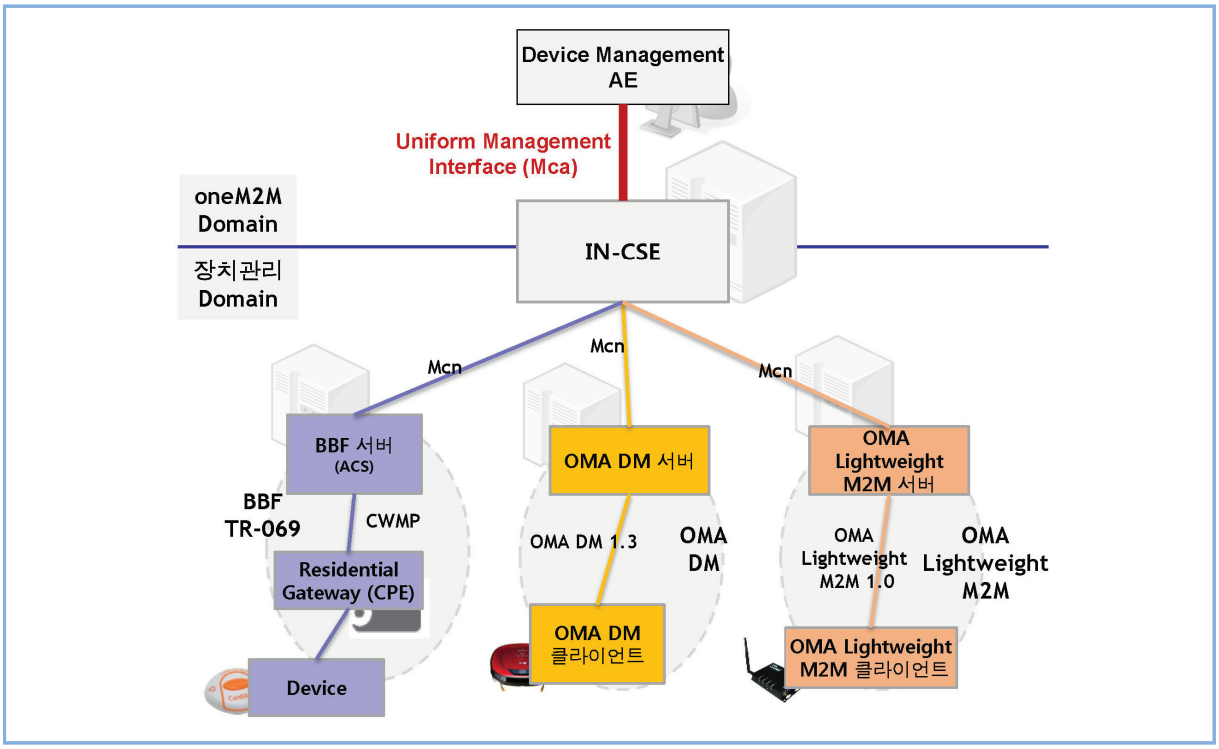



그림 〈6-4〉 oneM2M 서비스 계층과 기존 장치 관리 기술 연동

### 6.2.1. 〈mgmtObj〉를 통한 연동 방법

oneM2M에서는 〈mgmtObj〉 자원 타입을 통해서 기존 디바이스 관리 기술과의 연동이 가능하다. 〈mgmtObj〉 자원 타입은 타 자원 타입들과는 다르게 탬플릿(template)과 같은 개념으로, 실제 자원 생성 시에는 탬플릿에 기반하여 정의된 디바이스 관리 자원 타입에 따라 생성된다.

먼저 〈mgmtObj〉 자원 타입에 대해서 살펴본다. 〈mgmtObj〉 자원 타입의 자원 타입 특화(resource type specific) 속성들은 표 〈6-4〉와 같다.

표 〈6-4〉 〈mgmtObj〉 자원 타입 특화 속성

| 속성 이름 | 설명 |
|---|---|
| mgmtDefinition | 디바이스 관리 자원 타입 이름 (예: software, firmware, memory) |
| objectIDs | 디바이스 관리 자원 타입에 매칭되는 기존 디바이스 관리 기술의 자원 타입 (예: OMA DM MO)의 URN |
| objectPaths | 기존 디바이스 관리 기술의 자원 경로 |
| mgmtLink | 디바이스 관리 자원 타입에서 타 디바이스 관리 자원 타입을 참조할 때 사용 |
| [objectAttribute] | mgmtDefinition 속성에 명시된 디바이스 관리 자원 타입에 해당하는 속성 |
| description | 디바이스 관리 자원 타입에 대한 설명 |

앞에서 설명하였듯이 〈mgmtObj〉 자원 타입은 탬플릿으로써 디바이스 관리 자원 타입에 따라 [objectAttribute]의 속성들이 달라지게 된다. 실제 예를 보면 쉽게 이해되는데, 그림 〈6-5〉를 보면, [firmware] 디바이스 관리 자원 타입은 〈mgmtObj〉의 공통 속성들을 사용하면서 [firmware] 디바이스 관리 자원 타입에 맞는 속성들이 [objectAttribute] 영역에 확장되어 있음을 알 수 있다. 동일하게 [memory] 디바이스 관리 자원 타입은 〈mgmtObj〉의 공통 속성들을 사용하면서 [memory] 디바이스 관리 자원 타입에 맞는 속성들이 [objectAttribute] 영역에 확장되어 있음을 알 수 있다.

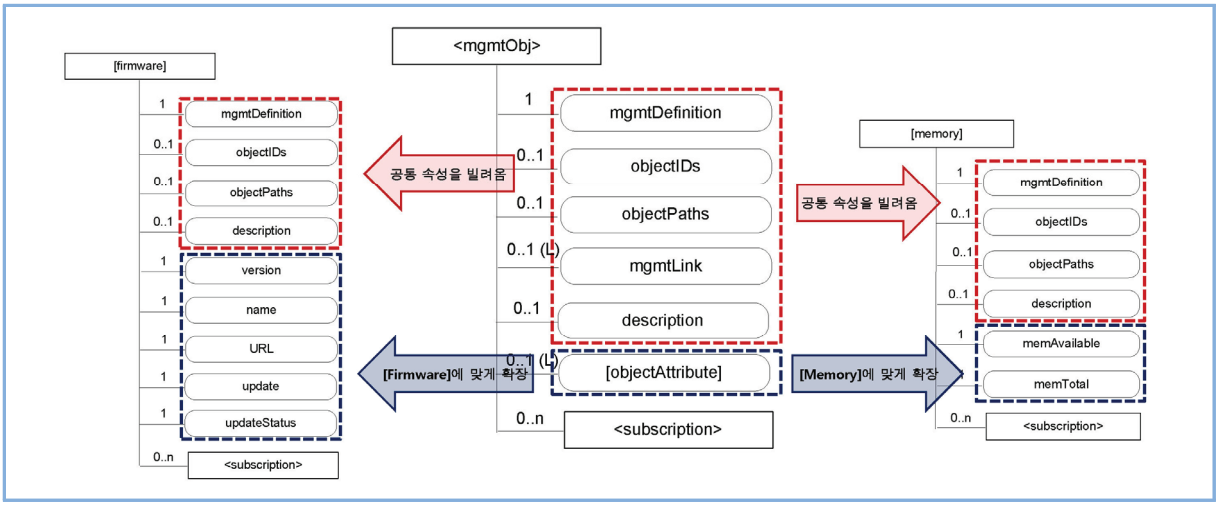



그림 〈6-5〉 〈mgmtObj〉를 통한 장치 관리 자원 타입 구조화

oneM2M에서는 총 12개의 디바이스 관리 자원 타입을 정의하였는데, 타입 별로 각기 다른 디바이스 관리 기능을 제공한다. oneM2M에서 정의한 디바이스 관리 자원 타입은 표 〈6-5〉와 같다.

표 〈6-5〉 oneM2M 지원 디바이스 관리 서비스

| 디바이스 관리 자원 타입 | 디바이스 관리 서비스 |
|---|---|
| firmware | 펌웨어 업데이트 |
| software | 소프트웨어 설치, 제거 등 관리 |
| memory | 메모리 관련 모니터링 (메모리 잔량. 전체 메모리 크기) |
| areaNwkInfo, areaNwkDeviceInfo | M2M Area Network 관련 정보 제공<br>(M2M Area Network에 존재하는 디바이스의 목록 등) |
| battery | 배터리 관련 모니터링 (배터리 레벨, 배터리 상태) |
| deviceInfo | 디바이스 기본 정보 제공<br>(디바이스 설명, 제조사, 모델명, 펌웨어, 하드웨어 버전 등) |
| deviceCapability | 디바이스의 주변장치 (USB, 카메라 등) 컨트롤 (켜기/끄기 등) |
| reboot | 리부팅, 공장초기화 수행 |
| eventLog | 보안, 시스템 로그 등 모니터링 |
| cmdhPolicy | 데이터 전달과 관련된 정책 설정 |

그림 〈6-6〉은 〈mgmtObj〉를 통한 동작 원리를 설명한다. IN-AE는 IN-CSE에 〈mgmtObj〉 자원 타입을 조작(생성/갱신/획득/삭제)한다. IN-CSE의 디바이스 관리 어댑터(Management Adaptor)는 oneM2M 조작 명령을 기존 디바이스 관리 명령으로 변환하여 디바이스 관리 서버(DM Server)에 전송한다. 디바이스 관리 서버는 해당 명령을 디바이스 관리 클라이언트(DM Client)에 전달하여 디바이스 관리 기능을 수행하고, 수행 결과를 디바이스 관리 서버에 전송하면, 디바이스 관리 서버는 이를 디바이스 관리 어댑터에 전송하여 디바이스 관리 응답을 oneM2M 응답으로 변환하여 IN-AE에게 전달한다.

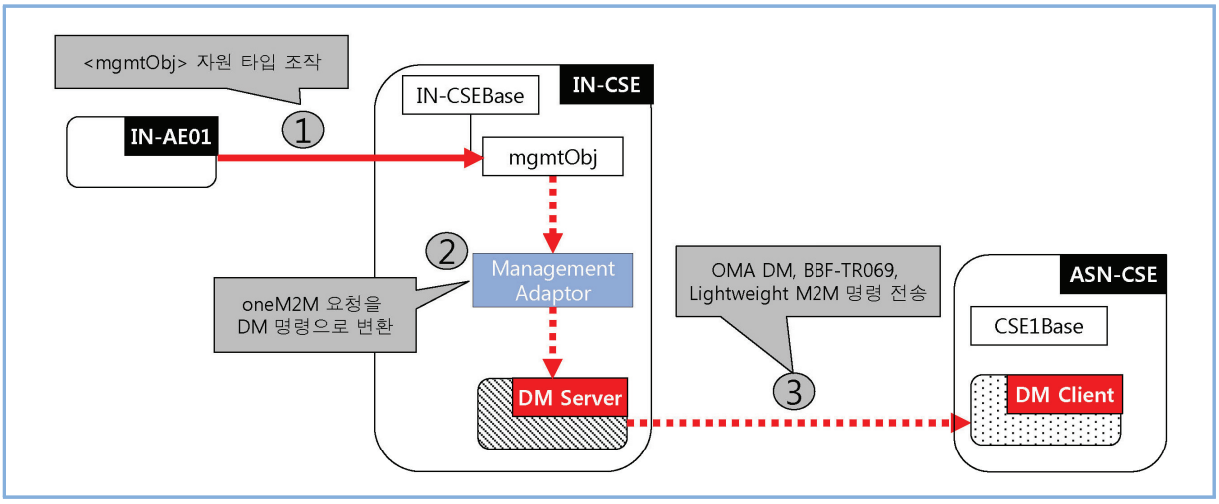



그림 〈6-6〉 〈mgmtObj〉를 통한 기존 장치 관리 기술 연동

여기서 디바이스 관리 어댑터의 역할은 oneM2M 조작 명령을 기존 디바이스 관리 기술 명령으로 변환하는 기능, 디바이스 관리 서버와 요청 및 응답 전송, 디바이스가 지원하는 디바이스 관리 기술(OMA DM, BBF TR069, OMA Lightweight M2M)에 맞는 디바이스 관리 서버 선택, 디바이스 관리 자원 검색이 있다. 그림 〈6-7〉은 IN-AE로부터 전달된 oneM2M 명령을 DM 1.3 명령으로 변환하는 디바이스 관리 어댑터 기능의 예를 보여준다.

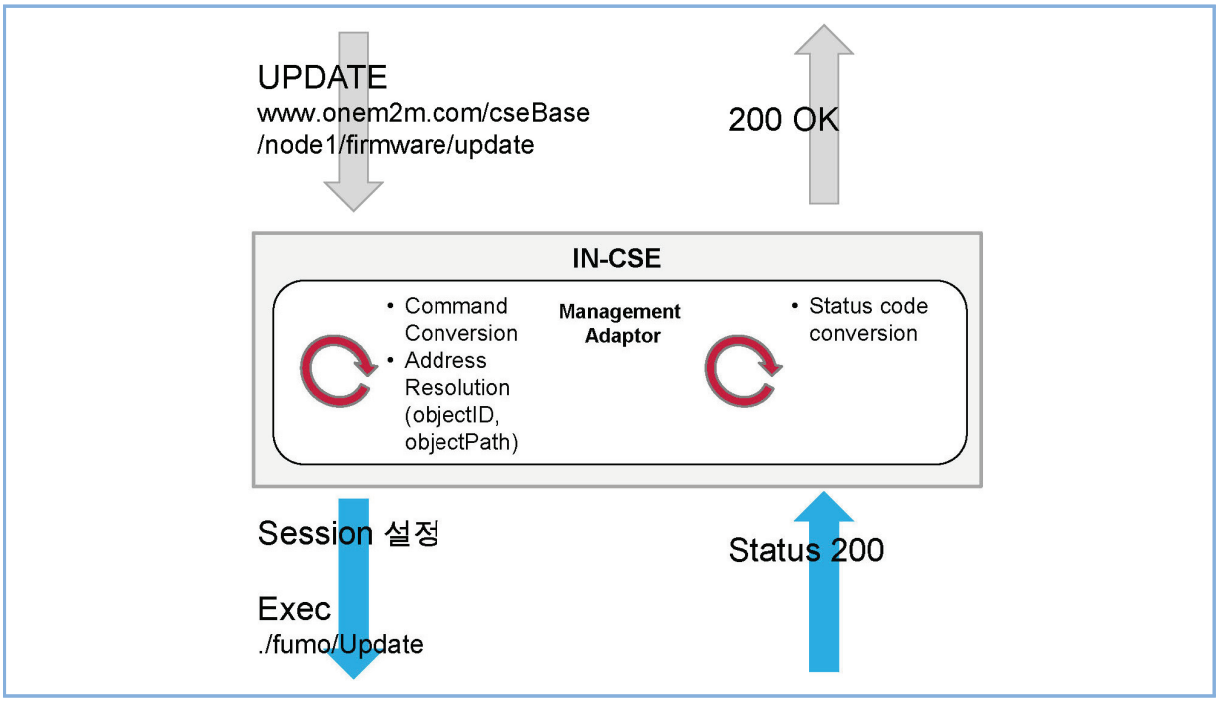



그림 〈6-7〉 oneM2M 명령을 OMA DM 1.3으로 변경 예제

### 6.2.2. 〈mgmtCmd〉를 통한 연동 방법

oneM2M에서는 〈mgmtCmd〉 자원 타입을 통해서 기존 디바이스 관리 기술과의 연동이 가능하다. 기술적으로는 oneM2M에서 지원하는 기존 디바이스 관리 기술 모두와 연동이 가능하지만, RPC(Remote Procedure Call)을 사용하는 BBF TR-069에 특화된 자원 타입이라 볼 수 있다. 〈mgmtCmd〉 자원 타입의 자원 타입 특화(resource type specific) 속성들은 표 〈6-6〉와 같다.

표 〈6-6〉 〈mgmtCmd〉 자원 타입 특화 속성

| 속성 이름 | 설명 |
|---|---|
| cmdType | 디바이스 관리 명령 (예: firmware download) |
| execReqArgs | 디바이스 관리 명령의 파라미터 (예: firmware download URI) |
| execEnable | 디바이스 관리 명령 수행 |
| execTarget | 디바이스 관리 명령의 타깃 정보 저장<br>(예: 타깃 노드의 노드 자원 또는 타깃 노드들의 〈group〉 자원) |
| execMode | 디바이스 관리 명령 수행 모드 설정<br>(예: 한번 수행, 반복적으로 수행, 랜덤하게 수행) |
| execFrequency | 디바이스 관리 명령간의 최소한의 시간 간격<br>(반복적으로 수행되는 execMode에서 사용) |
| execDelay | 디바이스 관리 명령간의 Delay 시간<br>(랜덤하게 수행되는 execMode에서 사용) |
| execNumber | 디바이스 관리 명령 수행 횟수 |

〈mgmtCmd〉 자원 타입의 경우 최소 두 번의 oneM2M 조작이 있어야 수행된다. 첫 명령에서는 〈mgmtCmd〉 자원 타입의 자원을 생성하고, 두 번째 명령에서는 execEnable 속성에 갱신 조작 명령을 전송하면 첫 명령에서 설정된 디바이스 관리 기술 명령이 기존 디바이스 관리 기술로 변환하여 전송된다. execEnable 속성에 갱신 조작 명령을 전송하면, 〈mgmtCmd〉 자원의 자식 자원으로 〈execInstance〉 자원 타입의 자원이 생성되는데, 해당 자원은 전송한 디바이스 관리 기술 명령의 결과를 저장하고 있다. 〈execInstance〉 자원 타입 특화(resource type specific) 속성들은 표 〈6-7〉과 같다.

**표 〈6-7〉 〈execInstance〉 자원 타입 특화 속성**

| 속성 이름 | 설명 |
|---|---|
| execStatus | 디바이스 관리 실행 상태 (예: 실행 시작, 실행 완료, 취소됨) |
| execResult | 디바이스 관리 실행 결과 |
| execDisable | 디바이스 관리 실행 취소 (갱신 조작 명령 전달 시 실행 취소 됨) |
| execTarget | 실행한 디바이스 관리 명령의 타깃 정보 저장<br>(예: 타깃 노드의 노드 자원 또는 타깃 노드들의 〈group〉 자원) |
| execMode | 실행한 디바이스 관리 명령 수행 모드 설정<br>(예: 한번 수행, 반복적으로 수행, 랜덤하게 수행) |
| execFrequency | 실행한 디바이스 관리 명령간의 최소한의 시간 간격<br>(반복적으로 수행되는 execMode에서 사용) |
| execDelay | 실행한 디바이스 관리 명령간의 Delay 시간<br>(랜덤하게 수행되는 execMode에서 사용) |
| execNumber | 실행한 디바이스 관리 명령 수행 횟수 |
| execReqArgs | 실행한 디바이스 관리 명령의 파라미터(예: firmware download URI) |

그림 〈6-8〉는 〈mgmtCmd〉를 통한 동작 원리를 설명한다. IN-AE는 IN-CSE에 〈mgmtCmd〉 자원 타입의 자원을 조작(생성/갱신/획득/삭제)한다. 특히 전달할 디바이스 관리 명령을 cmdType에 설정하고, 해당 명령의 파라미터를 execReqArgs에 설정, 디바이스 관리 명령의 타깃을 execTarget에 설정한다. 이후 execEnable에 oneM2M 갱신 명령을 전송한다. IN-CSE의 디바이스 관리 어댑터(Management Adaptor)는 oneM2M 조작 명령을 기존 디바이스 관리 명령으로 변환하여 디바이스 관리 서버(DM Server)에 전송한다. 디바이스 관리 서버는 해당 명령을 디바이스 관리 클라이언트에 전달하여 디바이스 관리 기능이 수행되고, 수행 결과를 디바이스 관리 서버에 전송하면, 디바이스 관리 서버는 이를 디바이스 관리 어댑터에 전송하고 〈execInstance〉 자원 타입의 자원을 생성하여 디바이스 관리 명령 수행 결과를 IN-AE에게 제공한다.

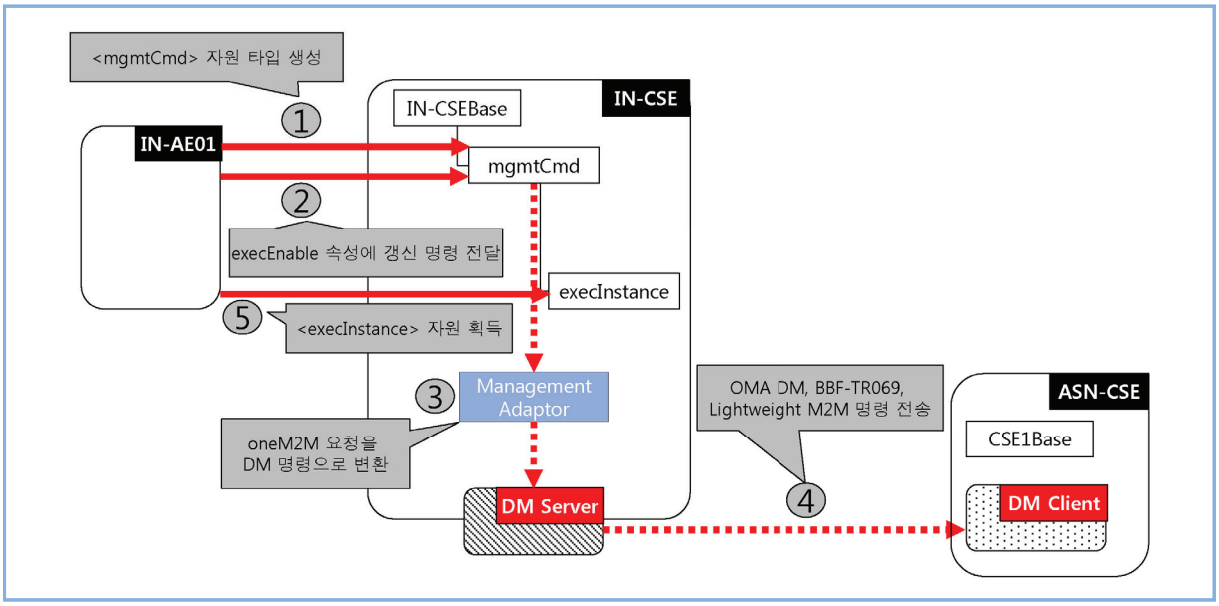



그림 〈6-8〉 〈mgmtCmd〉를 통한 기존 장치 관리 기술 연동

## 6.3. oneM2M과 기존 디바이스 관리 기술 매핑

oneM2M에서는 oneM2M과 기존 디바이스 관리 기술 연동에 있어 데이터 타입, 식별자, 자원, 명령, 응답 코드 연동 등에 대해서 TS-0005와 TS-0006에 상세히 정의해 놓았다. oneM2M과 기존 디바이스 관리 기술을 연동하고자 할 시에는 해당 문서를 참고하여 개발을 진행하면 된다.

## 6.4. oneM2M 계층 디바이스 관리 기술

6.1, 6.2, 6.3장을 통해서 oneM2M 계층과 기존 디바이스 관리 기술간의 연동에 대해서 알아보았다. 이러한 기존 디바이스 관리 기술과의 연동은 종래 디바이스 관리 기술을 보유하고 있는 M2M 서비스 제공자에게는 상당한 이점이 있으나, 종래 디바이스 관리 기술을 보유하지 않은 신규 M2M 서비스 제공자에게는 oneM2M 계층 개발은 물론 종래 디바이스 관리 기술 개발의 이중고가 발생한다. 이에 oneM2M 표준에서는 종래 디바이스 관리 기술 연동 및 oneM2M 계층에서의 디바이스 관리 기술 또한 제공한다.

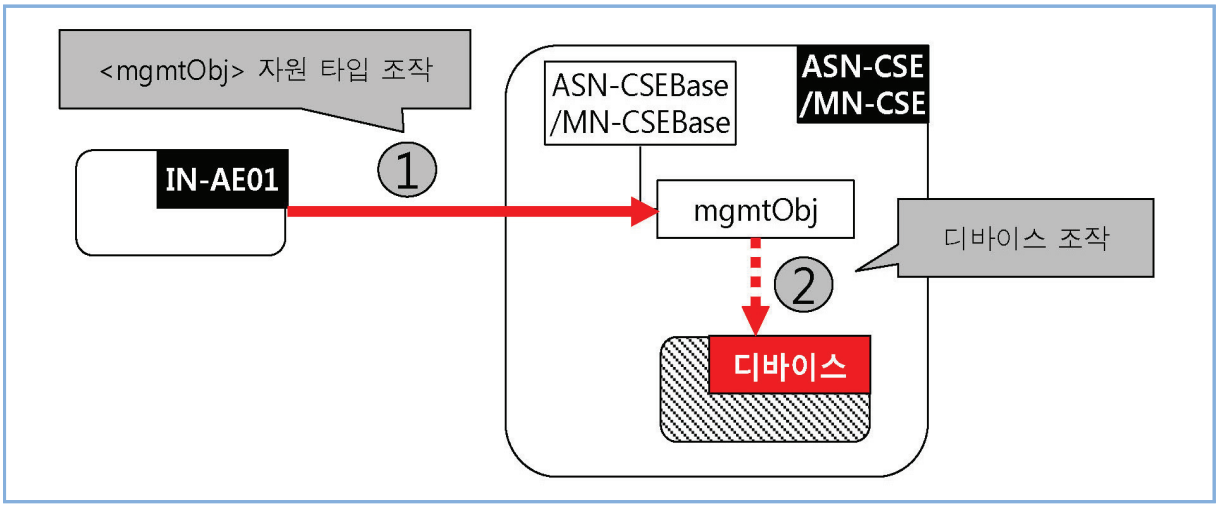



그림 〈6-9〉 oneM2M 계층 디바이스 관리 기술 동작 원리

M2M 서비스 제공자는 두 가지 방식 중 선호하는 방식을 선택하여 디바이스 관리를 수행하면 된다. 기존의 디바이스 관리 기술과의 연동은 IN-CSE의 〈mgmtObj〉 또는 〈mgmtCmd〉 자원 조작을 통해서 동작하였다면, oneM2M 계층 디바이스 관리 기술은 ASN-CSE, MN-CSE에 〈mgmtObj〉 자원을 조작함으로써 가능하다. 그림 〈6-9〉는 oneM2M 계층 디바이스 관리 기술 동작 원리를 보여준다. ASN-CSE 또는 MN-CSE에 〈mgmtObj〉 자원의 조작 요청을 전송하면, ASN-CSE 또는 MN-CSE는 〈mgmtObj〉 조작에 맞게 자체 디바이스의 연동을 수행함으로써 디바이스 관리가 가능하다.

## 6.5. 요약

본 장을 통해 기존 디바이스 관리 기술인 OMA DM, OMA Lightweight M2M, BBF TR-069를 소개하였고, 〈mgmtObj〉, 〈mgmtCmd〉 자원 타입을 통한 oneM2M 플랫폼과 기존 디바이스 관리 기술간의 연동 방법에 대해서 살펴보았다. 이에 덧붙여 기존 디바이스 관리 기술을 사용하지 않고 디바이스 관리를 하는 방법을 제공하고 있다. 이렇듯 oneM2M에서는 다양한 환경, 구성에서 디바이스를 관리할 수 있는 방법들을 지원하고 있으며, 이를 통해 M2M 서비스 공급자의 환경, 구성에 맞는 디바이스 관리가 가능하다.

# 7. 의미로 이해하고 행동하는 사물 인터넷

## 관련 oneM2M 표준 문서

TR-0007: Study of Abstraction and Semantic Enablements
TS-0002: oneM2M Requirements

사물 인터넷은 사물간의 통신 기능을 이용하여 사람의 개입 없이도 서비스를 지능적으로 처리할 수 있는 다양한 응용을 고려하고 있다. 즉, 의미로 이해하고 행동하는 사물 인터넷이 실현 가능하도록 하기 위하여 핵심 기술로 추상화(Abstraction) 및 시맨틱(Semantics) 기술이 고려되고 있다.

oneM2M에서 개발 중인 추상화 및 시맨틱 기술 보고서(Technical Report "Study of abstraction and semantic enablements", TR ABS)는 모니터링 및 제어 기능을 지원할 때 서로 다른 유형의 디바이스 간의 연동을 고려하는 oneM2M 추상화 능력과 oneM2M 엔티티와 데이터에 시맨틱 정보를 주기 위한 핵심 능력, 그리고 이를 활용한 디바이스 발견, 가상 디바이스 생성 등에 관한 핵심 기술을 소개하고 있다.

지금까지 다양한 형태의 시맨틱 관련 유즈 케이스를 발굴하고 이로부터 도출한 상세 요구사항을 분석하고 있으며, 이를 바탕으로 oneM2M 구조와 연계된 시맨틱 지원을 위한 구조를 담은 oneM2M 스펙을 개발 해 나갈 계획이다. 특히 시맨틱 기술과 관련해서는 주요 기능 항목에 대한 명세 및 세부 프로토콜 개발이 필요하다.

초기 oneM2M TR ABS의 작업 범주는 oneM2M 추상화 능력과 oneM2M 엔티티와 데이터에 대한 시맨틱 표현을 지원하는 oneM2M 능력에 초점을 맞추었다. 이를 통한 디바이스, 가상 디바이스, Things를 발견하기 수단 등에 대한 핵심 기술의 최근 현황 및 솔루션을 조사한 후에 추상화 및 시맨틱 측면에서 oneM2M 요구사항과 일치하는 기술 및 솔루션을 분석하고 oneM2M 구조 및 프로토콜에 의해 이러한 솔루션의 일부분을 어떻게 활용할 수 있을 지에 대한 분석하고자하는 것이었다. oneM2M 릴리즈 1을 통하여 TR ABS 개발을 1차로 완료한 상태이며, 릴리즈 2를 위하여 기존 TR에 좀 더 상세한 내용을 보완하는 작업이 MAS (관리, 추상화 및 시맨틱) 워킹 그룹에서 진행 중에 있다.

본 장에서는 추상화와 시맨틱 기술에 대한 핵심 개념을 살펴보고, 관련 기술을 지원하기 위한 요구사항 및 주요 기능, 주요 표준화 기구에서 진행된 관련 표준을 살펴 본 후에 oneM2M에서 본 기술을 적용하기 위해 필요한 모데링 기법 및 구조를 잡는데 있어 핵심적으로 고려해야 할 사안을 소개한다.

## 7.1. 개요

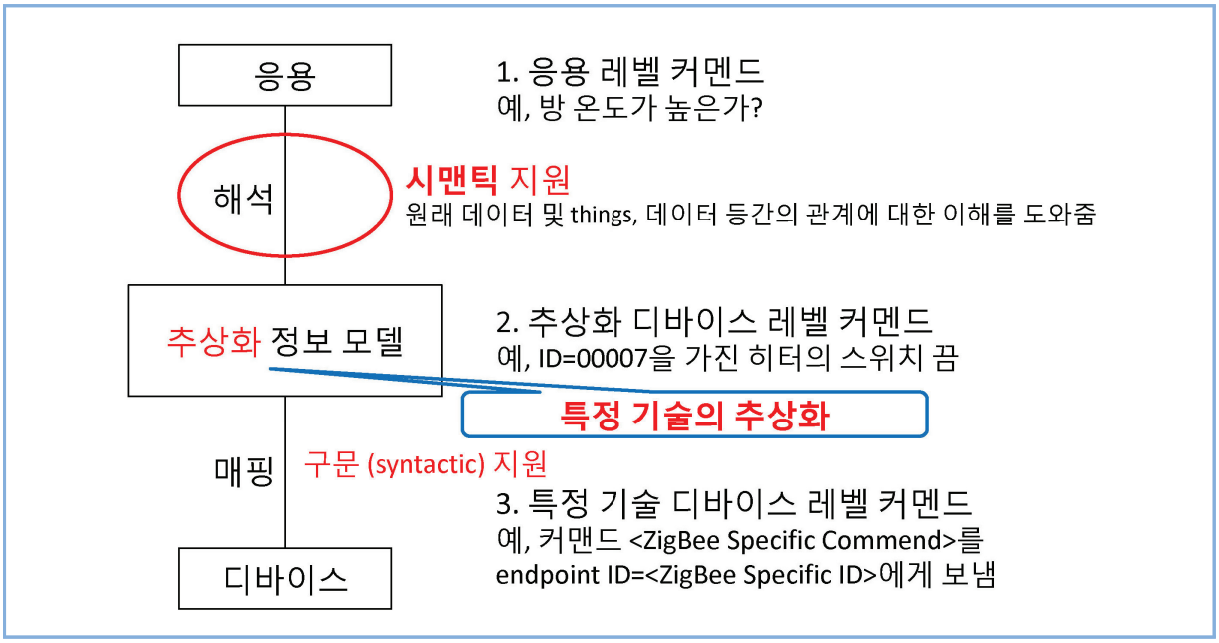



그림 〈7-1〉 추상화 및 시맨틱 개념

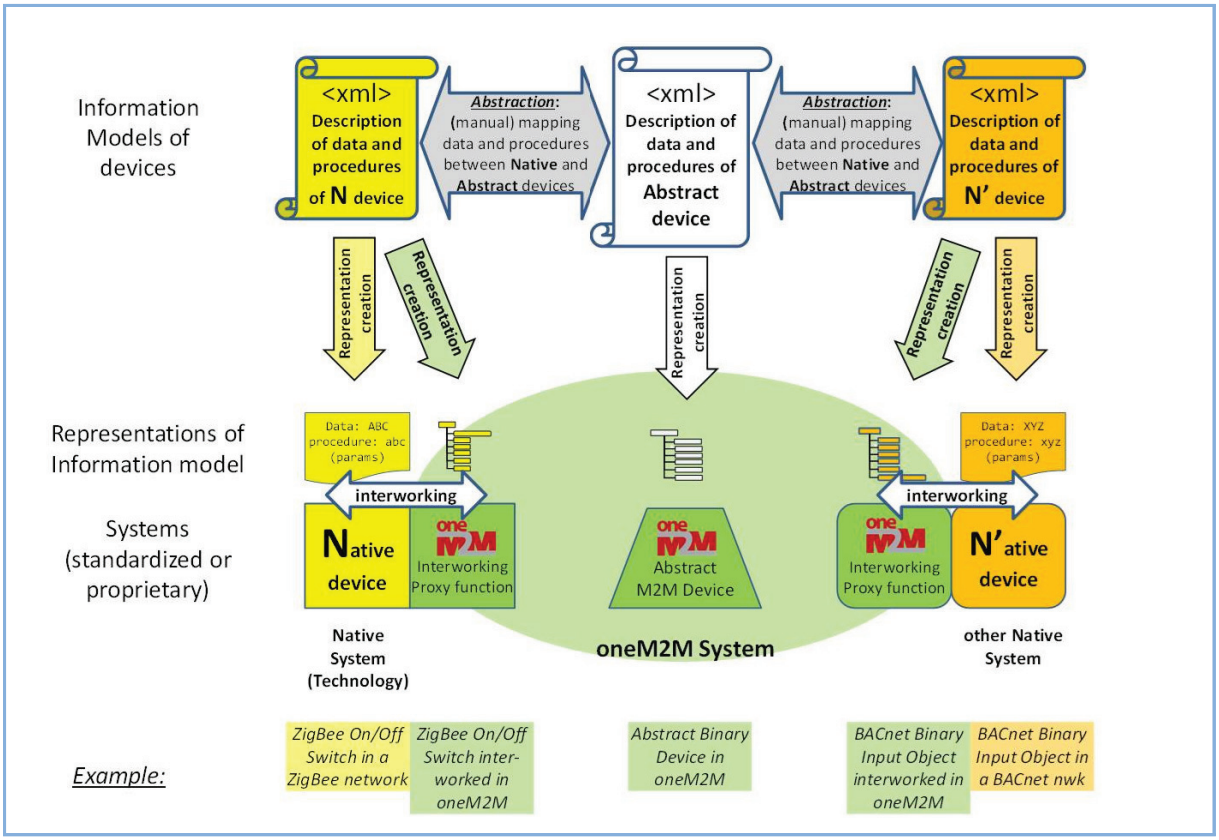



그림 〈7-2〉 인터워킹 및 추상화

현재 oneM2M에서 개발하고 있는 TR ABS에는 그림 〈7-1〉과 같이 단계별로 사물에 의미를 부가하여 좀 더 지능화된 서비스를 목표로 하고 있다. 우선 서로 다른 통신 모듈을 이용한 디바이스 간에 명령을 주고 받기 위해 물리적 계층 상위 단에 특정 디바이스와 독립적으로 가장 간단한 형태의 의미 부여를 통한 연동이 이루어질 수 있도록 하는 추상화 기능과 이를 바탕으로 좀 더 의미를 부여하여 각종 디바이스가 데이터로부터 이해를 도울 수 있도록 하여 지능을 부여할 수 있는 시맨틱 기술을 핵심으로 정의하고 있다. 이를 바탕으로 궁극적으로 사람을 대체할 수 있는 수준의 지능화된 서비스를 응용에서 지원 가능하게 된다.

기본적으로 추상화 기술은 하부 특정 기술의 복잡성을 숨기고자 하는 것이 초점이 맞추어져 있다. M2M 시스템은 특정 종단 디바이스 구현으로부터 M2M 응용과는 분리된다. 시맨틱 기술은 M2M 엔티티와 데이터에 의미를 부여하고자 하는 것으로 엔티티 유형의 동작(behavior)까지 시스템이 이해할 수 있도록 설계하고자 함이다.

## 7.2. 추상화 (Abstraction) 기술

### 7.2.1. 추상화 요구사항

M2M 서비스는 M2M 응용이 다바이스가 지원하는 특정 기술과 무관하여 디바이스의 기능을 액세스 가능하도록 일반적인 "추상화" 인터페이스를 이용할 수 있도록 해 준다. 표 〈7-1〉은 oneM2M 요구사항 기술 스펙에 정의된 추상화 상위 레벨 요구사항을 보여준다.

✓ 인터워킹 및 추상화에 대한 기본 개념: 인터워킹은 M2M 응용이 M2M 시스템에 연결되어 있는 서로 다른 기술(예, ZigBee, BACnet)로 된 디바이스를 사용가능하도록 하는 기술이다. 반면에 추상화의 목표는 M2M 응용이 외부(non oneM2M) 디바이스를 본래의 인터페이스 (native interface)에 대한 정보나 시맨틱을 이해해야하는 필요성 없이 액세스 가능하도록 하는 것이다.

추상화는 특정 규칙(rule)에 기반한 디바이스 응용 정보 모델과 추상화 응용 정보 모델간의 매핑 절차로서 추상화 리소스와 본래의 리소스간에 매핑이 가능하도록 한다.

표 〈7-1〉 추상화 상위 레벨 요구사항

| 요구사항 ID | 설명 |
|---|---|
| ABR-001 | M2M 시스템은 데이터 표현을 위한 일반적인 구조를 제공해야 한다. |
| ABR-002 | M2M 시스템은 M2M 응용, M2M 디바이스/게이트웨이 및 다른 디바이스에서 사용되는 메타데이터를 포함한 정보 모델간의 해석 메커니즘을 제공할 수 있어야 한다. |
| ABR-003 | M2M 시스템은 가상 디바이스 및 Things를 표현할 수 있는 능력을 제공해야 한다. |

✓ 인터워킹을 위한 정보 모델: 정보 모델(Information model)은 엔티티 상에서 수행될 수 있는 속성(properties), 상관관계 (relationships) 및 오퍼레이션을 포함할 수 있는 엔티티의 추상화되고 정형화된 표현이다. 엔티티가 다른 엔티티와 통신하도록 하는 인터페이스(데이터 유형 및 구조)를 기술한다. 파라미터의 네임, 값의 범위 및 서브구조를 기술한다. 만약 인터페이스가 절차 콜(procedure calls)을 이용한다면, 파라미터가 입력 혹은 출력 파리미터인지에 관한 정보를 반드시 포함해야 한다. 정보 모델의 표현은 특정 시스템에서 정보 모델의 시스템에 구체화된 구현(system specific implementation)으로 이루어진다.

지금까지 설명한 개념을 바탕으로 그림 〈7-2〉는 oneM2M과 기존 시스템에 연결된 디바이스들간의 인터워킹과 추상화에 대한 예를 보여준다.

### 7.2.2. 주요 표준화 기구 동향

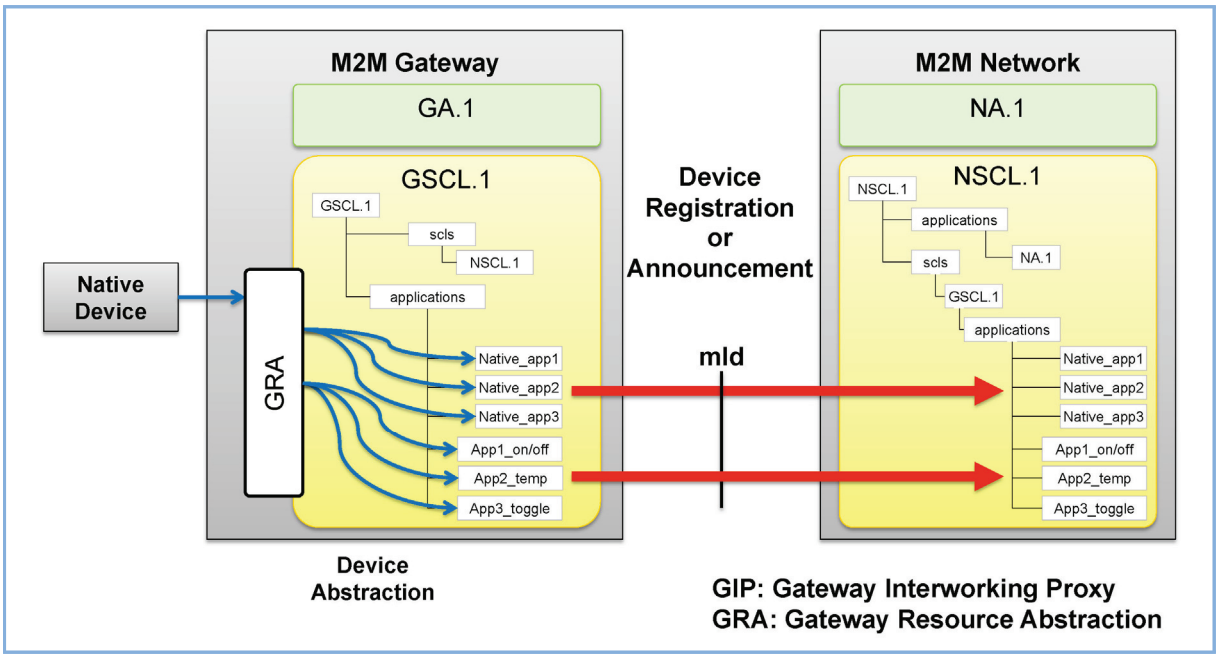



그림 〈7-3〉 디바이스 추상화를 지원하기 위한 상위 레벨 구조

ETSI : ETSI는 M2M 디바이스 아래 추상화와 관련된 핵심 개념을 정의하고 이를 ETSI M2M 구조에 적용하기 위한 방안을 개발하였다.

✓ 추상화: 특정 규칙에 따라 디바이스 정보 모델 및 추상화 정보 모델간의 매핑 프로세서

✓ 추상화 정보 모델: 디바이스 정보 모델로부터 추상화된 공통 기능의 정보 모델

✓ 디바이스 정보 모델: 물리 계층의 특정 기술(예, ZigBee)에 대한 정보 모델

그림 〈7-3〉에 제시한 디바이스 추상화 지원을 위한 ETSI M2M 구조에서 게이트웨이 리소스 추상화 (Gateway Resource Abstraction (GRA)) 능력은 다음과 같은 기능을 포함한다.

- ✓ GSCL(Gateway Service Capabilities Layer)에 있는 새로운 본래 리소스 추가를 감지
- ✓ Non ETSI 호환 리소스로서 본래 리소스로부터 추상화된 리소스 생성
- ✓ 본래 리소스를 해당되는 추상화된 리소스와 링크
- ✓ 추상화 리소스를 NSCL(Network Service Capabilities Layer)에 등록
- ✓ 갱신된 정보를 받을 수 있도록 본래 리소스에 등록
- ✓ 추상화 리소스를 해당 본래 리소와 동기화
- ✓ 추상화된 정보(예, 일반적인 속성 및 커맨드)와 하부 특정 네트워크 정보간의 기능 매핑 제공
- ✓ GRA는 GSCL 혹은 GSCL과 참조점 dIa와 통신하는 응용의 내부 능력이 될수도 있다. GRA는 리소스 추상화 및 연동 능력을 함께 제공하기 위하여 xIP (즉, GIP, NIP 및 DIP)와 통합될 수도 있다.

다음은 일반적인 M2M 추상화 리소스를 본래의 M2M 리소스에 매핑하기 위행 사용되는 매핑 원리를 보여준다. 여기에는 추상화 디바이스를 기술하는 두 가지 방식이 있다. 첫 번째 매핑 방식 그림 〈7-4〉 왼쪽은 각각의 추상화 디바이스를 응용으로 고려하는 것이다. 두 번째 매핑 방식 그림 〈7-4〉 오른쪽은 각 추상화 디바이스가 콘테이너 리소스로 간주되어 이 들이 소속되어 있는 네트워크 응용에 등록될 수 있도록 서브컨테이너를 이용하는 것이다.

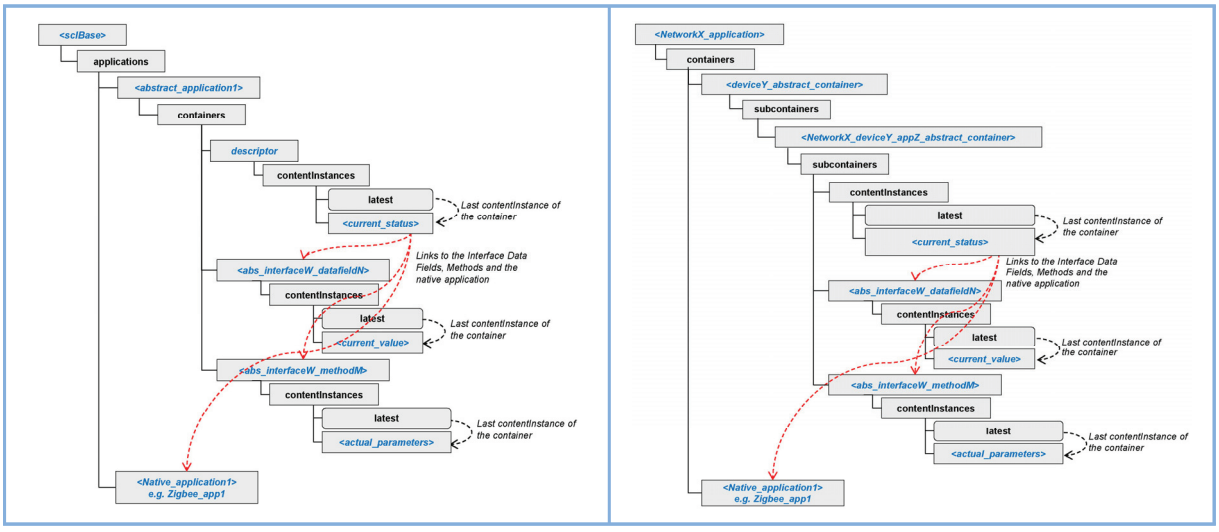



그림 〈7-4〉 ETSI M2M 리소스 및 추상화 리소스 매핑

### HGI (Home Gateway Initiative)

HGI에서는 디바이스 추상화 개념을 바탕으로 스마트 홈 추상화 계층인 SHAL(Smart Home Abstraction Layer)을 정의하였으며 SHAL은 다음과 같은 특징을 가진다.

- ✓ 가전 기기를 홈 자동화 기술과 독립적으로 공통 표현으로 매핑한다.
- ✓ 특정 프로토콜 응용으로부터 프로토콜에 독립적인 요구로 변환하고 이를 적절한 드라이버에 전달한다.
- ✓ 특정 기술에 종속되지 않도록 가전기기를 추상화 응용 인터페이스로 표현한다.

그림 〈7-5〉에 제시한 SHAL의 주요 목표는 다음과 같다.

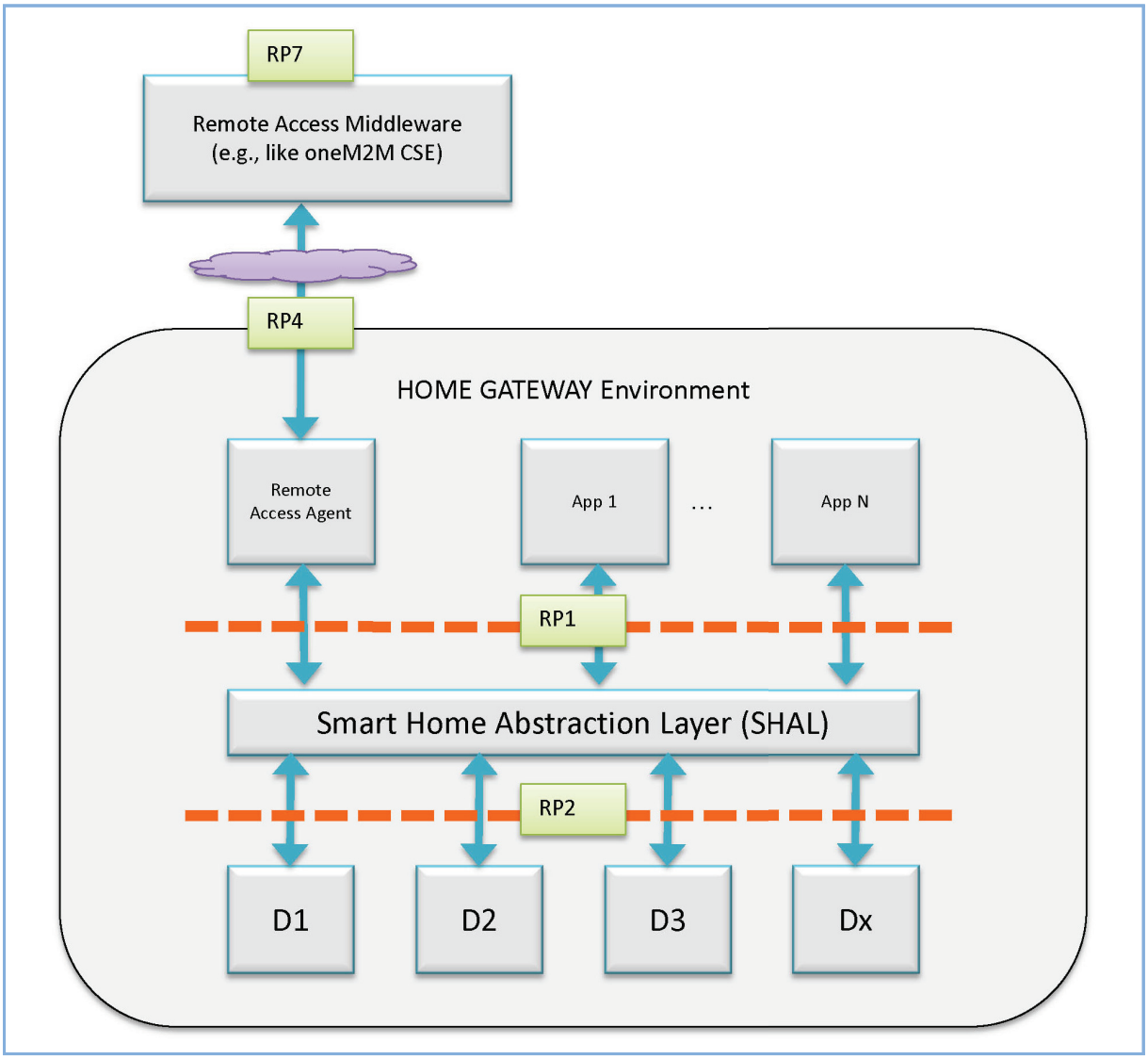



그림 〈7-5〉 HGI 홈 추상화 계층

- ✓ 응용 개발자가 홈 가전 기기에 커맨드를 보내고, 제어 및 쿼리를 하기 위한 단일화된 APIs를 제공
- ✓ 응용 개발자가 ZigBee, Z-Wave, 무선 m-bus 등에 대한 어떤 것도 알 필요가 없도록 하기 위하여 하부 HAN(Home Area Network) 기술의 독립성 제공
- ✓ 응용 프로그램이 서로 다른 HGI 호환 디바이스간 쉽게 이식 가능하도록 함
- ✓ 서비스 불통 없이 추가적인 HAN 기술 지원이 가능하도록 시스템 확장이 용이하도록 함
- ✓ 응용은 특정 기술 기능을 이용한 패스스루(pass-through) 메커니즘을 이용할 수 있도록 해야 함

## 7.3. 시멘틱 M2M 시스템 기술

### 7.3.1. 시맨틱 요구사항

시맨틱은 메터 데이터나 에노테이션을 이용한 기계가 해석 가능한 형태로 표현이 될 수 있도록 해 준다. 이러한 표현에는 데이터, 사용자, 디바이스, 응용, 환경 등에 대한 다양한 종류의 정보가 포함된다. 시맨틱은 또한 M2M 디바이스 및 데이터의 서로 다른 속성(attribute)을 기술 할 수 있도록 도와준다. 표 〈7-2〉는 oneM2M 요구사항 기술 스펙에 정의된 시맨틱 상위 레벨 요구사항을 보여준다.

표 〈7-2〉 시맨틱 상위 레벨 요구사항

| 요구사항 ID | 설명 |
|---|---|
| SMR-001 | M2M 시스템은 리소스 및 M2M 응용의 시맨틱 표현을 관리할 수 있는 능력 (생성, 검색, 갱신, 삭제, 연계/링크, 등)을 제공해야 한다. |
| SMR-002 | M2M 시스템은 M2M 응용이 이를 활용 가능하도록 (Things 간의 관계를 포함한) 시맨틱 표현을 위한 공통 모델링 언어를 지원해야 한다. |
| SMR-003 | M2M 시스템은 시맨틱 표현을 위한 서로 다른 모델링 언어간에 연동 능력을 제공 가능해야 한다. |
| SMR-004 | M2M 시스템은 시맨틱 표현에 기반하여 M2M 리소스를 발견할 수 있는 능력을 제공해야 한다. |
| SMR-005 | M2M 시스템은 M2M 시스템 외부에 있는 시매틱 표현을 접근할 수 있는 능력을 제공해야 한다. |
| SMR-006 | M2M 시스템은 M2M 응용, M2M 시스템으로부터 시맨틱 표현에 기반한 M2M 데이터 분석을 수행할 수 있는 능력을 제공 가능해야 한다. |
| SMR-007 | M2M 시스템은 M2M 응용, M2M 시스템으로부터 M2M 데이터를 이용한 시맨틱 메쉬업을 수행할 수 있는 능력을 제공 가능해야 한다. (예, 가상 디바이스 생성, 새로운 M2M 서비스 제공, 등) |

### 7.3.2. 시맨틱 기술

시맨틱을 위한 핵심 기술을 크게 다음 세가지로 나눌 수 있다.

- ✓ **시멘틱 에노테이션:** M2M 서비슬 제공하기 위한 다양한 엔티티(예, 데이터, 사용자, 응용 등)의 시맨틱 정보를 제공
- ✓ **온톨로지**[11]**:** 물리(physical), 가상(virtual) 및 추상(abstraction) 엔티티의 시맨틱을 모델링하기 위하여 온톨로지를 사용, 온톨로지는 표준 언어를 사용하여 기계가 읽을 수 있는 형태로 표현이 가능함
- ✓ **시맨틱 처리**
  - **시맨틱 발견:** 시맨틱 정보에 기반하여 리소스나 서비스를 발견하고 연결시키기 위하여 M2M 발견 메커니즘을 확장
  - **시맨틱 추론:** 시맨틱 에노테이션된 데이터의 분류 및 새로운 연계관계를 도출
  - **시맨틱 메쉬업:** 새로운 가상 디바이스를 생성하고 이를 바탕으로 새로운 M2M 서비스를 제공

### 7.3.3. 시맨틱을 위한 핵심 기능

그림 〈7-6〉은 다양한 형태의 M2M응용을 위한 시맨틱을 지원하기 위한 일반적인 기능 모델을 보여준다. 그림 〈7-6〉에 있는 기능은 다음 3가지 부분으로 논리적으로 구성된다.

- ✓ 서비스 액세스: 다양한 M2M 응용과의 인터페이스를 제공
- ✓ 추상화 및 시맨틱: M2M 데이터와 리소스에 시맨틱을 지원하기 위한 핵심 기능을 수행
- ✓ 데이터 액세스: M2M 데이터를 액세스 하기 위한 디바이스 혹은 게이트웨이와 연결성을 제공

11) 온톨로지는 특정한 영역을 표현하는 데이터 모델로서 특정한 영역(Domain)에 속하는 개념과, 개념 사이의 관계를 기술하는 정형(Formal) 어휘의 집합으로 정의된다.

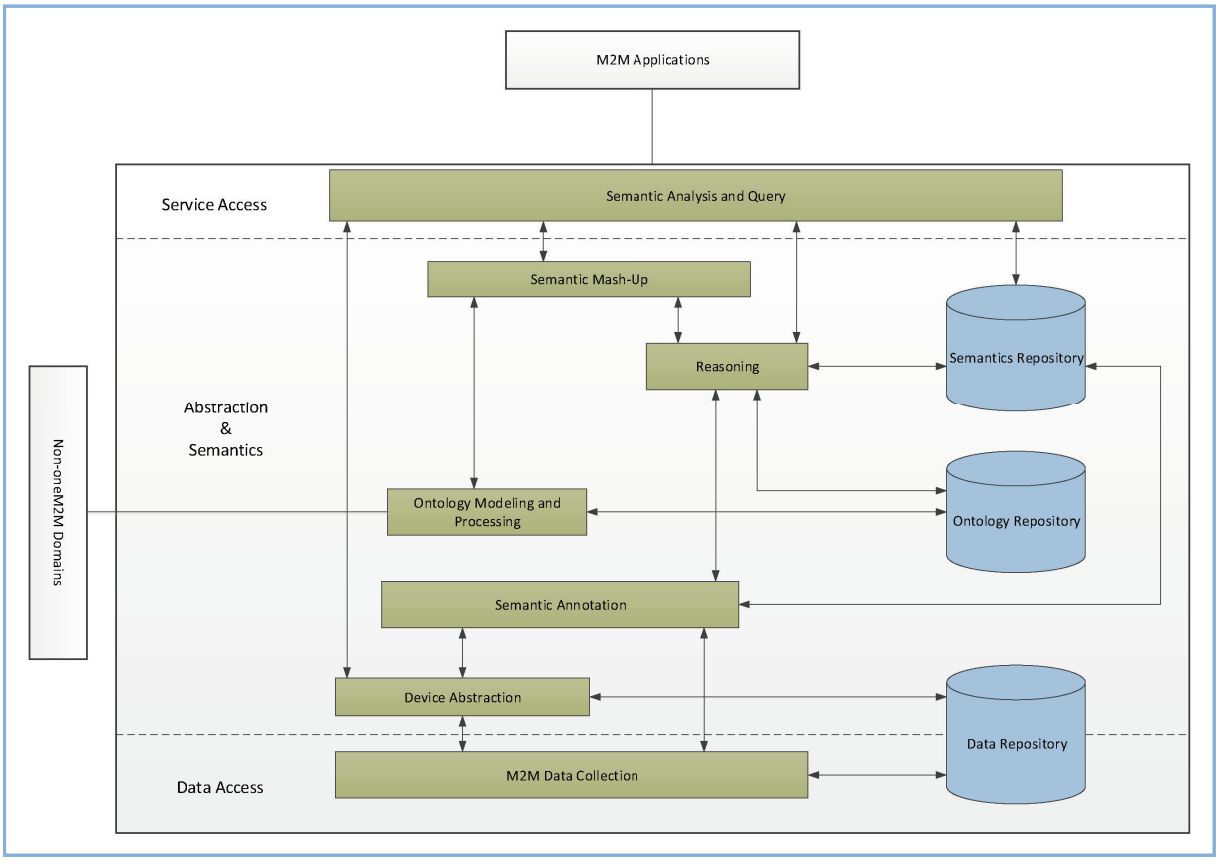



그림 〈7-6〉 시맨틱 지원을 위한 일반 기능 모델

**시맨틱 분석 및 쿼리 (Semantic Analysis & Query):** 시맨틱 쿼리 분석에서는 M2M 응용으로부터 받은 요청(REQUEST)을 시맨틱하게 분석을 수행하는 곳이다. 분석 결과를 바탕으로 시맨틱 정보를 요청하기 위하여 시맨틱 쿼리 메시지를 만들어 이를 추상화 및 시맨틱 계층에 있는 기능 컴포넌트(예, 온톨로지 리포지토리, 추론, 시맨틱 메쉬업, 등)에 전달하는 역할을 수행한다. 요구한 정보를 획득한 후에 M2M 응용에 응답을 보낸다.

**추론 (Reasoning):** 추론은 시맨틱 에노테이션된 데이터로부터 새로운 묵시적 지식을 만들어 내고 복잡한 사용자 쿼리에 응답하기 위한 메커니즘이다. 본 기능은 확정사실(asserted facts) 혹은 공리(axioms)로부터 논리적 결과를 유추할 수 있도록 소프트웨어 형태로 구현될 수 있다.

**온톨로지 리포지토리 (Ontology Repository):** 온톨로지 리포지토리는 온톨로지의 저장소이다. 온톨로지는 현실을 표현가기 위한 기본 개념의 공유된 어휘(vocabulary)를 정의하고 오퍼레이션을 포함한 개념을 명세하는 인공 언어(linguistic artifact)로서 정의할 수 있다.

온톨로지 리포지토리는 OWL[12]혹은 RDF[13]로 기술된 온톨로지를 저장, 검색 및 관리할 수 있는 방법을 제공한다. 다양한 목적(예, 온톨로지의 출판, 공유, 인덱싱, 검색, 재사용, 등)을 위하여 상당히 많은 개념을 가진 대규모 데이터 집합을 처리할 수 있어야 한다. 이를 위하여 쿼리 언어(예, RDF Data Query Language (RDQL), QWL Query Language (QWL-QL), SPARQL Protocol And RDF Query Language (SPARQL), 등)를 지원한다.

**온톨로지 모델링 및 처리 (Ontology Modelling and Processing):** 온톨로지 모델링은 도메인을 설계하고 개념에 따라 추론을 지원할 수 있는 온톨로지를 만들기 위한 과정이다. 온톨로지 모델링을 위한 언어로는XML-based RDF, RDF Schema(RDFS), OWL 등이 있다.

온톨로지 처리는 내, 외부의 M2M 도메인으로부터 출판되고 설계된 오톨로지를 분류, 저장 및 발견 기능 등을 수행하는 절차이다. 온톨로지는 리소스에 시맨틱이 가능하도록 공유되거나 사용될수 있도는 단일화된 언어(예, RDFS/OWL)로 온톨로지 레포지토리에 변환되거나 저장된다.

**시맨틱 메쉬업(Semantic Mash-up):** 시맨틱 메쉬업은 M2M 시스템에 있는 기존 M2M 리소스로부터 시멘틱 표현으로부터 관련 정보를 획득하여 실세계에 존재하지 않는 새로운 가상 디바이스 생성을 통해 새로운 서비스를 지원할 수 있는 기능을 제공한다.

**시맨틱 에노테이션(Semantic Annotation):** M2M 리소스의 시맨틱 에노테이션은 서로 다른 M2M 응용에 일관성있게 데이터 해석 및 데이터 상호호환성을 제공하기 위하여 M2M 리소스에 시맨틱 정보를 추가하는 방법이다. 시맨틱 에노테이션된 M2M 리소스는 리소가 어떤 데이터를 제공할 수 있고, 이런 데이터가 무엇인지를 이해하고 있는 M2M 응용에 의해 연결된다. 이런 에노테이션은 좀더 의미있는 표현을 제공하고 기존 M2M 시스템에 M2M 데이터까지도 표현 가능하게 해 준다. 여기서 시맨틱 정보는 RDF와 OWL에 의해 에노테이션된다.

**시맨틱 레포지토리 (Semantics Repository):** 시맨틱 리포지토리는 리소스의 시맨틱 에노테이션 정보와 추론 결과로부터 새로운 묵시적인 시맨틱 정보를 저장한다. 또한 쿼리를 위한 언어(예, RDF Data Query Language (RDQL), QWL Query Language (QWL-QL), SPARQL Protocol And RDF Query Language (SPARQL) 등)를 지원한다.

**디바이스 추상화 (Device Abstraction):** 디바이스 추상화는 특정 규칙을 바탕으로 디바이스 응용 정보 모델과 추상화 응용 정보 모델간의 매핑 프로세스이다. 이는 다중,

12) OWL(Web Ontology Language)은 관계들 간의 hierarchy, 관계 인스턴스 내에서의 논리적 제약조건 등을 포함한 언어로. 정밀하고 논리적인 추론을 필요로 하는 경우에 사용한다

13) RDF(Resource Description Framework)는 XML에서 발전한 형태이며, subject, object, predicate으로 이루어지며, 단순하게 개념 혹은 인스턴스 사이의 관계를 나타낸다.

서로 다른 그렇지만 시맨틱하게 유사한 디바이스가 추상화된 응용 정보 모델의 기능을 제공하는 가상 디바이스를 통해 통신이 가능하도록 해 준다.

**데이터 레포지토리 (Data Repository):** 데이터 레포지토리는 기본적으로 새로운 데이터를 저장한다. 또한 저장된 데이터를 검색, 수정 및 삭제하는 기능을 제공한다.

**M2M 데이터 수집 (M2M Data Collection):** 센서 혹은 게이트웨이와 연결되어 있는 디바이스로부터 데이터(raw data)가 수집되어 데이터 레포지토리에 저장된다.

### 7.3.4. 주요 표준화 기구 동향

ETSI 시맨틱 기술

ETSI는 oneM2M 표준화 이전에 그림 〈7-7〉과 같이 M2M에 대한 표준 개발을 진행하면 이미 시맨틱 연동, 커맨드 매핑 및 발견을 위한 시맨틱 엔진과 M2M 온톨로지를 참조하고 상위 시맨틱 응용을 지원할 수 있는 인터페이스(mla (REST))를 정의해 두었다.

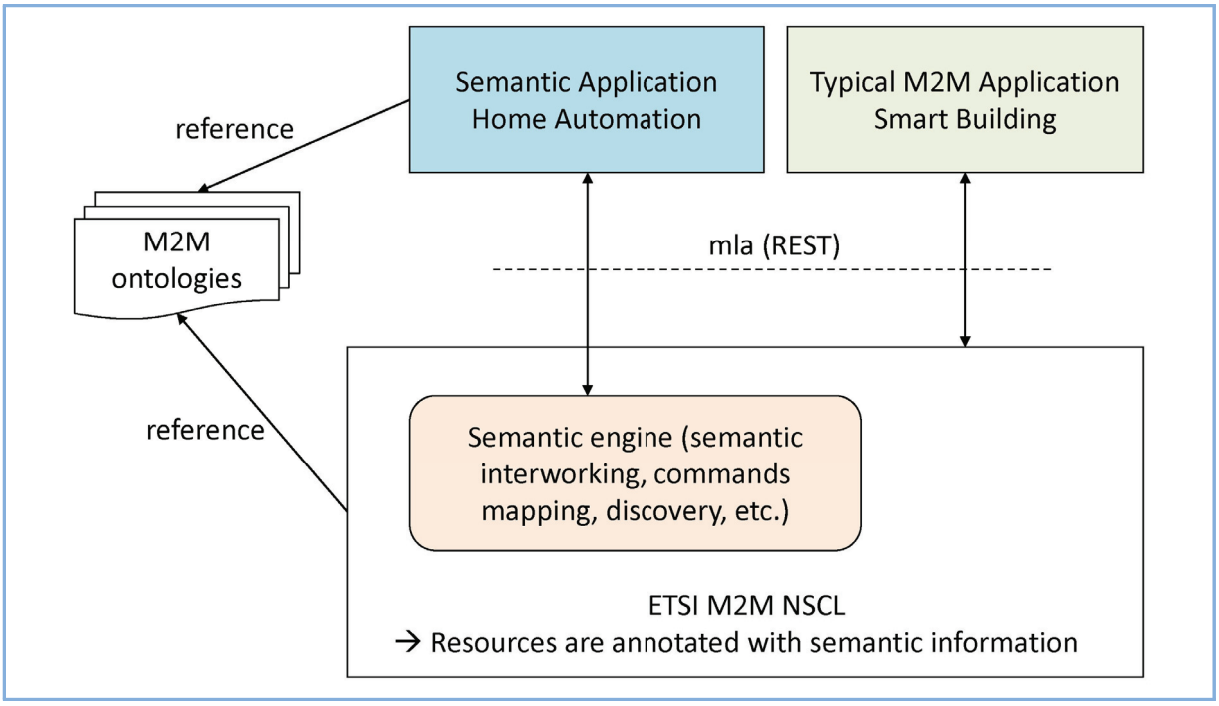



그림 〈7-7〉 ETSI 시맨틱 M2M 시스템

특히, 여기에서 M2M 리소스에 시맨틱 정보를 담기 위한 시맨틱 에노테이션 방법을 명세하였으며, 이를 위한 상세 속성(attibutes) 등도 표 〈7-3〉과 같이 정의해 두었다. 여기에서 응용 리소스의 서브 리소스의 부분으로서 Namespace URI의 예를 보여준다.

ETSI에서 정의한 시맨틱 엔진에서 제공할 수 있는 기능의 전형적인 예로서 시맨틱 가상 메쉬업 절차를 만들었다.

- ✓ 웹 서비스 도메인에서, 메쉬업은 새로운 서비스를 생성하기 위하여 하나 이상의 웹 리소스로부터 웹 데이터를 구성하기 위한 방법이다. 예로서 단일 위치에 모아진 웹 콘텐츠를 결합하는 다중 검색 엔진과 새로운 집합체(aggregator)로부터 웹 검색 결과를 혼합하는 메타크라울러(metacrawlers)가 있다. 유사하게, 메쉬업 기술은 M2M 시스템에 새로운 M2M 리소스를 생성하기 위해 사용될 수 있다.
- ✓ M2M 시스템에서 M2M 응용은 물리적 리소스와 유사하게 행동하고 새로운 정보(예, 차량의 평균 속도 등)를 제공하는 "가상(virtual) things"를 만들어 낼 수 있다. 이런 "가상 things"는 다른 M2M 리소스와 동일하게 M2M 시스템 내에서 검색되고 발견될 수 있다. 그렇지만 물리적 things과는 반대로 가상 things은 단지 소프트웨어만으로만 구현되고 네트워크 연결성을 요구하지 않는다.
- ✓ 새로운 가상 thing이 M2M 시스템에 등록되거나 만들어질 때, 멤버 M2M 리소스의 리스트가 thing의 속성으로서 함께 저장된다. 만약 가상 thing이 쿼리를 수신하는 시점에 동적으로 정보를 수집한다면, 멤버 리소를 수집하는 미리 프로그래밍 된 쿼리가 다른 정보와 함께 저장되게 된다.
- ✓ 일단 가상 thing이 NSCL에 추가되게 되면, 다른 M2M 리소스와 동일하게 취급되어 처리되게 된다. 이것은 가상 things이 발견될 수 있도록 M2M 응용에 노출(expose)되어졌음을 의미한다.

그림 〈7-8〉은 시맨틱 가상 메쉬업 처리의 예를 보여준다.

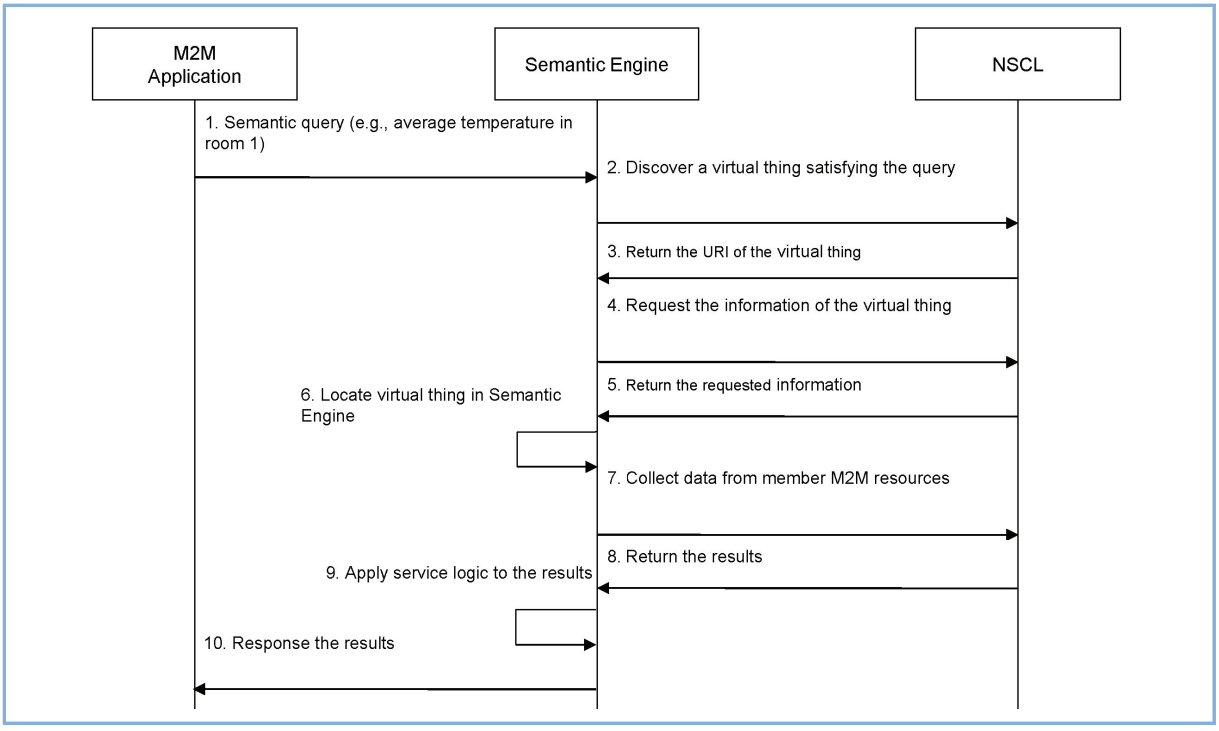



그림 〈7-8〉 시맨틱 가상 메쉬업 절차

표 〈7-3〉 시맨틱 정보를 에노테이션하기 위한 속성 (Attributes)

| 이름 | 설명 |
|---|---|
| semanticInfo | 이 속성은 Thing의 시맨틱 표현을 포함한다. 이 온톨로지는 Thing의 의미가 무엇인지를 보여준다. |
| Relations | 이 속성은 다른 Things와의 상관관계를 기술하기 위하여 사용된다. 이 속성은 한쌍으로 된 포맷(예, 〈relation : link to other thing〉)을 가질 수 있다. 예를 들면, 만약 Zigbee 센서가 Zigbee 컨트롤러 #1에 의해 제어된다면, 〈m2m:controlledBy – URI to Zigbee controller #1〉이 Zigbee 컨트롤러에게 관계를 표현해 주는 하나의 방식이 될 수 있다. |
| namespaceURIs | 이 속성은 namespace URIs를 표현하기 위하여 사용된다. 이 속성은 namespace 프리픽스(prefix)와 namespace URI에 대한 정보를 포함한다. 예를 들면, URI http://www.etsi-m2m.org을 가지는 m2m namespace는 다음과 같이 표현될 수 있다. m2m=http://www.etsi-m2m.org/sensor#. |

### OGC Sensor Web Enablement

- ✓ OGC(Open Geospatial Consortium)에서는 지리공간정보를 통합하여 처리하기 위한 플랫폼 기술로서 센서웹(sensor web)을 개발하였다. 센서웹은 인공물과 자연물에 컴퓨터 기능을 갖는 다양한 센서를 설치하고 이를 웹으로 연결시켜 시설물 및 환경, 교통 상태, 재난재해 등을 모니터링 하는 개념이다.
- ✓ Sensor Web Enablement(SWE)는 OGC에서 모든 종류의 센서 시스템과 웹에 연결된 센서들을 이용하기 위하여 개발된 혁신적인 개방형 표준 프레임 워크이다. 이러한 SWE의 목표는 웹이나 인터넷에 접속 가능한 센서들이나 기구들이 웹을 통하여 어디서든지 조작가능하고 응용 가능하도록 하는 것이다.

OGC는 웹에서 센서들의 상호 운용성을 확보하기 위하여 설계된 여러 개의 표준들과 Best Practices 문서를 승인하였다. SWE의 기본적인 요소들이 다음과 같이 센서정보에 대한 표준들과 센서데이터를 위한 웹 서비스 표준들로 구분되어 정의되었고, 프로토타이핑을 통하여 테스트되었다.

센서 정보의 모델과 스키마에 대한 표준으로 다음 표준이 있다.

- ✓ SWE Common – SWE 사양들에 공동으로 사용되는 데이터 모델
- ✓ Sensor Model Language (SensorML) – 다양한 센서들을 추상화한 표준 모델로, 센서에 대한 메타데이터와 수행가능한 함수와 태스크, 인자를 제공
- ✓ Observations and Measurements (O&M) – 센서가 관측 또는 측정한 센싱 정보를 인코딩하는 표준 모델과 스키마

- ✓ Transducer Markup Language (TML) – 센서와 구동장치를 합한 트랜스듀서에 관한 정보를 모델링하는 함수와 데이터를 획득하고 저장 및 전달하는 메시지 포맷
- ✓ Sensor Observation Service (SOS) – 센서 또는 센서 시스템으로부터 관측된 데이터에 대한 접근을 제공하는 표준 인터페이스
- ✓ Sensor Alert Service (SAS) – 센서 관측에 기반하여 센서 값으로부터 경보(alert)를 사용자에게 전달하는 표준 인터페이스
- ✓ Sensor Planning Service (SPS) – 사용자가 센서에 임무를 부여하고 이를 수행하는 것을 지원하는 표준 인터페이스
- ✓ Web Notification Service (WNS) – SAS 등의 웹 서비스와 사용자간의 비동기적인 메시지 전달을 위한 인터페이스

### W3C 시맨틱 센서 네트워크 온톨로지

W3C(World Wide Web Consortium)의 시맨틱 센서 네트워크 온톨로지(Semantic Sensor Network Ontology, SSNO) 표준은 센서 네트워크에 대한 의미적 연동에 필요한 핵심 온톨로지의 구조, 어휘, 표현방법, 사용 시나리오 등에 대해 정의하며, 핵심 온톨로지의 주요 모듈별 클래스와 속성들을 정의하였다. 이로서 시맨틱 지원을 위하여 센서 네트워크에서 수집되는 다양한 종류의 센서들의 센싱 데이터들에 대해 의미적 연관성을 표현할 수 있도록 하기 위한 핵심 온톨로지이다.

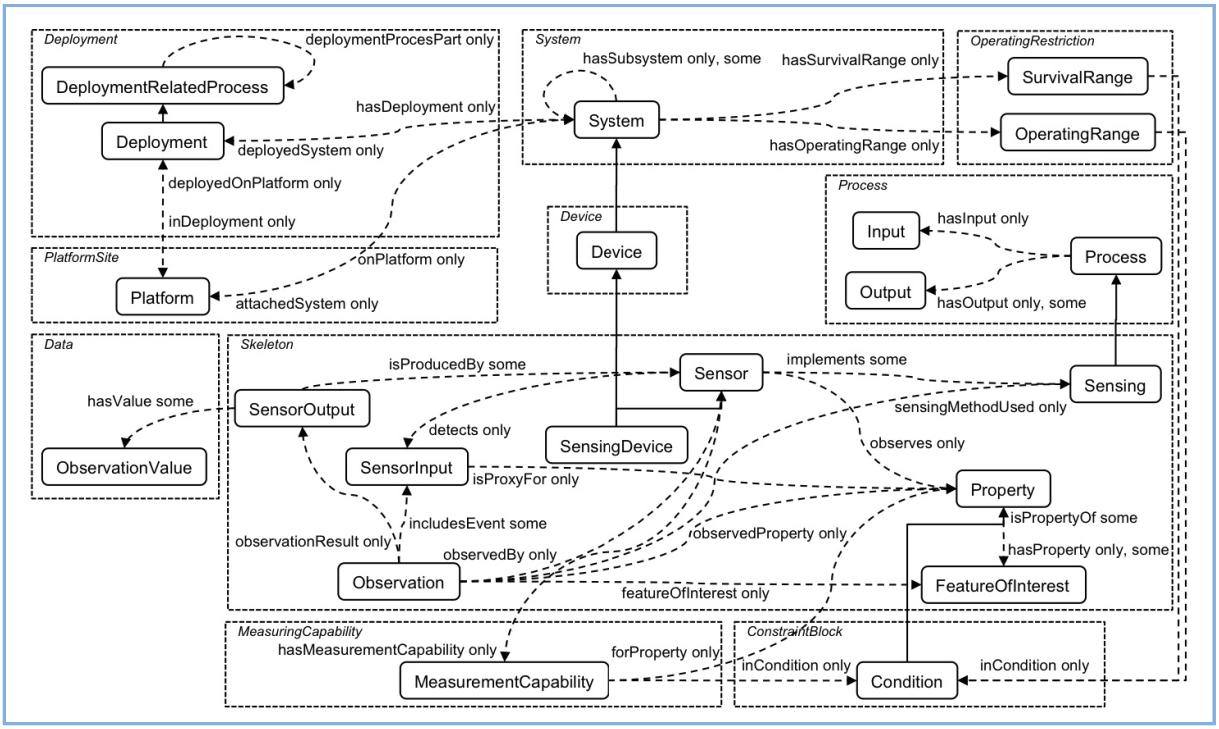



그림 〈7-9〉 SSNO의 클래스 및 속성 개요

자극(Stimuli)-센서(Sensors)-관찰(Observations) 온톨로지 설계 패턴은 시맨틱 센서 웹과 특히 링크드 데이터에 대한 언어 및 온톨로지에 기반한 모든 유형의 센서 또는 관찰을 위한 것이다. 이 패턴은 여러 가지 응용 영역에 대한 재사용이 가능하도록 최소 온톨로지 노력의 원칙에 따라 개발되었다. 이는 다른 상위 온톨로지에 맞게 조정되지 않았으며, 자극, 센서 및 관찰의개념을 중심으로 한 최소한의 클래스 및 연관성이 도입되었다. 그림 〈7-9〉의 SSNO의 구조에서 모듈에는 센서 또는 그 관찰에 대한 특정 측면을 표현하는데 사용되는 속성 및 클래스가 있다. 예를 들면, 센서, 관찰, 특정 기능, 센싱 프로세스(센서의 작동 및 관찰 방법 등), 센서의 배치 또는 플랫폼 부착 방법, 센서의 측정 기능은 물론 특수 환경 하에서 센서의 생존 속성 등이 있다.

### 7.3.5. 시맨틱 지원을 위한 상세 요구사항

oneM2M에서 시맨틱 표준화를 진행하면서 여러 가지 기술적 이슈에 대한 논의를 바탕으로 앞서 제시한 상위 레벨 요구사항 외에 다음과 같은 상세 요구사항을 발굴하였다.

**시멘틱 에노테이션**

- ✓ M2M 시스템은 관련 온톨로지에 기반해 M2M 시스템에 의해 처리(전달)될 응용 관련 데이터(예, 컨테이너)의 시맨틱 에노테이션을 지원해야 한다.
- ✓ M2M 시스템은 M2M 시스템의 엔티티, 이들의 정보 모델 및 관계를 포함하는 온톨로지에 의해 기술되어야 한다.
- ✓ M2M 시스템은 동일 M2M 리소스에 대하여 다중 시맨틱 온톨로지의 동시 사용을 지원할 수 있어야 한다.
- ✓ M2M 시스템은 시맨틱 표현을 발표할 수 있는 능력을 제공해야 한다.
- ✓ M2M 시스템은 시매틱 표현을 분석하고 해석할 수 있어야 한다.
- ✓ M2M 시스템은 서비스 발견, 메쉬업 및 데이터 분석을 위하여 온톨로지를 이용한 디바이스 및 서비스 정보를 기술할 수 있는 능력을 제공해야 한다.
- ✓ M2M 시스템은 M2M 시스템 엔티티(예, AEs (Application Entities) 혹은 CSEs (Common Services Entities))가 M2M 시스템 내에 시맨틱 표현을 발표할 수 있는 능력을 제공해야 한다.

**온톨로지**

- ✓ M2M 시스템은 다른 정보 모델을 기반으로 표현된 온톨로지와 oneM2M 시스템을 표현하는 온톨로지 간의 링크를 제공해야 한다.
- ✓ M2M 시스템은 새롭게 생성되는 oneM2M 응용을 지원하기 위하여 확장된 온톨로지를 지원 가능해야 한다.

- ✓ M2M 시스템의 리소스에 의해 표현되지 않은 엔티티를 포함하는 온톨로지를 지원할 수 있어야 한다.
- ✓ M2M 시스템은 M2M 응용에서 일반적으로 사용되는 공통 온톨로지(예, 위치, 시간 온톨로지 등)를 지원할 수 있어야 한다.
- ✓ M2M 시스템은 시맨틱 추론에 기반한 새로운 온톨로지의 생성, 사용 및 발표를 지원해야 한다.
- ✓ M2M 시스템은 oneM2M의 외부에서 정의된 온톨로지를 가져오거나 조회할 수 있어야 한다.
- ✓ M2M 시스템은 되도록이면 사람의 도움 없이 M2M 시스템 내부 혹은 외부에서 정의된 온톨로지의 변경에 적응될 수 있도록 해야 한다.
- ✓ M2M 시스템은 M2M 응용, M2M 디바이스/게이트웨이 및 다른 디바이스에 의해 사용된 정보 모델간에 해석 메커니즘을 제공 가능해야 한다.
- ✓ M2M 시스템은 데이터 표현을 위한 일반적인 구조를 제공 가능해야 한다.
- ✓ M2M 시스템은 서비스 발견, 메쉬업 및 데이터 분석을 위하여 온톨로지를 이용한 디바이스 및 서비스 정보를 표현할 수 있는 능력을 제공해야 한다.
- ✓ M2M 시스템은 다양한 M2M 디바이스(예, 가전기기)를 위하여 시맨틱 디바이스 템플릿을 지원할 수 있어야 한다.
- ✓ M2M 시스템은 M2M 디바이스 및 이와 연계된 실제 세계 엔티티(예, 룸)의 시맨틱 정보를 설계하기 위한 공통 온톨로지를 지원할 수 있어야 한다.
- ✓ M2M 시스템은 M2M 응용 및 3rd party 시스템에 의해 발견되고 검색될 수 있는 분류된 온톨로지 정보를 지원해야 한다.

### 시맨틱 쿼리

- ✓ M2M 시스템은 시맨틱 정보에 기반한 대상 M2M 디바이스의 발견이 가능하도록 시맨틱 쿼리를 지원해야 한다.

### 시맨틱 메쉬업

- ✓ M2M 시스템은 시맨틱 표현 및 추론을 이용 기존 디바이스 리소스에 기반한 가상 디바이스 리소스를 메쉬업하는 능력을 제공해야 한다.
- ✓ M2M 시스템은 데이터 분석(예, 계산)이 가능하도록 M2M 시스템 내에 임베디드 시맨틱 메쉬업 처리 능력을 제공해야 한다.
- ✓ M2M 시스템은 새로운 서비스를 생성하기 위하여 하나 이상의 리소스로부터 시맨틱 데이트를 집합(aggreage)시킬 수 있어야 한다.

✓ M2M 시스템은 시맨틱 정보를 획득하여 새로운 서비스를 위한 가상 디바이스의 생성을 지원할 수 있다.

✓ M2M 시스템은 가상 디바이스와 Things을 표현할 수 있는 능력을 제공해야 한다.

#### 시맨틱 추론

✓ M2M 시스템은 시맨틱 에노테이션된 데이터에 기반한 시멘틱 온톨로지 추론을 지원할 수 있어야 한다.

✓ M2M 시스템은 시맨틱 규칙에 기반한 시맨틱 추론을 지원할 수 있어야 한다.

✓ M2M 시스템은 참조된 온톨로지에 따라 시맨틱 에노테이션된 정보로부터 묵시적인 지식을 획득하기 위한 추론 능력을 지원할 수 있어야 한다.

#### 시맨틱 인지에 의해 발생된 동작

✓ M2M 시스템은 시맨틱 에노테이션 및 온톨로지로 기술된 서비스 로직을 해석하고 적용하는 능력을 제공해야 한다.

## 7.4. oneM2M에서 추상화 및 시맨틱 지원 방안

### 7.4.1. 설계 측면 고려사항

릴리즈1에서 oneM2M 시스템은 응용이 리소스 개념에 기반하여 불분명 특정 응용 컨테이너를 이용하여 서로 상호작용이 될 수 있도록 한다. 서로 다른 인코딩이 적용된 유사 데이터(예, 온도)를 사용하는 응용간에 상호작용을 가능하도록 하기 위하여 데이터가 추상화된 공통 포맷으로 변환되어 서비스가 제공될 수 있도록 공통 추상화 계층이 만들어진다.

이런 추상화 계층은 통신과 관련하여 응용 개발자가 하부의 여러 통신 파트너의 다양성을 고려할 필요가 없게 해 준다. 그러나 대부분의 경우에 응용은 릴리즈1 발견 기능이 아주 제약되어 있고 시맨틱에 기반하여 다른 응용을 발견할 수 없기 때문에 여전히 이전 통신 파트너를 알 필요가 있다. 데이터는 이 데이터의 구조나 시맨틱에 대한 정보를 제공하지 않고 블랙박스로 처리된다. 따라서 이런 정보는 해당 응용에 의해 미리 알아야 하고, 발견될 수 없다.

서로 다른 응용에 의해 정보나 기능 재사용이 가능한 좀 더 유연한 시스템은 oneM2M 시스템 내에서 교환되는 응용 데이터의 구조나 시맨틱이 좀 더 명확하게 될 수 있다. 단순히 상호작용의 목적이 아닌 제공된 기능에 대한 정보의 공유 목적으로 공통 추상화를 적용할 수 있다.

사물인터넷이 되면서 응용이 실제 물리 세상까지 범위가 확장되고 있다. 센싱 뿐만 아니라 이 정보를 이용하여 관련 액션을 통해 물리 세상을 변화시킬 수 있는 단계로 진화되고 있다. 광범위한 측면에서 물리 세상은 Things에 의해 구성되는 Things은 빌딩, 룸, 창문, 도로, 병 등이 해당될 수 있다. 궁극적으로 이런 상호작용을 중재하는 디바이스를 이용한 응용이 이런 Things과도 상호작용할 수 있도록 하는 것이다.

### 도메인 모델, 디바이스 및 Things 구조

그림 〈7-10〉 (a)에 제시한 도메인 모델에서 oneM2M 서비스는 직접적으로 응용 및 디바이스와 통신에 연계되어 있다. 궁극적으로는 응용과 Things 사이의 중계 작용을 지원하여 실 세계도 함께 지원될 수 있도록 설계되어야 한다. 따라서 디바이스와 Things이 어떻게 적절하게 모델링되고 어떻게 표현될 수 있는지, 또한 각 모델이 향후 oneM2M 시스템에 어떻게 사용될 수 있는지를 살펴본다.

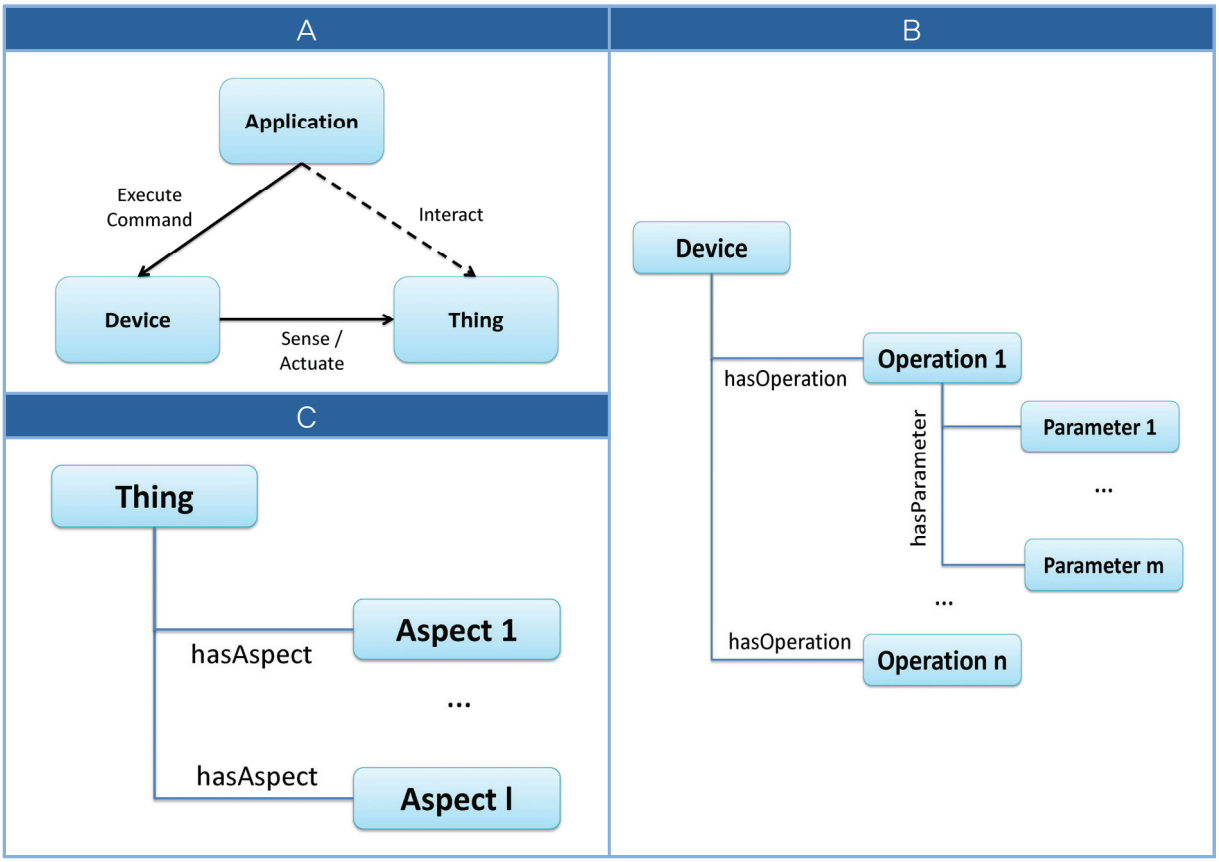



그림 〈7-10〉 도메인 모델, 디바이스 구조, Thing 구조

그림 〈7-10〉 (b)와 (c)에 제시한 디바이스 구조와 Thing 구조는 디바이스와 Tings의 핵심 개념을 바탕으로 oneM2M 시스템이 관련 기능 지원이 가능하도록 하기 위하여 필요한 사항을 보여준다.

기본적으로 Things은 서로 다른 Aspect를 가지고 있다. 예를 들면, 룸은 크기와 높이, 온도, 소음 정도와 같은 정보를 가질 수 있다. 이런 Aspect는 정보를 제공하는 디바이스와도 직접적으로 연계되어 있다. 현재의 소음 정도 혹은 디바이스가 히터나 에어컨일 경우에 현재 온도 같은 것에 영향을 준다. 이런 다양한 Aspect에 다른 Tings과 상관 관계를 가지고 설계되어야 한다.

디바이스의 경우에 이런 Aspect들이 특정 Things 유형과 함께 정의될 수 있다. 디바이스 유형은 계층적으로 정의될 수 있다. 단일 오퍼레이션(MyOperation1)이 정의된 디바이스 슈퍼 유형이 있다. 디바이스 유형 1과 디바이스 유형 2는 디바이스 슈퍼 유형을 상속받는다. 즉, 이런 유형의 인스턴스는 또한 MyOperation 1을 지원한다. 추가로 디바이스 유형 1은 디바이스 유형 1의 모든 디바이스와 이로부터 상속받은 모든 유형의 디바이스를 위해 MyOperation 2를 정의할 수 있다. 여기에서 오퍼레이션은 필수 혹은 선택일 수 있다.

개념, 인스턴스에 의해 제공된 주어진 구조는 유형에 기반하여 모델링 될 수 있다. 지금까지 내용을 종합하면 그림 〈7-11〉과 같이 오퍼레이션과 파라미터에 제한된 접근 방식을 도입하는 것을 목적으로 디바이스의 개념과 구조를 보여 줄 수 있다. 이러한 디바이스의 개념 및 구조는 oneM2M에 정의될 수 있고, 관련 기능 구현을 위한 기본으로 적용할 수 있다.

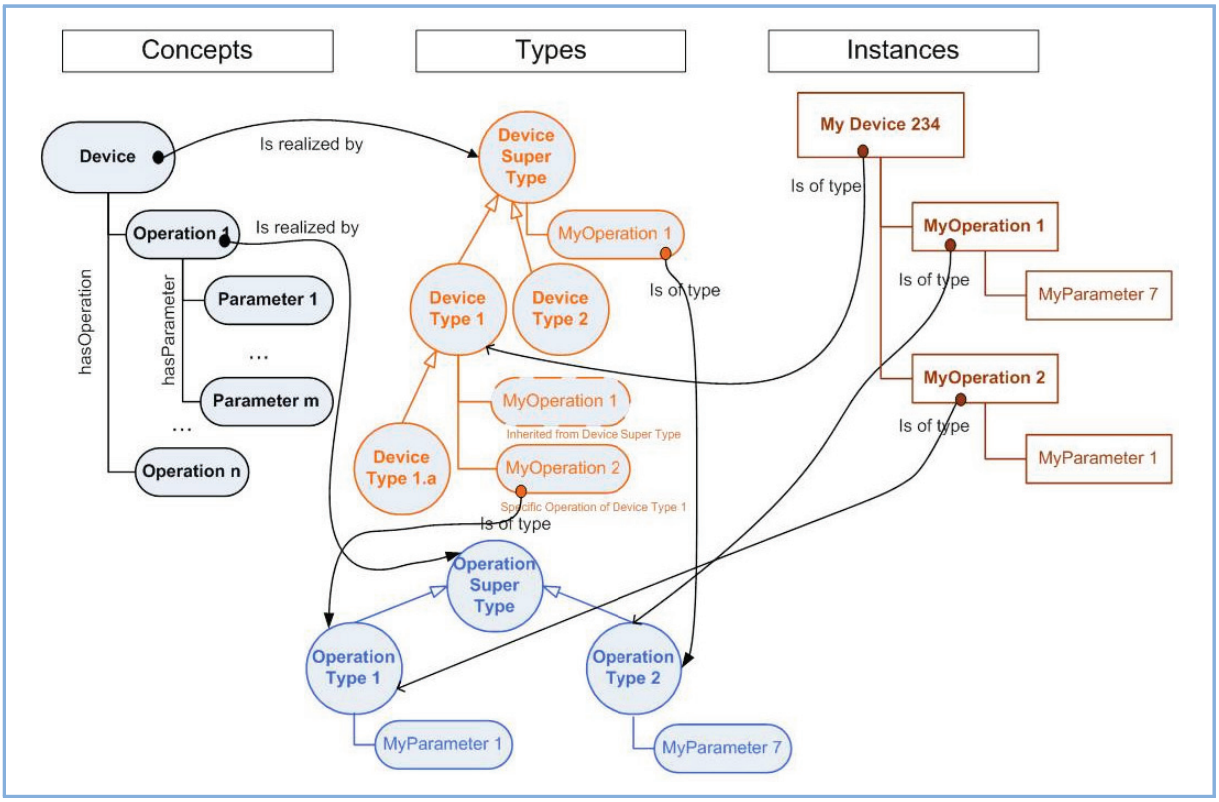



그림 〈7-11〉 HGI 디바이스의 예에 대한 개념, 유형 및 인스턴스에 대한 개요 및 상관관계

### 온톨로지를 이용한 디바이스 및 디바이스 템플릿 모델링

모델링은 2계층 접근 방식을 따른다. 상위 계층은 디바이스 유형을 모델링하기 위한 것이고 하위 계층은 디바이스 인스턴스를 모델링 하기 위한 것이다. 상위 계층은 온톨로지의 클래스 및 속성으로 모델링된 주어진 디바이스 유형 템플릿을 제공한다. 이를 바탕으로 디바이스 유형은 이 온톨로지의 클래스와 속성의 인스턴스로서 모델링된다. 하위 계층은 상위 계층에서 템플릿 클래스의 인스턴스로서 모델링 되었던 동일한 디바이스 유형이 디바이스 온톨로지의 클래스로서 해석된다. 실제 개개의 디바이스는 이런 클래스의 인스턴스 생성을 통해 모델링 될 수 있다.

다음은 디바이스 유형이 디바이스 유형 템플릿에 따라 어떻게 모델링 되고, 각 개개의 디바이스가 디바이스 유형에 따라 어떻게 모델링 되는지 예를 보여준다. 그림 〈7-12〉는 클래스와 관련 인스턴스로 구성되어 상위 계층과 하위 계층을 구성하는 3가지 부분을 보여준다.

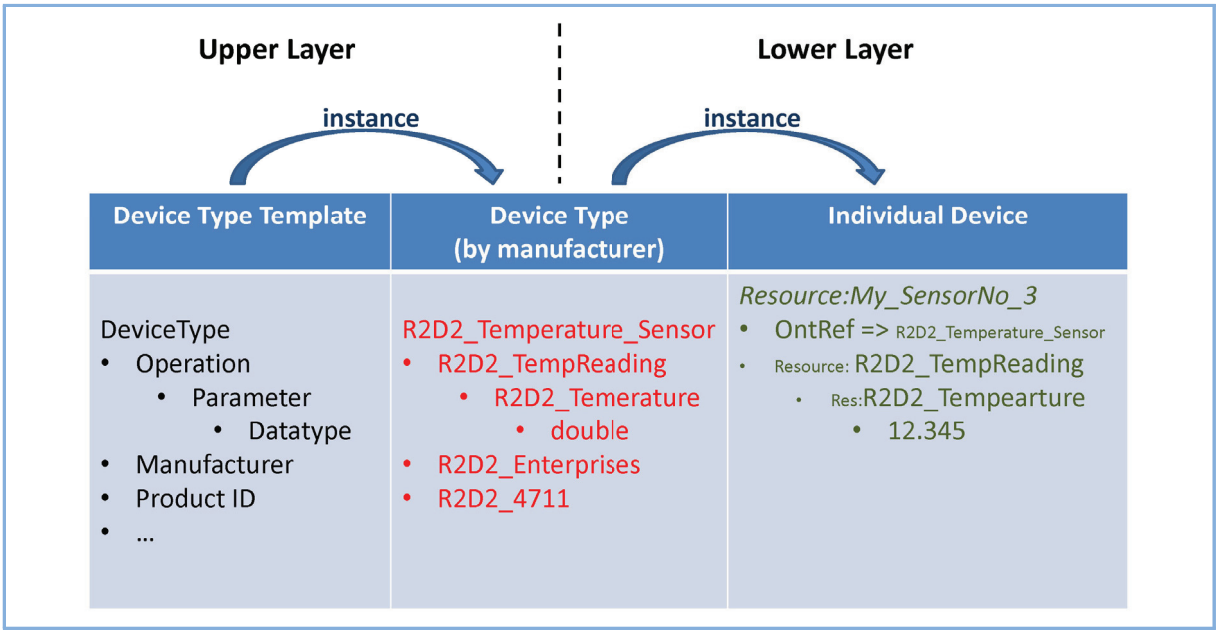



그림 〈7-12〉 디바이스 유형 템플릿, 디바이스 유형 및 개별 디바이스의 모델링

다음에서 좀 더 자세히 디바이스 템플릿을 기술하는 온톨로지 클래스 표 〈7-4〉와 속성 표 〈7-5〉를 보여준다. 여기서 적절한 서브 클래스가 공통 클래스의 특수성을 모델링 하기 위해 사용될 수 있다. 이 경우 다른 유형의 오퍼레이션을 구분을 하기 위해서 사용된다.

파라미터의 클래스는 InputParameter와 OutputParameter로 나누어진다.

InputParameter는 오퍼레이션의 입력과 관련되고, OutputParameter는 오퍼레이션의 출력과 연계된다.

표 〈7-4〉 디바이스 템블릿을 위한 온톨로지 클래스

| Class => SubClass | 설명 |
|---|---|
| DeviceType | 서로 같은 디바이스의 클래스에 대한 제조업자 정의 name/ID 이다. |
| Operation | 디바이스의 오퍼레이션을 식별한다. |
| => OutputOperation | 오퍼레이션이 출력 메시지만을 만들어된다. 디바이스는 서로 관련된 입력/ACK를 예상할 수 없다. |
| => InputOperation | 오퍼레이션은 입력 메시지 만으로 되어있다. 디바이스는 서로 관련된 입력/ACK를 만들어낼 수 없다. |
| => In-OutOperation | 오퍼레이션은 앱력 메시를 받아서 관련된 출력/ACK를 만들어낸다. |
| Parameter | 오퍼레이션의 파리미터를 식별한다. |
| => InputParameter | 오퍼레이션의 입력과 관련된 파라미터를 식별한다. |
| => OutputParameter | 오퍼레이션의 출력과 관련된 파라미터를 식별한다. |
| DataType | 파라미터의 데이터 유형 (예, xsd: double)을 식별한다. |
| Manufacturer | 제조업자의 Name/ID |
| Product ID | 디바이스의 유형을 식별하기 위한 제조업자 정의 handle/ID (예, Type/Model-number) |

표 〈7-5〉 디바이스 템플렛에 대한 오브젝트 및 데이터유형 속성

| Domain | Property | Range | |
|---|---|---|---|
| DeviceType | hasOperation | Operation | Object Properties |
| Operation | hasParameter | Parameter | |
| Parameter | hasParameterType | Datatype | |
| DeviceType | hasManufacturer | xsd:string | Datatype Properties |
| DeviceType | hasProductID | xsd:string | |

다음 표는 R2D2_Temperature_Sensor의 개별 디바이스 인스턴스를 기술하기 위한 온톨로지 클래스 표 〈7-6〉과 속성 표 〈7-7〉를 보여준다. 슈퍼클래의 계층 구조가 정의(Temperature_Sensor) 되고 디바이스 슈퍼클래스가 소개되고, 서로 다른 제조업체의 여러 유형의 온도 센서를 발견하는데 유용하게 사용될 수 있다. 또한 응용에 표준화된 인터페이스를 제공하는 추상화 온도 센서가 R2D2_Temperature_Sensor와 동일 레벨(즉, TemperatureSensor의 서브클래스) 혹은 별도의 계층구조로 소개된다. 별도 계층 구조에서는 구체적인 온도 센서와의 링크가 명시적으로 모델링 된다.

표 〈7-6〉 R2D2_Temperature_Sensor에 대한 온톨로지 클래스

| Class => SubClass | 설명 |
|---|---|
| Device<br>=> Temperature_Sensor<br>=> R2D2_Temperature_Sensor | R2D2_Temperature_Sensor 인스턴스의 특정 인스턴스를 위한 사용자 정의 name/ID<br>(예, My_SensorNo_3) |
| R2D2_TempReading | R2D2_Temperature_Sensor 인스턴스의 특정 오퍼레이션 |
| R2D2_Temperature | R2D2_TempReading 인스턴스의 특정 파라미터 |
| Metadata | R2D2_Temperature의 값과 관련된 메타데이터 |

표 〈7-7〉 R2D2_Temperature_Sensor에 대한 오브젝트 및 데이터타입 속성

| Domain | Property | Range | |
|---|---|---|---|
| R2D2_Temperature_Sensor | hasTemperatureOperation | R2D2_TempReading | Object Properties |
| R2D2_TempReading | hasTemperatureParameter | R2D2_Temperature | |
| R2D2_Temperature | hasMetadata | Metadata | |
| R2D2_Temperature | hasValue | xsd:double | Datatype Properties |

다음은 일반적으로 많이 사용되는 RDF/XML 주석 대신에 좀 더 간결하고 읽기 쉬운 Turtle 표현을 이용하여 OWL에서 R2D2_Temperature_Sensor의 개별 디바이스 인스턴스가 어떻게 모델링 되는지는 보여주는 예이다.

```
Device Instance (in OWL/Turtle)
    @prefix : 〈http://InstanceOntology#〉 .
    @prefix owl: 〈http://www.w3.org/2002/07/owl#〉 .
    @prefix rdf: 〈http://www.w3.org/1999/02/22-rdf-syntax-ns#〉 .
    @prefix xsd: 〈http://www.w3.org/2001/XMLSchema#〉 .
    @prefix dev: 〈http://DeviceOntology#〉 .
    @prefix dev-temp: 〈http://DeviceTemplateOntology#〉 .
    @base 〈http://InstanceOntology〉 .
    〈http://InstanceOntology〉      rdf:type          owl:Ontology;
                                   owl:imports       〈http://DeviceOntology〉
    :Resource:My_SensorNo_3        rdf:type          dev-temp:R2D2_Temperature_Sensor;
                                   owl:NamedIndividual;
                                   dev:hasTemperatureOperation:R2D2_TempReading.
    :R2D2_TempReading              rdf:type          dev-temp:R2D2_TempReading;
                                   owl:NamedIndividual;
                                   dev:hasTemperatureParameter:R2D2_Temperature.
    :R2D2_Temperature              rdf:type          dev-temp:R2D2_Temperature;
                                   owl:NamedIndividual;
                                   dev:hasValue"23.45"^^xsd:double.
```

### 7.4.2. oneM2M 구조 고려사항

지금까지 진행된 표준화 과정에서 핵심 이슈는 추상화 및 시맨틱을 지원하기 위한 주요 요구사항을 정의하는 것이었고, 이를 바탕으로 궁극적으로 oneM2M에서 정의한 구조와 연계하여 시맨틱을 위한 새로운 구조를 잡는 것이다. 이런 측면에서 oneM2M 릴리즈1에서는 구조 측면의 주요 이슈를 살펴보고 크게 3가지 단계별 접근 방안을 제시하였다.

**레벨 1 – 시맨틱 에노테이션:** 첫 번째 구조 옵션는 시맨틱 지원에 oneM2M 플랫폼 내에 시맨틱 에노테이션을 가지는 것으로 제한되는 경우다. oneM2M 릴리즈 1 규격에 정의해 둔 ontologyRef를 사용할 수 있다. 여기서 ontologyRef는 각자의 정보를 표현하기 위하여 사용되는 온톨로지를 식별하는 URI이다. 응용이 ontologyRef 속성 정보를 읽고 시맨틱 정보를 확인하고 추가적인 정보(예, 시맨틱 유형, 데이터 구조 및 다른 정보와의 관계, 등)를 검색하기 위하여 URI를 사용한다.

그림 〈7-13〉은 시맨틱 에노테이션이 어떻게 이용될 수 있는지를 보여준다. 온톨로지 레퍼런스는 일종의 시맨틱 인프라의 부분인 온톨로지를 가리킨다. 응용은 온톨로지 레퍼런스를 읽고 시맨틱 인프라로부터 좀 더 많은 정보를 액세스하기 위하여 이를 활용한다.

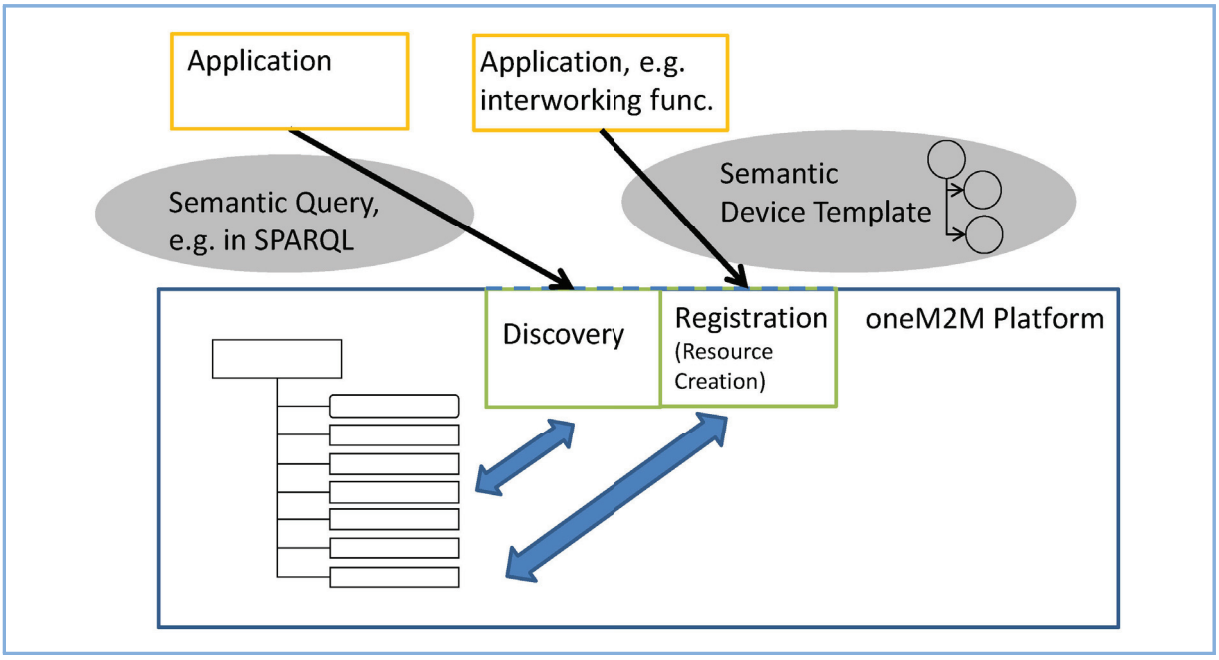



그림 〈7-13〉 시맨틱 에노테이션

이 구조에서는 플랫폼 내에서는 어떤 시맨틱 기능도 제공하지 않지만 시맨틱 에노테이션 기능을 가지고 예를 들면 인덱스 구조를 생성하거나 추론을 위하여 응용이 플랫폼 상위에 있는 시맨틱 기능을 이용할 수 있게 하는 기본적인 단계이다.

**레벨 2 – 플랫폼 기능으로서 시맨틱 기술을 이용:** 두 번째 구조 옵션은 시맨틱을 이용하기 위한 일부 기능이 oneM2M 플랫폼 내에 들어간다. 하지만 모든 시맨틱 기능이 플랫폼 내부에 들어가는 것이 아니라 일부는 인터페이스를 통해 외부와 상호작용한다.

이 옵션에서는 명시적으로 특정 기능을 지원하기 위하여 시맨틱 모델링이 목표가 된다. 이것은 종종 특정 oneM2M 플랫폼을 고려하기 때문에 기존 온톨리지는 외부에서 사용될 수 없음을 의미한다. 시맨틱 확장 기능을 위해 발견 기능과 리소스 생성의 기본으로서 디바이스 템플릿을 모델링하기 위해 시맨틱을 이용하는 두가지 예가 있다.

의미있는 발견 기능이 가능하도록 하기 위하여, 쿼리가 시맨틱 형태로 만들어질 수 있다. 이것은 SPARQL 같은 기존 쿼리 언어거나 oneM2M에 특정된 언어가 될 수 있다. 쿼리 결과는 oneM2M 리소스를 가리킨다.

시맨틱 모델링은 특정 디바이스 인스턴스를 표현하는 리소스의 구조를 정의하기 위해 사용될 수 있다. 이런 모델의 장점은 개념과 관계가 명시적으로 모델링될 수 있고 이후에 발견 기능과 같은 다른 측면을 위하여 재사용될 수 있다. 시맨틱 모델은 기존 온톨로지의 일반적인 부분이 될 수 있지만 핵심적인 것은 oneM2M 리소스 구조에 정확하게 맞아야 한다.

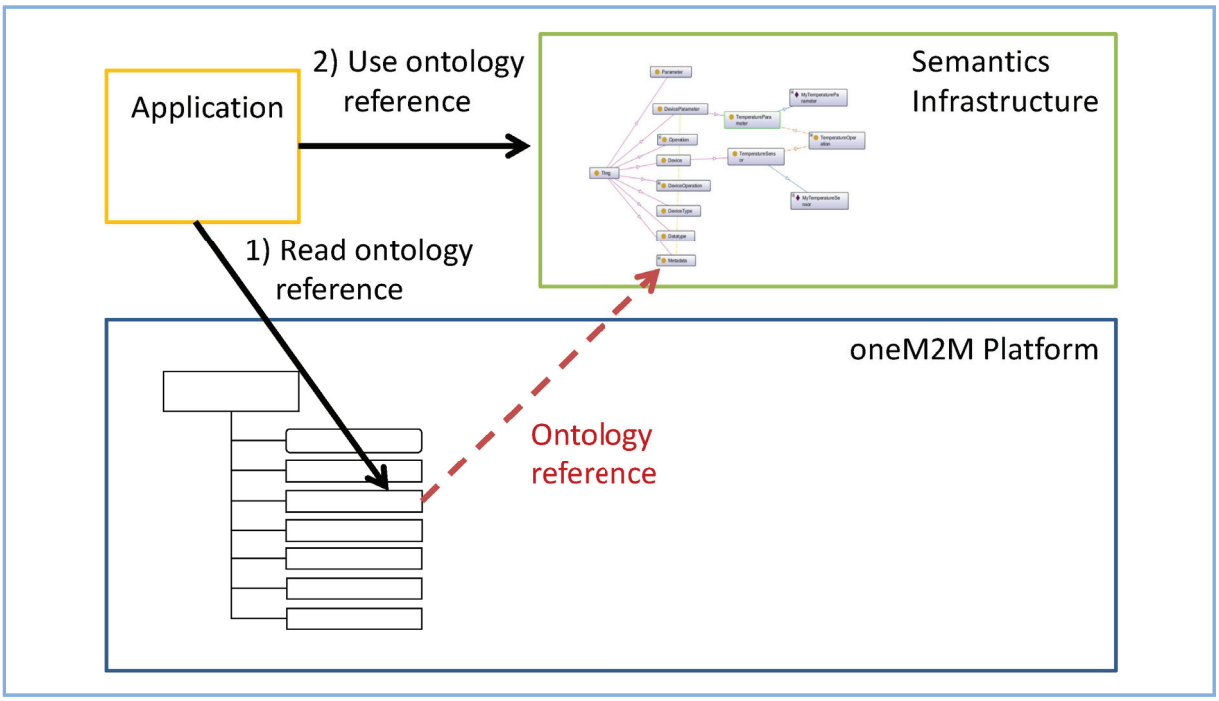



그림 〈7-14〉 플랫폼 기능으로서 시맨틱 기술을 이용

그림 〈7-14〉는 시맨틱 엘리멘트로서 두 가지 기능(즉, 발견, 등록(리소스 생성))을 가진 플랫폼을 보여준다. 응용은 시맨틱 기능을 지원하기 위하여 인터페이스를 통해

이들 기능과 상호작용한다. 발견의 경우 시맨틱 쿼리가 사용되고, 등록의 경우 시맨틱 모델링된 디바이스 템플릿을 이용한다.

**레벨 3 – 완전 시맨틱 플랫폼:** 완전 시맨틱 플랫폼 구조 옵션에서는 전체 플랫폼이 시맨틱 형태로 모든 측면을 다룬다. 모든 정보가 온톨로지 기반이고 가능한 기존 온톨로지를 이용한다. 특정 oneM2M 인터페이스를 가지는 대신에 공통 시맨틱 인터페이스와 툴이 플랫폼과 상호작용을 위하여 사용된 이런 시맨틱 oneM2M 플랫폼은 시맨틱 웹과 같은 기존 시맨틱 플랫폼과 쉽게 통합될 수 있다.

그림 〈7-15〉는 완전 시맨틱 구조 옵션에 대한 개요를 나타낸다. 응용은 시맨틱 인터페이스를 통해 플랫폼과 상호작용한다. 추론 엔진과 같은 일반적인 시맨틱 기능은 추가적인 정보를 얻을 수 있도록 해야하는데, 이런 정보는 시맨틱 웹과 쉽게 상호 링크 될 수 있다. 기존 온톨로지는 가능한한 다시 사용 가능하다.

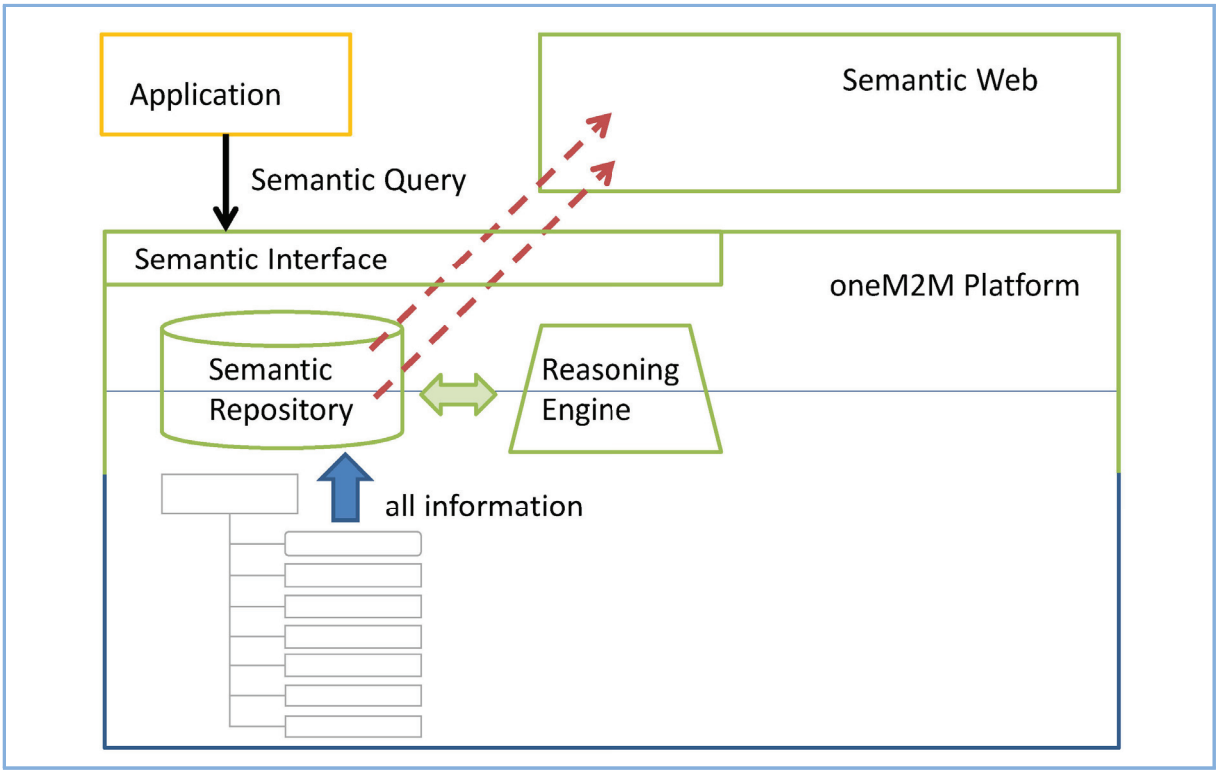



그림 〈7-15〉 완전 시맨틱 플랫폼

현재 oneM2M 릴리즈 1에서는 가장 기본적인 레벨에서 추상화 및 시맨틱이 가능한 수준의 구조를 제시하고 있다. 한 가지 예로 Non oneM2M 디바이스와 릴리즈1에 있는 지역 네트워크 (Area Network)와의 연동을 생각할 수 있다.

oneM2M 릴리즈 1에서 시맨틱 지원은 지역 네트워크와 non oneM2M 디바이스와의 연동을 지원하기 위하여 릴리즈 1에 이미 사용될 수 있는 시맨틱의 일부 제한된 측면만을 고려한다. 릴리즈 1에 사용하도록 제한되어 온톨로지에 포함될 구조 정보 (개념, 상관관계) 정도만이 될것이고, 이는 응용 엔티티(AEs)와 컨테이너의 "ontologyRef"를 통해 구현될 수 있다. 릴리즈1에 대한 이런 접근방식은 발견, 추론 혹은 메쉬업과 같은 많은 동적 시맨틱 정보를 사용하는데 아직 한계가 있다.

기본적인 기능에 대한 개요를 그림 〈7-16〉에서 보여준다. 그림 〈7-16〉은 M2M 응용(예, 유틸리티 응용)이 oneM2M 시스템에 있는 인터워킹 프록시를 통해 연동하는 어떻게 non-oneM2M 디바이스(예, MBus/OCSEM 혹은 ZigBee 기술을 탑재한 센서/미터)를 액세스할 수 있는지에 대한 예를 보여준다.

이 예에서 보여준 시나리오는 한 노드 내에 두 개의 인터워킹 프록시를 가진 것으로 설계되었다. 다른 시나리오에서는 두 개 네트워크 담당하기 위해 단지 하나의 인터워킹 프록시를 가지는 형태도 고려할 수 있다.

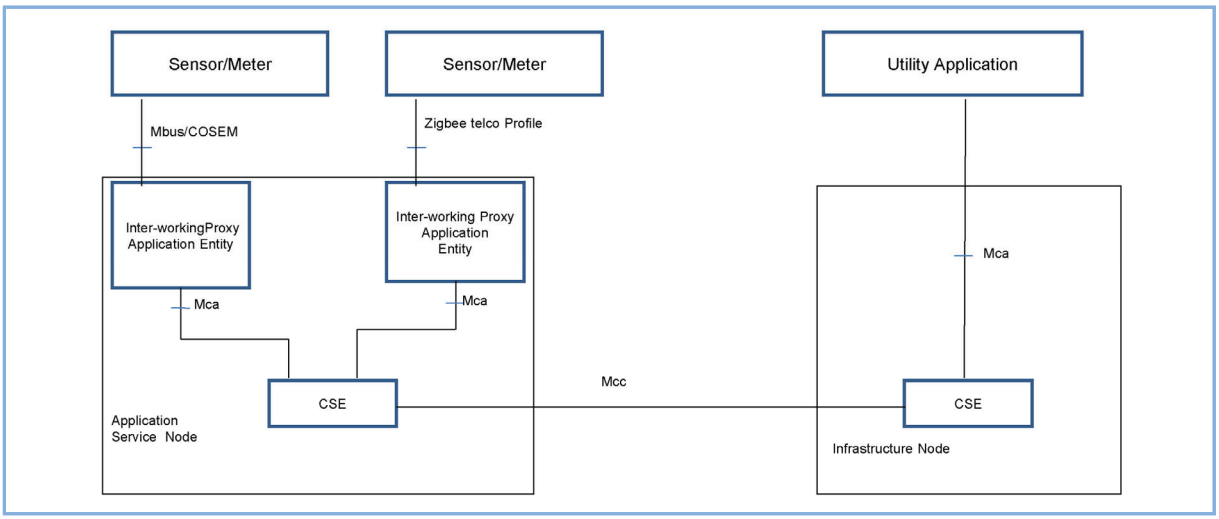



그림 〈7-16〉 인터워킹 프록시 응용을 이용한 연동 시나리오

## 7.5. 요약

지금까지 릴리즈 1의 핵심 결과물 중의 하나로 작업된 ABS TR의 주요 내용을 살펴보았다. oneM2M ABS TR에서는 시맨틱에 관련된 유스 케이스를 분석하고, 추상화와 시맨틱에 관련된 기술 및 요구사항 분석으로 통하여 oneM2M 시스템에 관련 기술을 어떻게 적용할 것인지에 대한 상세 내용을 담고 있다.

향후 릴리즈 2 표준 문서 개발에 있어 현재 개발 중인 추상화 및 시맨틱 기술 보고서는 상세 요구사항 명세, oneM2M과 연계된 구조 개발 및 이를 바탕으로 기술 스펙 개발에 역점을 두고 추가 표준화 작업이 진행될 예정이다.

온톨로지는 특정한 영역을 표현하는 데이터 모델로서 특정한 영역(Domain)에 속하는 개념과, 개념 사이의 관계를 기술하는 정형(Formal) 어휘의 집합으로 정의된다.

OWL(Web Ontology Language)은 관계들 간의 hierarchy, 관계 인스턴스 내에서의 논리적 제약조건 등을 포함한 언어로, 정밀하고 논리적인 추론을 필요로 하는 경우에 사용한다.

RDF(Resource Description Framework)는 XML에서 발전한 형태이며, subject, object, predicate으로 이루어지며, 단순하게 개념 혹은 인스턴스 사이의 관계를 나타낸다.

# oneM2M 서비스 플랫폼 표준 해설서

발행인 : 임차식
발행처 : 한국정보통신기술협회
463-824, 경기도 성남시 분당구 분당로 47
Tel : 031-724-0114, Fax : 031-724-0109
발행일 : 2014. 11.

oneM2M 서비스 플랫폼 표준 해설서

TTA 한국정보통신기술협회 Telecommunications Technology Association
경기도 성남시 분당구 분당로 47
(031)724-0114, http://www.tta.or.kr