

# TTA Standard

정보통신단체표준(국문표준)  
TTAS.KO-06.0169 /R1

제정일: 2007년 12월 26일  
개정일: 2009년 6월 18일

센서 네트워크 공통 인터페이스

(The Standard Interface for Heterogeneous  
Sensor Networks)



## 센서 네트워크 공통 인터페이스

(The Standard Interface for Heterogeneous Sensor Networks)



본 문서에 대한 저작권은 TTA 에 있으며, 이 문서의 전체 또는 일부에 대하여 상업적 이익을 목적으로 하는 무단 복제 및 배포를 금합니다.

Copyright© Telecommunications Technology Associations(2007). All Rights Reserved.

# 서 문

## 1. 표준의 목적

유비쿼터스 센서 네트워크는 IT839 3대 인프라 중 하나로서 다양한 센서 네트워크와 이를 사용하는 응용 서비스들의 유기적인 결합으로 사용자 중심 컴퓨팅 환경을 만들어내는 인프라이며 기술이다. 사용자에게 서비스를 제공하는 다양한 응용이 복수의 이기종 센서 네트워크로부터 정보를 수집하고 센서 네트워크를 관리하기 위해서는 표준화된 센서 네트워크에 대한 접근 인터페이스가 필요하다. 본 표준은 다양한 응용이 하나의 혹은 다수의 이기종 센서 네트워크와 통신하기 위해서 요구되는 표준 인터페이스 규격을 정의한다.

## 2. 주요 내용 요약

본 표준의 주요 내용은 이기종 센서 네트워크와 이를 이용하는 호스트간 통신 프로토콜을 정의하며, 통신 프로토콜에 이용되는 메시지를 정의한다. 또한 센서 데이터의 유형별 표준 타입을 정의함으로써 서로 다른 측정 방법에 의해서 제공되는 동일한 센싱 유형에 대한 표준 단위 및 데이터 타입을 정의한다. 본 표준에 정의된 통신 프로토콜, 메시지 포맷, 표준 센싱 타입값을 통해서 센서 네트워크 추상화가 제공된다. 표준은 추가 기능 반영이 용이하도록 설계되었다.

## 3. 표준 적용 산업 분야 및 산업에 미치는 영향

본 표준은 서비스 별로 다양하게 구축되는 센서 네트워크를 표준화된 인터페이스를 통해서 접근하게 함으로써, 센서 네트워크에 대한 효율성을 제고할 것으로 기대한다. 또한 응용 서비스 개발을 용이하게 함으로써, 다양한 콘텐츠를 제공하는 서비스 개발을 촉진시켜서 USN 산업 활성화를 앞당기는데 기여할 것이다.

## 4. 참조 표준(권고)

### 4.1 국외표준(권고)

- 해당 사항 없음

### 4.2 국내표준

- TTAS.KO-06.0169 센서 네트워크 공통 인터페이스 (2007-12-26)

**5. 참조표준(권고)과의 비교**

- 국내표준(TTAS.KO-06.0169)은 2007년 12월 TTA를 통해서 제정된 센서네트워크공통인터페이스 표준이며, 본 표준은 TTA표준에 대한 개정의 의미를 갖음. Xml 인터페이스를 포함하며, 모니터링, 인증, 센서네트워크 제어관련 인터페이스가 추가되었음.

**6. 지적재산권 관련사항**

- 본 표준의 '지적재산권 요약서' 제출 현황은 TTA 웹사이트에서 확인할 수 있다.

**7. 적합인증 관련사항**

**7.1 적합인증 대상 여부**

- 해당사항 없음

**7.2 시험표준제정여부(해당 시험표준번호)**

- 해당사항 없음

**8. 표준의 이력**

판수	제/개정일	제.개정내역
제1판	2007.12.26	제정
제2판	2009.6.18	개정

# Preface

## 1. The Purpose of Standard

Ubiquitous Sensor Network is the infrastructure and technology that provides user-centered computing environment, where various types of sensor networks and applications are cooperating harmoniously, as the one of IT 839 infrastructure. To collect sensor data and to manage those sensor networks, there need some standard interfaces between various kinds of sensor networks and hosts. This standard defines the common interface between heterogeneous sensor network and host.

## 2. The summary of contents

The main contents of this standard are communication protocol, message formats between sensor network and host. In addition, it defines some standard sensor data types. By using this communication protocol, message formats and sensor data types, this standard provides sensor network abstraction. This standard is designed to be extended easily for the future available functions.

## 3. Applicable fields of industry and its effect

This standard makes various kinds of sensor network application development easy. By facilitating sensor network application development, this standard will contribute greatly to Ubiquitous Sensor Network Industry.

## 4. Reference Standards (Recommendations)

### 4.1 International Standards (Recommendations)

- Nothing is related

### 4.2 Domestic Standards

- TTAS.KO-06.0169 'The Standard Interface for Heterogeneous Sensor Networks '(2007-12-26)

**5. Relationship to Reference Standards (Recommendations)**

- This standard is the revised version of TTAS.KO-06.0169 approved by TTA December 2007. It extends the previous version to include XML-type message formats and additional messages for sensor network monitoring, sensor network authentication and sensor network control functions.

**6. The Statement of Intellectual Property Rights**

-IPRs related to the present document may have been declared to TTA. The information pertaining to these IPRs, if any, is available on the TTA Website.

**7. The Statement of Conformance Testing and Certification**

- Nothing is related

**8. The History of Standard**

Edition	Issued date	Contents
The 1 <sup>st</sup> edition	2007.12.26	Established
The 2 <sup>nd</sup> edition	2009.6.18	Revised

## 목 차

1. 표준의 목적 .....	1
2. 주요 내용 요약 .....	1
3. 표준 적용 산업 분야 및 산업에 미치는 영향 .....	1
4. 참조 표준(권고).....	1
4.1 국외표준(권고) .....	1
4.2 국내표준.....	1
5. 참조표준(권고)과의 비교.....	2
6. 지적재산권 관련사항 .....	2
7. 적합인증 관련사항 .....	2
7.2 시험표준제정여부(해당 시험표준번호) .....	2
8. 표준의 이력 .....	2
1. The Purpose of Standard .....	3
2. The summary of contents .....	3
3. Applicable fields of industry and its effect.....	3
4. Reference Standards (Recommendations) .....	3
4.1 International Standards (Recommendations).....	3
4.2 Domestic Standards .....	3
5. Relationship to Reference Standards (Recommendations).....	4
6. The Statement of Intellectual Property Rights.....	4
7. The Statement of Conformance Testing and Certification.....	4
8. The History of Standard.....	4
1. 개요 .....	1 3
2. 표준의 구성 및 범위 .....	1 3
3. 정의 .....	1 4
3.1. 용어 정의 .....	1 4
4. 센서 네트워크 공통 인터페이스 개요.....	1 6
4.1 센서 네트워크 공통 인터페이스 기능 요구 사항 .....	1 7
4.2. 센서 네트워크 통신 프로토콜 .....	1 7
4.2.1 호스트와 센서 네트워크간 연결 설정 .....	1 8
4.2.1.1 SN-initiated 모드 .....	1 9
4.2.1.2 host-initiated 모드 .....	1 9
4.2.2 통신 채널 .....	1 9
4.3 센서네트워크 공통 메시지.....	2 0
4.3.1 메시지 유형별 코드 .....	2 0
4.3.1.1 메시지유형(MessageType).....	2 0
4.3.1.2 메시지 유형별 코드 값.....	2 1
4.3.1.3 주소 표현 방법 .....	2 2

4.3.2 메시지의 종류 .....	2 3
4.3.2.1 연결제어 메시지 .....	2 3
4.3.2.1.1 연결설정 및 해지 .....	2 3
4.3.2.1.2 채널인증 .....	2 5
4.3.2.1.3 연결 제어 전체 시나리오 .....	2 7
4.3.2.2 요청/응답 메시지 .....	2 7
4.3.2.2.1 네트워크 정보 요청, 응답 .....	2 7
4.3.2.2.2 버퍼링 메시지 요청, 응답 .....	2 8
4.3.2.2.3 명령 제어 요청, 응답 .....	2 9
4.3.2.2.4 명령 갱신 요청, 응답 .....	3 1
4.3.2.2.5 센서네트워크 제어 요청, 응답 .....	3 3
4.3.2.2.6 노드 제어 요청, 응답 .....	3 5
4.3.2.2.7 요청에 대한 응답 코드 .....	3 6
4.3.2.3 명령 및 보고 메시지 .....	3 7
4.3.2.3.1 센싱 명령/보고 메시지 .....	3 7
4.3.2.3.1.1 일회성 명령과 보고 .....	3 8
4.3.2.3.1.2 연속성 명령과 보고 .....	4 1
4.3.2.3.1.3 일회성 이벤트 명령 .....	4 4
4.3.2.3.1.4 일회성 집계명령과 보고 .....	4 4
4.3.2.3.1.5 연속성 집계명령과 보고 .....	4 5
4.3.2.3.2 구동기 명령/보고 메시지 .....	4 6
4.3.2.3.3 모니터링 명령/보고 메시지 .....	4 8
4.3.2.3.4 에러보고 메시지 .....	5 1
4.3.2.3.5 에러보고 유형과 코드 값 .....	5 3
4.3.2.3.6 네트워크 갱신 보고 메시지 .....	5 3
4.3.2.4 확인 .....	5 4
4.3.2.4.1 채널확인 .....	5 4
4.3.2.4.2 오류 확인 .....	5 5
4.3.3 예약된 값 .....	5 6
4.3.3.1 노드/트랜스듀서 식별자 (8bytes) .....	5 6
4.3.3.2 명령 식별자 .....	5 6
4.3.3.3 시간의 표현 .....	5 6
5. 바이트 스트림 메시지 .....	5 7
5.1 메시지 구조 .....	5 7
5.2 공통 상수 .....	5 7
5.2.1 노드 유형 .....	5 8
5.2.2 트랜스듀서 유형 .....	5 8
5.2.3 트랜스듀서 연산 유형 .....	5 8
5.2.4 구동기 동작 유형 .....	5 8

5.2.5 구동기 동작 값 .....	5 8
5.2.6 연산자 .....	5 8
5.2.6.1 논리연산자 .....	5 8
5.2.6.2 관계연산자 .....	5 9
5.2.7 함수 .....	5 9
5.2.8 네트워크 동작유형 .....	5 9
5.2.9 배터리 정보 표현 유형 .....	5 9
5.2.10 모니터링 파라미터 .....	5 9
5.3 연결제어 메시지 .....	5 9
5.3.1 ReqConnCtrl .....	5 9
5.3.2 ConnReqCtrl .....	6 0
5.3.3 ConnResCtrl .....	6 0
5.3.4 DisConnReqCtrl .....	6 1
5.3.5 AuthReqCtrl .....	6 1
5.3.6 AuthResCtrl .....	6 2
5.4 요청/응답 메시지 .....	6 3
5.4.1 NetworkInfoReq .....	6 3
5.4.2 NetworkInfoRes .....	6 3
5.4.3 BufferDataReq .....	6 4
5.4.4 BufferDataRes .....	6 4
5.4.5 CmdActionReq .....	6 4
5.4.6 CmdActionRes .....	6 4
5.4.7 UpdateCmdReq .....	6 5
5.4.8 UpdateCmdRes .....	6 6
5.4.9 ControlNetworkReq .....	6 6
5.4.10 ControlNetworkRes .....	6 7
5.4.11 ControlNodeReq .....	6 7
5.4.12 ControlNodeRes .....	6 8
5.5 명령 메시지 .....	6 8
5.5.1 InstantCmd .....	6 8
5.5.2 ContinuousCmd .....	6 9
5.5.3 InstantEventCmd .....	7 0
5.5.4 InstantAggCmd .....	7 0
5.5.5 ContinuousAggCmd .....	7 1
5.5.6 RunActuatorCmd .....	7 2
5.5.7 MonitoringStartCmd .....	7 2
5.5.8 MonitoringStopCmd .....	7 3
5.6 보고 메시지 .....	7 4
5.6.1 SensingValueRpt .....	7 4

5.6.2 RunActuatorRpt.....	7 4
5.6.3 FinishRpt .....	7 5
5.6.4 MonitoringRpt .....	7 5
5.6.5 ErrorRpt.....	7 6
5.6.6 UpdateRpt.....	7 7
5.7 확인 메시지 .....	7 9
5.7.1 ChannelCheckCtrl .....	7 9
5.7.2 ChannelConfirmCtrl .....	7 9
5.7.3 NakChk.....	7 9
6. XML 문서 메시지 .....	8 0
6.1 메시지 구조 .....	8 0
6.2 HTTP 응답 .....	8 0
6.2.1 알 수 없는 클라이언트.....	8 0
6.2.2 유효하지 않은 메시지 .....	8 0
6.2.2.1 XML 문서가 포함되지 않은 경우 .....	8 0
6.2.2.2 포함된 XML 문서가 유효하지 않는 경우 .....	8 0
6.2.3 수신할 수 없는 유형의 메시지 .....	8 1
6.2.4 연결제어/확인 메시지 .....	8 1
6.2.4.1 연결설정 정보 요청 (ConnReqCtrl).....	8 1
6.2.4.2 채널인증 (AuthReqCtrl).....	8 1
6.2.4.3 채널확인 (ChannelCheckCtrl).....	8 1
6.2.5 요청/응답 메시지 .....	8 1
6.2.6 명령 메시지 .....	8 1
6.2.7 보고 메시지 .....	8 1
6.3 메시지 스키마 .....	8 2
7 표현 .....	1 0 4
7.1 값의 표현 .....	1 0 4
7.2 날짜와 시간 .....	1 0 4
7.2.1 TimeStamp.....	1 0 4
7.2.2 LifeTime.....	1 0 4
7.3 센싱값.....	1 0 4
부록 : 센서데이터유형정의(예제) .....	1 0 5

# Contents

1. Overview .....	1 3
2. Constitutions and Scope .....	1 3
3. Terms .....	1 4
3.1. Definitions .....	1 4
4. Sensor Network Common Interface Overview .....	1 6
4.1 Functional Requirements.....	1 7
4.2. Sensor Network Common Interface Protocol.....	1 7
4.2.1 Connection Establishment.....	1 8
4.2.1.1 SN-initiated mode .....	1 9
4.2.1.2 host-initiated mode .....	1 9
4.2.2 Channels .....	1 9
4.3 Messages .....	2 0
4.3.1 Message code .....	2 0
4.3.1.1 MessageType .....	2 0
4.3.1.2 Message code values .....	2 1
4.3.1.3 Addressing .....	2 2
4.3.2 Message Classification .....	2 3
4.3.2.1 Connection Control.....	2 3
4.3.2.1.1 Connection establishment/release .....	2 3
4.3.2.1.2 Channel authentication .....	2 5
4.3.2.1.3 Scenario.....	2 7
4.3.2.2 Request/Response .....	2 7
4.3.2.2.1 Network information .....	2 7
4.3.2.2.2 Buffered data .....	2 8
4.3.2.2.3 Command control .....	2 9
4.3.2.2.4 Command update .....	3 1
4.3.2.2.5 Sensor network control.....	3 3
4.3.2.2.6 Sensor node control.....	3 5
4.3.2.2.7 Response code .....	3 6
4.3.2.3 Command and Report .....	3 7
4.3.2.3.1 Sensing.....	3 7
4.3.2.3.1.1 Instant command/report.....	3 8
4.3.2.3.1.2 Continuous command/report .....	4 1
4.3.2.3.1.3 Instant event command/report .....	4 4
4.3.2.3.1.4 Instant aggregation command/report .....	4 4
4.3.2.3.1.5 Continuous aggregation command/report.....	4 5

4.3.2.3.2 Actuator command/report .....	4 6
4.3.2.3.3 Monitoring command/report .....	4 8
4.3.2.3.4 Error report .....	5 1
4.3.2.3.5 Error report types & codes .....	5 3
4.3.2.3.6 Network update report .....	5 3
4.3.2.4 Ack .....	5 4
4.3.2.4.1 Channel check .....	5 4
4.3.2.4.2 Nack .....	5 5
4.3.3 Reserved values .....	5 6
4.3.3.1 node/transducer identifier .....	5 6
4.3.3.2 command identifier .....	5 6
4.3.3.3 Time .....	5 6
5. Binary message formats .....	5 7
5.1 Message structure .....	5 7
5.2 Common constants .....	5 7
5.2.1 Node types .....	5 8
5.2.2 Transducer types .....	5 8
5.2.3 Operation types .....	5 8
5.2.4 Actuator operation type .....	5 8
5.2.5 Actuator operation value .....	5 8
5.2.6 Operator .....	5 8
5.2.6.1 Logical operator .....	5 8
5.2.6.2 Relational operator .....	5 9
5.2.7 Functions .....	5 9
5.2.8 Network operation mode .....	5 9
5.2.9 Battery information .....	5 9
5.2.10 Monitoring parameters .....	5 9
5.3 Connection Control .....	5 9
5.3.1 ReqConnCtrl .....	5 9
5.3.2 ConnReqCtrl .....	6 0
5.3.3 ConnResCtrl .....	6 0
5.3.4 DisConnReqCtrl .....	6 1
5.3.5 AuthReqCtrl .....	6 1
5.3.6 AuthResCtrl .....	6 2
5.4 Request/Response .....	6 3
5.4.1 NetworkInfoReq .....	6 3
5.4.2 NetworkInfoRes .....	6 3
5.4.3 BufferDataReq .....	6 4
5.4.4 BufferDataRes .....	6 4

5.4.5 CmdActionReq.....	6 4
5.4.6 CmdActionRes .....	6 4
5.4.7 UpdateCmdReq .....	6 5
5.4.8 UpdateCmdRes.....	6 6
5.4.9 ControlNetworkReq.....	6 6
5.4.10 ControlNetworkRes .....	6 7
5.4.11 ControlNodeReq.....	6 7
5.4.12 ControlNodeRes .....	6 8
5.5 Command.....	6 8
5.5.1 InstantCmd .....	6 8
5.5.2 ContinuousCmd .....	6 9
5.5.3 InstantEventCmd.....	7 0
5.5.4 InstantAggCmd .....	7 0
5.5.5 ContinuousAggCmd .....	7 1
5.5.6 RunActuatorCmd.....	7 2
5.5.7 MonitoringStartCmd.....	7 2
5.5.8 MonitoringStopCmd.....	7 3
5.6 Report.....	7 4
5.6.1 SensingValueRpt .....	7 4
5.6.2 RunActuatorRpt.....	7 4
5.6.3 FinishRpt .....	7 5
5.6.4 MonitoringRpt .....	7 5
5.6.5 ErrorRpt.....	7 6
5.6.6 UpdateRpt.....	7 7
5.7 Ack.....	7 9
5.7.1 ChannelCheckCtrl .....	7 9
5.7.2 ChannelConfirmCtrl .....	7 9
5.7.3 NakChk.....	7 9
6. XML message formats.....	8 0
6.1 Message structure .....	8 0
6.2 HTTP response.....	8 0
6.2.1 Unknown client .....	8 0
6.2.2 Invalid message .....	8 0
6.2.2.1 XML document is not included .....	8 0
6.2.2.2 Invalid XML document.....	8 0
6.2.3 Unsupported message type .....	8 1
6.2.4 Connection control/ack.....	8 1
6.2.4.1 ConnReqCtrl .....	8 1
6.2.4.2 AuthReqCtrl .....	8 1

6.2.4.3 ChannelCheckCtrl.....	8 1
6.2.5 Request/Response.....	8 1
6.2.6 Command.....	8 1
6.2.7 Report.....	8 1
6.3 Message schema.....	8 2
7 Expression.....	1 0 4
7.1 Value.....	1 0 4
7.2 Date & Time.....	1 0 4
7.2.1 TimeStamp.....	1 0 4
7.2.2 LifeTime.....	1 0 4
7.3 Sensing value.....	1 0 4
Appendix : Sensing Type table(example).....	1 0 5



# 센서 네트워크 공통 인터페이스

## The Standard Interface for Heterogeneous Sensor Networks

### 1. 개요

홈 네트워크, 텔레매틱스, RFID, 센서 네트워크 등 현재 다양한 분야에서 유비쿼터스 컴퓨팅 세상을 만들기 위한 노력이 진행되고 있다. 유비쿼터스 컴퓨팅 환경에서는 사용자를 둘러싼 모든 환경이 모두 정보 제공자이면서 정보 가공자가 되고, 정보 수요자가 된다. 물리적 환경을 이루는 온도, 습도, 압력, 염도, 광원 등에 대한 다양한 센서가 존재하며 이들 센서가 장착된 기기 별로 통신 프로토콜이 다양하며, 물리적 특성 또한 다양하다. 연구소, 학계, 업계 등 다양한 단체에서 다양한 유무선 센서, 구동기를 효과적으로 이용하여 보다 지능적인 서비스를 제공할 수 있도록 센서의 하드웨어로부터, 통신 프로토콜, 센서 네트워크 관리, 효과적인 센싱 정보 수집/처리/가공 등에 이르는 연구가 진행 중이다. 센서 네트워크로부터 정보를 수집하고 처리하고 가공하기 위해서는 센서 네트워크와의 통신 인터페이스가 필요하고 현재는 각 센서 네트워크 특성 별 혹은 이를 이용하는 서비스의 목적 별로 응용 의존적인 인터페이스가 정의되어 사용되어 왔다. 결국 이러한 개발 방향은 과도한 중복 투자를 유발하여 효과적인 센서 네트워크 설치 및 운용에 큰 저해 요인이 되어 왔다.

따라서 본 제안에서는 현재 서비스 별로 구축된 로컬화된 센서 네트워크 인터페이스를 대체할 수 있는 표준화된 센서 네트워크 인터페이스를 제안함으로써, 제안된 인터페이스를 준수한다면 서로 다른 특성의 센서 네트워크를 효과적으로 다양한 응용 서비스들이 이용할 수 있게 된다.

### 2. 표준의 구성 및 범위

본 표준은 다양한 센서 네트워크와의 일관된 통신을 제공하기 위한 인터페이스를 제공하는데 있어서 필요한 기술적 요구 사항을 명시하고, 요구 사항이 반영된 인터페이스를 정의하고 있다. 본 인터페이스는 프로토콜 및 메시지 형식을 정의하며, 센싱 타입에 대한 표준 단위 및 데이터 타입을 정의한다.

### 3. 정의

#### 3.1. 용어 정의

##### 노드 식별자 (Node Identifier)

센서네트워크내의 다수의 센서노드를 유일하게 구분하기 위한 식별자

##### 동기식 처리 방식(synchronous processing method)

호스트의 요청처리방식으로, 센서네트워크에 대한 처리요청(제어)을 전송하고, 응답을 기다리는 처리 방식

##### 명령(command)

센서네트워크 공통인터페이스로 전달되는 메시지의 유형으로, 센싱센싱정보요청/모니터링정보요청을 포함

##### 명령채널(command channel)

공통메시지중 명령, 보고, 확인, 채널 확인 메시지 교환에 사용되는 채널

##### 메시지(message)

호스트와 어댑터간 교환되는 메시지

##### 메타데이터 (Meta-Data)

센서네트워크에 대한 정보를 기술하기 위한 데이터로 센서네트워크의 구성현황, 센서네트워크의 능력(Capability), 센서네트워크 식별자, 센싱값유형 식별자 등을 포함

##### 보고(report)

센서네트워크로부터 센싱값이나 현재 상태를 호스트로 전달할때 사용되는 메시지

##### 비동기식(asynchronous processing method)

호스트의 요청처리방식으로, 센서네트워크에 대한 처리요청(제어)을 전송하고, 응답을 기다리지 않고 다른 요청을 처리 할 수 있는 처리 방식. 요청 처리시 시간이 센싱정보 수집 요청의 경우 비동기식으로 처리

##### 센서네트워크 동작 모드

호스트와 센서네트워크간에 센싱값 및 정보를 주고받는 동작 방식으로 Push 모드와 Pull 모드를 칭함

##### 센서네트워크 식별자 (Sensor network Identifier)

USN 미들웨어 및 응용에 의해 유일하게 센서네트워크를 구분하기 위한 사용되는 식별자

**센싱값 유형 식별자 (Sensing Type Identifier)**

센서에 의해 측정되는 값의 유형에 부여되는 식별자로, 온도, 습도와 같은 센싱 유형과 측정단위(섭씨, 화씨, etc)가 맵핑된 식별자

**어댑터(adaptor)**

센서네트워크단에서 센서네트워크공통인터페이스를 처리하는 기능을 갖는 소프트웨어 모듈

**요청(request)**

센서네트워크상태정보, 실행중인 명령에 대한 제어등 동기식처리가 필요한 호스트로부터의 요청

**연결설정 정보 요청자**

센서네트워크와 호스트간 연결설정을 요청하는 센서네트워크단의 논리적 모듈로 어댑터 혹은 별도 모듈이 이에 해당됨

**연결설정 채널(connection channel)**

호스트와 연결설정정보요청자간 연결설정을 위해서 사용되는 채널

**연속성 명령(Continuous Command)**

센싱명령의 유형 중 주기에 기반한 지속적 응답을 요청하기 위해 lifetime, interval이 주어지는 명령

**응답(response)**

요청(request)에 대한 응답

**이벤트 명령(Event Command)**

센싱명령중 조건이 주어지는 명령

**일회성 명령(Instant Command)**

센싱명령중 일회성으로 센싱을 요청하는 명령

**제어채널(control channel)**

호스트와 어댑터간 요청, 응답, 확인, 채널 확인 메시지가 교환되는 동기식 채널

**질의(query)**

호스트의 센서네트워크에 대한 센싱 및 구동기 연산 요청

**집계 명령(Aggregation Command)**

센싱명령중 집계에 대한 요청 명령

**채널(channel)**

호스트와 어댑터간 정보교환을 주고받는 통신 방법

**트랜스듀서(Transducer)**

센서노드에 탑재될 수 있는 센서 또는 구동기

#### Push 모드 센서네트워크 동작

호스트와 센서네트워크에 있어 명시적 요청없이 센서네트워크 동작 설정값에 따라 센싱값 값을 보고하는 동작 방식

#### Pull 모드 센서네트워크 동작

호스트와 센서네트워크에 명시적 요청없이 센서네트워크 동작 설정값에 따라 센싱값 값을 보고하는 동작 방식

#### 하드웨어 스펙 ID (Hardware Specification Identifier, HwSpecID)

노드, 센서, 구동기등에 대한 하드웨어 사양을 식별하는 식별자

#### 확인 메시지(check message)

제어채널, 명령채널의 연결상태를 확인하기 위한 메시지

#### 호스트

센서네트워크를 이용하여 특정 서비스를 제공하고자하는 노드로 일반 응용 프로그램 혹은 USN 미들웨어

## 4. 센서 네트워크 공통 인터페이스 개요

센서 네트워크 공통인터페이스는 호스트와 센서 네트워크간 교환될 공통 메시지를 표준화된 규격으로 정의함으로써, 이기종 센서 네트워크에 대한 추상화 기능을 제공하는 표준 인터페이스이다. 호스트는 센서 네트워크를 이용하여 서비스를 제공하고자 하는 응용 서비스이거나 혹은 응용 서비스와 센서 네트워크 간의 연결, 질의 및 센싱 데이터 전달, 센싱 데이터 처리 등의 기능을 제공하는 미들웨어일 수 있다.

센서 네트워크 공통 인터페이스에서 정의한 기능들은 센서 네트워크에 대한 메타데이터가 관리되고 있다는 전제로 정의된다. 센서 네트워크에 관련된 메타데이터는 센서 네트워크 노드의 하드웨어적/소프트웨어적 특성, 통신 프로토콜 정보, 토폴로지, 노드 수, 구동기 수, 센싱값 유형 등을 포함하며, 센서 네트워크의 메타데이터를 저장/검색/수정/관리하는 모듈을 통해서 일관되게 관리됨을 가정한다. 센서 네트워크 메타데이터 정의 및 기타 접근 인터페이스는 본 표준 문서의 범위를 벗어난다(TTAS.KO-06.0167, TTAS.KO-06.0168).

센서네트워크는 특성상 지원하는 능력이 천차만별이다. 즉 제공되는 기능, 동작하는 방식등이 센서네트워크를 구성하는 노드들의 스펙 및 응용에 따라서 차이가 많이 난다. 따라서 본 표준에서 정의된 메시지는 해당 센서네트워크가 지원하는 능력에 따라서 선별적으로 구현하며, 단 동일한 기능을 구현할 경우에는 반드시 본 표준의 메시지 구조를 따르도록 한다. 단, 센서네트워크 연결/인증/확인(ConnReqCtrl, ConnResCtrl, DisConnReqCtrl/AuthReqCtrl, AuthResCtrl/ ChannelCheckCtrl,

ChannelConfirmCtrl, NakChk)에 관련된 메시지와 현재 연결되어 있는 센서네트워크의 정보 수집(NetworkInfoReq, NetworkInfoRes)에 관련된 명령어는 반드시 구현하도록 한다.

#### 4.1 센서 네트워크 공통 인터페이스 기능 요구 사항

센서 네트워크 추상화를 제공하기 위한 센서 네트워크 공통 인터페이스는 정보를 제공하는 센서 네트워크와 이를 이용하고자 하는 호스트 단에 구현되어 추상화 기능을 제공하게 된다. 센서 네트워크 추상화를 위해서는 다음과 같은 기능이 요구된다.

표준화된 센서 네트워크 연결 및 해지 기능  
표준화된 센싱 데이터 질의 및 수집 정보 보고 기능  
센서 네트워크 메타 데이터 질의/보고 기능  
센서 네트워크 모니터링 기능  
센서 네트워크 제어 기능  
센서네트워크 인증 기능 및 메시지 암호화기능

추가적으로 센서 네트워크 인터페이스는 사용 환경적 특성에 따라서 구현이 용이하면서도 확장성있게 설계되어야 한다. 센서 네트워크 단에서 센서 네트워크 인터페이스 기능을 제공하는 노드는 계산 능력, 전원 문제에 있어서 복잡한 처리가 불가능할 수도 있으므로, 유연한 구조로 설계되어야한다.

#### 4.2. 센서 네트워크 통신 프로토콜

센서 네트워크를 이용하여 정보를 획득하고자 하는 호스트는 본 표준에서 제안하는 공통 인터페이스를 이용하여 센서 네트워크와 통신한다. 센서 네트워크는 본 표준에서 제안하는 인터페이스를 처리하는 모듈을 구현한다. 해당 모듈은 논리적인 객체로서 센서 네트워크 내부 혹은 외부 어디에나 존재할 수 있다. 본 표준에서는 이와 같은 기능을 제공하는 노드를 센서 네트워크 어댑터(Sensor Network Adaptor)라고 정의한다. 센서 네트워크 어댑터는 센서 네트워크 내부의 통신 프로토콜, 토폴로지, 센서 노드의 개수, 센서 노드에 장착된 센서 유형, 구동기 유형 등의 정보를 모두 파악하여 호스트로부터의 요청을 센서 네트워크 내부적으로 이해할 수 있는 프로토콜로 변환하여 전달하는 기능을 제공한다. 센서네트워크 어댑터와 센서네트워크간의 역할 분담 및 통신 방법은 본 표준의 범위를 넘어선다.

다음 그림 4-1과 그림 4-2는 호스트가 응용 프로그램인 경우와, 응용 프로그램과 센서 네트워크간 중간 처리를 제공하는 미들웨어인 경우로 구분하여 표현한 서비스 프레임워크이다.

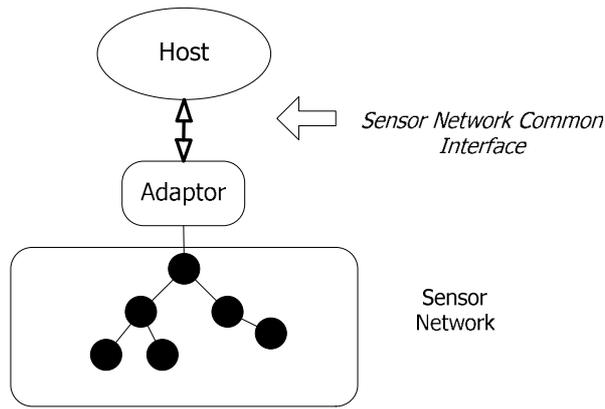


그림 4- 1 호스트가 응용 프로그램인 경우의 통신 프레임워크

센서네트워크는 어댑터를 통해서 호스트인 응용 프로그램과 연결되어 서비스를 제공한다. 이 경우 호스트와 어댑터는 센서네트워크 공통 인터페이스를 이용하여 통신한다.

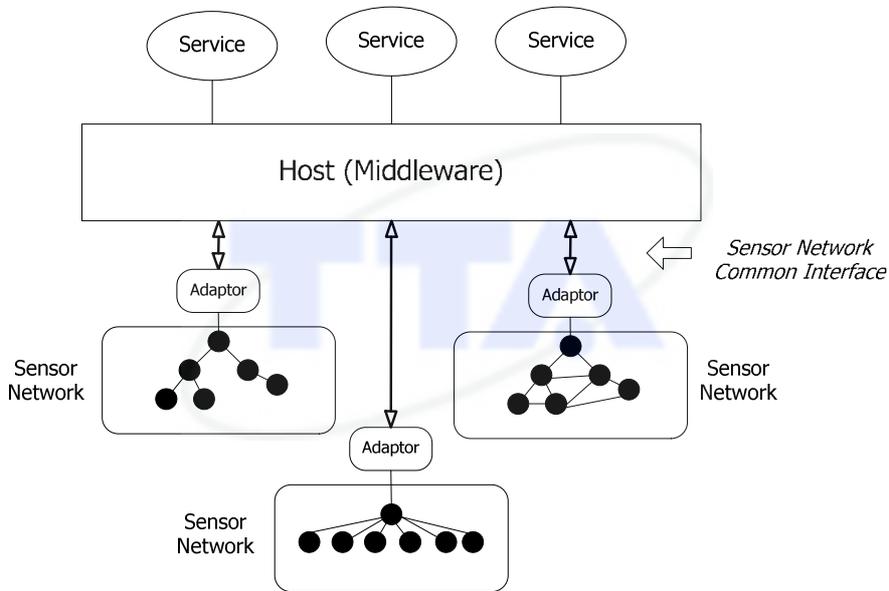


그림 4- 2 미들웨어가 호스트인 경우의 통신 프레임워크

다양한 응용서비스는 미들웨어를 통해서 다양한 센서네트워크와 통신하며, 호스트인 미들웨어와 센서네트워크간 통신은 본 규격에서 정의하는 센서네트워크 공통 인터페이스를 이용한다. 미들웨어와 응용서비스간의 인터페이스는 본 표준의 범위를 벗어난다.

#### 4.2.1 호스트와 센서 네트워크간 연결 설정

호스트와 센서네트워크간의 연결은 크게 두 가지 모드로 구분된다. 기본 모드는 준비된 센서네트워크가 호스트에 연결을 요청하는 SN-initiated 모드이며, 다른 하나는 호스트가 필요시 센서네트워크로 연결을 요청하는 host-initiated모드이다.

#### 4.2.1.1 SN-initiated 모드

호스트와 센서 네트워크 간 연결은 다음의 두 단계로 이루어진다.

**(1 단계) 센서 네트워크가 호스트에 연결을 요청하는 단계.** 연결 요청을 수신한 호스트는 센서 네트워크에 대한 신뢰성을 확인한 후 연결 정보를 전달한다. 호스트에 대한 연결 요청자는 어댑터일수도 그 외의 다른 노드일 수도 있다. 연결 정보는 둘간에 이용할 통신 프로토콜, 연결 방법과 연결 파라미터를 포함한다. (ex. Tcp/ip통신의 경우, 통신프로토콜=tcp/ip, 연결파라미터는 ip address와 port번호)

**(2 단계) 센서네트워크와 호스트의 연결이 성립되는 단계.** 호스트로부터 수신한 연결 정보를 이용해서 센서 네트워크는 호스트에 각 채널을 연결하고, 호스트는 연결을 허용한다.

#### 4.2.1.2 host-initiated 모드

호스트가 센서네트워크에 연결을 요청한 후, 4.2.1.1의 절차를 수행한다.

### 4.2.2 통신 채널

센서 네트워크(어댑터)와 호스트는 채널을 통해서 통신하며, 채널은 세 가지로 분류된다. 명령채널은 센싱 데이터 처리 및 모니터링을 위한 명령/보고를 전달하는데 사용되며 비동기적으로 메시지들이 교환된다. 명령채널을 이용하여 전달되는 명령어는 일반적으로 센서네트워크 내부적으로 일정시간 처리시간이 소요된다. 다음은 제어채널로 메타 데이터 요청/응답 및 센싱 명령에 대한 제어 요청/응답을 전달하는데 사용되며 동기적으로 메시지를 교환하는 제어채널이 있다. 마지막으로 처음 센서 네트워크가 호스트에 연결 요청할 때 사용하는 연결채널이 존재한다.

이러한 채널의 분류는 센서 네트워크와 호스트간 교환되는 메시지에 대한 처리 방법을 구분하기 위함이며, 논리적인 분류로서 구현에 있어서는 필요에 따라서 구분하지 않을 수도 있다.

물리적 통신 채널은 tcp 메시지를 이용하여 구현될 수도 있고, http 메시지를 이용하여 구현될 수도 있다.

채널의 분류는 통신관련노드, 처리 방법(동기식/비동기식)에 따라서 분류한다.

#### 연결 채널(connection channel)

센서네트워크가 호스트에 연결을 요청하기 위해서 사용하는 채널이다. 해당 채널을 위한 포트 번호는 사전에 고정적으로 정해 놓고 사용한다. 호스트에 연결을 요청하는 노드는 어댑터일 수도 혹은 센서네트워크의 다른 노드에 구현될 수도 있다.

#### 명령 채널(command channel)

센서 네트워크는 연결 채널을 통해서 획득한 연결정보를 이용해서 명령 채널의 포

트번호를 할당 받는다. 센서 데이터 요청/보고, 모니터링 정보 요청/보고를 위한 명령 어들이 명령 채널을 이용한다. 센서 네트워크는 명령을 수신하고 그에 대한 응답을 생성하기 위해서 처리 시간이 필요하다. 명령 채널은 비동기식(asynchronous)으로 처리된다. 센서 네트워크 어댑터와 호스트간 사용되는 채널이다.

**제어 채널(control channel)**

센서 네트워크는 연결 채널을 통해서 획득한 연결정보를 이용해서 제어 채널의 포트번호를 전달 받는다. 센서 네트워크에 대한 제어 요청/응답, 연결해지는 제어 채널을 이용한다. 제어 채널은 동기식(synchronous)으로 처리되어 해당 제어가 제대로 처리되었는지 호스트가 확인한 후 다른 연산을 처리할 수 있도록 한다.

	통신관련노드	처리방법	비고
연결 채널	연결요청자 ↔ 호스트	동기식	연결요청자는 어댑터 혹은 그 외의 노드일 수 있음
명령 채널	호스트 ↔ 어댑터	비동기식	
제어 채널	호스트 ↔ 어댑터	동기식	

표 4- 1 채널 분류

**4.3 센서네트워크 공통 메시지**

센서네트워크 공통인터페이스는 통신 프로토콜에 따라 교환되는 메시지를 정의하며, 해당 메시지는 binary 메시지로도 표현가능하며, xml을 이용하여 표현할 수도 있다.

**4.3.1 메시지 유형별 코드**

메시지를 기능별로 분류하고, 각 메시지에 대한 개별 코드값을 정의한다.

**4.3.1.1 메시지유형(MessageType)**

메시지유형은 메시지의 의미를 표시하는 것으로서 다음과 같은 구조를 갖는다.

Message Group (4 bit)	Message Type (12 bit)
--------------------------	--------------------------

메시지 그룹(Message Group)은 다음과 같이 구분된다.

메시지 그룹		값 (16진수)	메시지 종류
연결제어/확인	연결설정/해지	0	ReqConnCtrl, ConnReqCtrl, ConnResCtrl, DisConnReqCtrl
	인증		AuthReqCtrl, AuthResCtrl
요청	정보	1	NetworkInfoReq, BufferDataReq

	명령	2	CmdActionReq, UpdateCmdReq
	네트워크	3	ControlNetworkReq, ControlNodeReq
응답	정보	4	NetworkInfoRes, BufferDataRes
	명령	5	CmdActionRes, UpdateCmdRes
	네트워크	6	ControlNetworkRes, ControlNodeRes
명령	센싱/구동기	7	InstantCmd, ContinuousCmd, InstantEventCmd, InstantAggCmd, ContinuousAggCmd, RunActuatorCmd
	모니터링	8	MonitoringStartCmd, MonitoringStopCmd
보고	센싱/구동기	A	SensingValueRpt, RunActuatorRpt, FinishRpt
	모니터링	B	MonitoringRpt
	기타	C	ErrorRpt, UpdateRpt
확인	채널확인	D	ChannelCheckCtrl, ChannelConfirmCtrl
	오류확인		NakChk
예약		9	
		E	
		F	

표 4- 2 메시지 그룹

4.3.1.2 메시지 유형별 코드 값

메시지 그룹		채널	메시지 유형	코드 값
연결 제어 (0x0000~0x0FFF)	연결설정 /해지	연결	ReqConnCtrl	0x0001 (1) <sub>10</sub>
		연결	ConnReqCtrl	0x0002 (2) <sub>10</sub>
		연결	ConnResCtrl	0x0003 (3) <sub>10</sub>
		명령, 제어	DisConnReqCtrl	0x000F (15) <sub>10</sub>
	인증	명령, 제어	AuthReqCtrl	0x0100 (256) <sub>10</sub>
		명령, 제어	AuthResCtrl	0x0101 (257) <sub>10</sub>
요청 (0x1000~0x3FFF)	정보	제어	NetworkInfoReq	0x1000 (4096) <sub>10</sub>
		제어	BufferDataReq	0x1001 (4097) <sub>10</sub>
	명령	제어	CmdActionReq	0x2000 (8192) <sub>10</sub>
		제어	UpdateCmdReq	0x2001 (8193) <sub>10</sub>
	네트워크	제어	ControlNetworkReq	0x3000 (12288) <sub>10</sub>
		제어	ControlNodeReq	0x3001 (12289) <sub>10</sub>
응답	정보	제어	NetworkInfoRes	0x4000 (16384) <sub>10</sub>

(0x4000~0x6FFF)	명령	제어	BufferDataRes	0x4001 (16385) <sub>10</sub>
		제어	CmdActionRes	0x5000 (20480) <sub>10</sub>
		제어	UpdateCmdRes	0x5001 (20481) <sub>10</sub>
	네트워크	제어	ControlNetworkRes	0x6000 (24576) <sub>10</sub>
		제어	ControlNodeRes	0x6001 (24577) <sub>10</sub>
명령 (0x7000~0x8FFF)	센싱/구 동기	명령	InstantCmd	0x7000 (28672) <sub>10</sub>
		명령	ContinuousCmd	0x7001 (28673) <sub>10</sub>
		명령	InstantEventCmd	0x7002 (28674) <sub>10</sub>
		명령	예약	0x7003
		명령	InstantAggCmd	0x7004 (28676) <sub>10</sub>
		명령	ContinuousAggCmd	0x7005 (28677) <sub>10</sub>
		명령	RunActuatorCmd	0x7A00 (31232) <sub>10</sub>
	모니터링	명령	MonitoringStartCmd	0x8000 (32768) <sub>10</sub>
		명령	MonitoringStopCmd	0x8001 (32769) <sub>10</sub>
보고 (0xA000~0xCFFF)	센싱/구 동기	명령	SensingValueRpt	0xA000 (40960) <sub>10</sub>
		명령	RunActuatorRpt	0xA001 (40961) <sub>10</sub>
		명령	FinishRpt	0xAFFF (45055) <sub>10</sub>
	모니터링	명령	MonitoringRpt	0xB000 (45056) <sub>10</sub>
	네트워크	명령	ErrorRpt	0xC000 (49152) <sub>10</sub>
		명령	UpdateRpt	0xC001 (49153) <sub>10</sub>
확인 (0xD000~0xDFFF)	채널확인	명령, 제어	ChannelCheckCtrl	0xD000 (3582) <sub>10</sub>
		명령, 제어	ChannelConfirmCtrl	0xD001 (3583) <sub>10</sub>
	오류확인	명령, 제어	NakChk	0xDFFF (3328) <sub>10</sub>

표 4- 3 메시지 유형별 코드값

4.3.1.3 주소 표현 방법

몇몇 메시지는 메시지 내부에 상대방이 연결해야할 주소를 전달해야하는 경우가 있다. 이러한 경우 주소는 다음과 같은 형식으로 표현한다.

□ (통신방법)://서버주소/info?포트명=포트값{&포트명=포트값}\*  
 예) tcpip://123.456.78.90/info?ControlPort=1234&CommandPort=1234

- 어댑터가 소켓 통신을 하며 접속할 서버의 주소는 123.456.78.90 이고, 제어채널과 명령채널의 포트는 1234 번 임을 의미

예) tcpip://123.456.78.90/info?ReqConnPort=1234

- 어댑터가 123.456.78.90 ip 에서 호스트의 연결요청을 소켓을 이용해 대기하고 있으며, 그 포트는 1234 번임을 의미

예) http://123.456.78.90/info?ControlPort=1234&CommandPort=5678

- 어댑터가 HTTP 통신을 하여 접속할 서버의 주소는 123.456.78.90 이고, 제어채널은 1234 번 포트, 명령채널은 5678 번 포트임을 의미

### 4.3.2 메시지의 종류

센서네트워크는 그 성능에 따라 정의된 공통 메시지 규격을 모두 지원하거나 일부만 지원할 수 있다. 지원하지 않는 메시지가 전달되는 경우에 어댑터는 이를 무시하고, 이에 대한 어떠한 처리도 하지 않는 것으로 정의한다.

#### 4.3.2.1 연결제어 메시지

호스트와 어댑터가 메시지를 주고 받기 위해서는 먼저 지정된 통신 방법으로 연결을 맺어야 한다. 이러한 연결을 맺기 위해 필요한 것을 연결제어 메시지라고 한다.

호스트와 어댑터간 연결은 연결채널을 통한 메시지 교환 이후, 생성되는 제어채널과 명령채널에 대한 인증 처리가 연속적으로 이루어진다. 이후, 센싱관련 명령어와 센서네트워크 제어 및 정보 수집 명령들이 교환될 수 있다.

##### 4.3.2.1.1 연결설정 및 해지

메시지 이름	메시지유형 값	설명
ReqConnCtrl	0x0001	호스트가 어댑터에 연결을 요청
ConnReqCtrl	0x0002	연결설정 정보 요청
ConnResCtrl	0x0003	연결설정 정보 요청에 대한 응답
DisConnReqCtrl	0x000F	연결해제 요청

표 4- 4 연결설정 및 해지 메시지

호스트와 어댑터 간의 통신을 위해서는 먼저 연결을 맺어야 한다. 연결을 맺기 위한 방법에는 host-initiated 방식과 SN-initiated 방식이 있다. host-initiated 방식의 경우, 호스트는 연결을 원하는 어댑터(센서네트워크)에게 호스트에 연결하도록 요청한다. 해당 요청을 수신한 어댑터는 연결설정 정보 요청 메시지(ConnReqCtrl)를 이용해서 호스트에게 연결을 요청한다.

SN-initiated 방식의 경우, 어댑터가 호스트에 연결을 요청하는 방식이다. 어떠한 방식이든 일단 연결을 맺은 후에는 제어채널을 통해 명시적으로 연결해제를 요청하여야 한다. 연결설정정보요청자와 어댑터의 경우 물리적으로 동일하게 구현될 수 있으며, 필요에 따라서 분리할 수도 있다. 또한 연결설정정보요청자와 어댑터가 분리된 경우, 연결설정정보요청자가 호스트로부터 획득한 정보를 어댑터에 전달하는 방식은 본 문서의 범위를 벗어난다.

host-initiated 모드: 호스트가 어댑터에게 연결을 하도록 요청하는 방식으로, 호스트는 사전에 연결요청을 보낼 주소를 알고 있어야 한다.

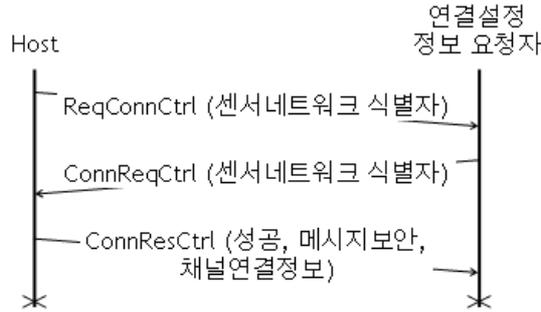


그림 4- 3 호스트의 연결설정 요청 시나리오

위의 시나리오는 호스트가 어댑터 측의 연결설정 정보 요청자에 연결을 맺을 것을 요구하는 경우에 대한 시나리오이다. 호스트는 연결을 맺기 원하는 센서네트워크 측의 연결설정 정보 요청자에 연결설정 요청 메시지(ReqConnCtrl)를 전달하고 이를 수신한 연결설정 정보 요청자는 연결설정 정보 요청 메시지(ConnReqCtrl)를 전달하여 어댑터 각 채널의 연결설정 정보를 수신받게 된다.

SN-initiated 모드: 연결설정정보요청자는 호스트에게 연결설정을 요청하고 이를 수신한 후 제어채널, 명령채널에 대한 정보를 호스트로부터 획득한다. 어댑터는 연결설정정보요청자로부터 제어채널, 명령채널에 대한 정보를 획득한후 해당 정보를 이용하여 호스트에 인증을 요청한다.

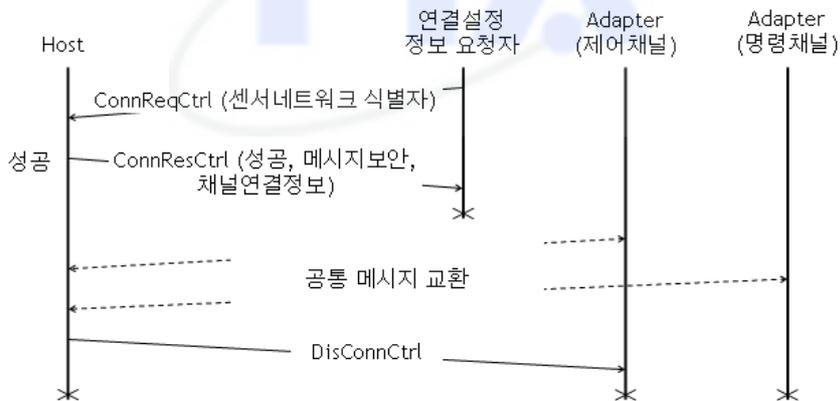


그림 4- 4 어댑터의 연결설정 정보 요청 성공 시나리오

위의 시나리오는 연결설정 정보 요청자가 연결설정 정보 요청 메시지(ConnReqCtrl)를 전달한 이후의 연결설정 및 종료 과정을 표현한다. 어댑터 측의 연결설정 정보 요청 메시지(ConnReqCtrl)를 수신받은 호스트는 해당 연결설정 요청이 유효한지 확인하고, 유효하다면 어댑터 측의 각 채널을 위한 연결설정 정보를 전달하게 된다. 호스트와 어댑터의 제어채널과 명령채널은 미리 정의된 메시지를 주고 받으며 정보를 전달하다가 연결종료 메시지(DisConnCtrl)를 전달하는 것으로 연결을 종료한다.

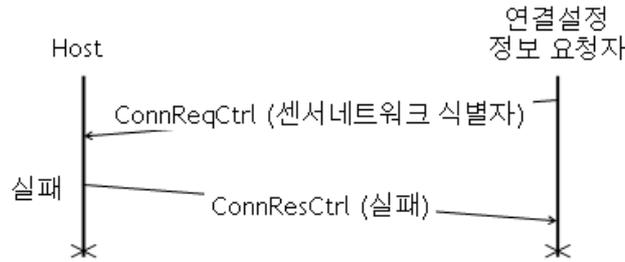


그림 4- 5 어댑터의 연결설정 정보 요청 실패 시나리오

위의 시나리오는 어댑터의 연결설정 정보 요청 메시지(ConnReqCtrl)가 유효하지 않은 경우에 대한 시나리오이다. 어댑터의 연결설정 정보 요청이 유효하지 않다면, 호스트는 실패 정보에 대한 코드를 연결설정 정보 응답 메시지(ConnResCtrl)를 전송하고 별도의 메시지 없이 연결을 종료한다.

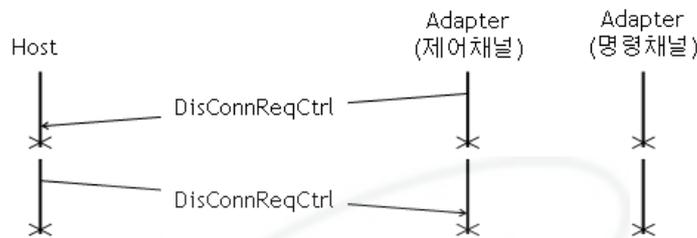


그림 4- 6 연결해제 시나리오

연결해제 요청 메시지(DisConnCtrl)는 제어채널로 전달되며, 제어 채널이 연결해제 요청 메시지를 받으면 명령채널도 함께 종료된다. 연결해제 요청 메시지(DisConnCtrl)는 어댑터가 먼저 전달하거나 혹은 호스트가 먼저 전달할 수 있다. DisConnReqCtrl를 수신한 경우 어댑터는 수행중이던 모든 연산과 저장값들을 초기화하여 다시 연결되었을 경우, 이전 수행값을 호스트로 올리지 않도록 한다.

4.3.2.1.2 채널인증

메시지 이름	메시지유형 값	설명
AuthReqCtrl	0x0100	인증 요청
AuthResCtrl	0x0101	인증 결과

표 4- 5 채널인증 메시지

연결설정 정보 메시지(ConnResCtrl)에는 어댑터의 각 채널에 대한 연결설정 정보와 기밀성 적용을 어떻게할것인가에 대한 정보가 포함되어 있다. 어댑터의 각 채널은 연결 설정 정보에 따라 호스트와 연결을 맺고 제일 먼저 채널에 대한 인증 요청 메시지(AuthReqCtrl)를 전송하여야 한다. 만약, 이외의 메시지가 전송된다면 연결은 자동 종

료된다.

채널인증에 성공한 경우 호스트는 성공하였음을 인증 결과 메시지(AuthResCtrl)에 담아 전송하고, 실패한 경우에는 실패하였음을 담아 전송하고, 접속을 종료한다.

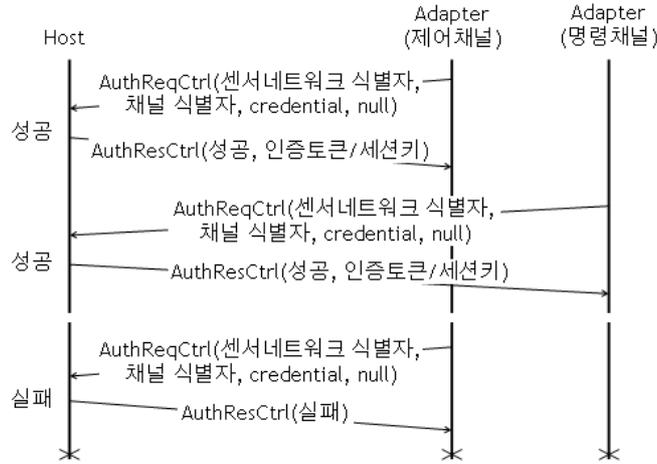


그림 4-7 채널 연결 및 인증 시나리오

위의 시나리오는 분석된 연결설정 정보에 따라 어댑터의 각 채널이 연결되고, 채널에 대한 인증을 수행하는 과정에 대한 시나리오이다. 호스트에 연결된 각 채널은 인증 요청 메시지(AuthReqCtrl)를 전달하여 인증을 요청하고, 성공적으로 확인된 경우 메시지의 암호/복호화가 필요한 경우에는 향후 메시지 암호/복호화에 필요한 정보를 담아 인증 결과 메시지(AuthResCtrl)를 전달한다. 인증에 실패한 경우에는 실패원인을 전달하고, 별도의 메시지 없이 연결을 해지한다. 연결 해지는 인증에 실패한 채널뿐만 아니라 어댑터의 모든 채널에 해당된다.

만약 각 채널이 물리적으로 동일한 소켓을 사용하여 통신하는 경우에는 채널에 대한 인증과정을 한번만 거치도록 한다.

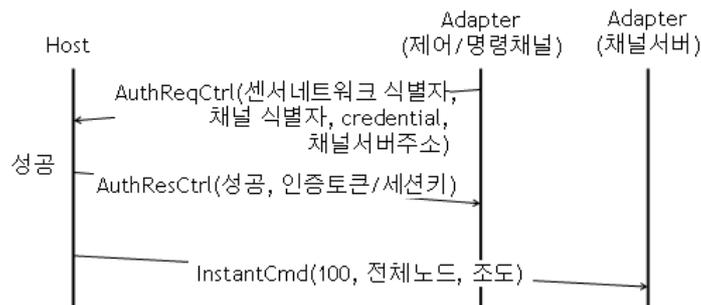


그림 4-8 비 연결성 통신 프로토콜에서의 채널 연결 및 인증 시나리오

위의 시나리오는 비 연결성 통신 프로토콜에서의 채널 연결 및 인증 시나리오이다. 비 연결 통신 프로토콜 중 하나인 HTTP는 인증 요청 메시지(AuthReqCtrl)에 향후 호스트가 명령이나 요청을 전달할 수 있는 주소를 담아 보냄으로써 인증 성공 메시지

(AuthResCtrl)을 받은 이후에 수행되는 명령이나 요청에 대한 채널 인증을 보장하게 된다.

#### 4.3.2.1.3 연결 제어 전체 시나리오

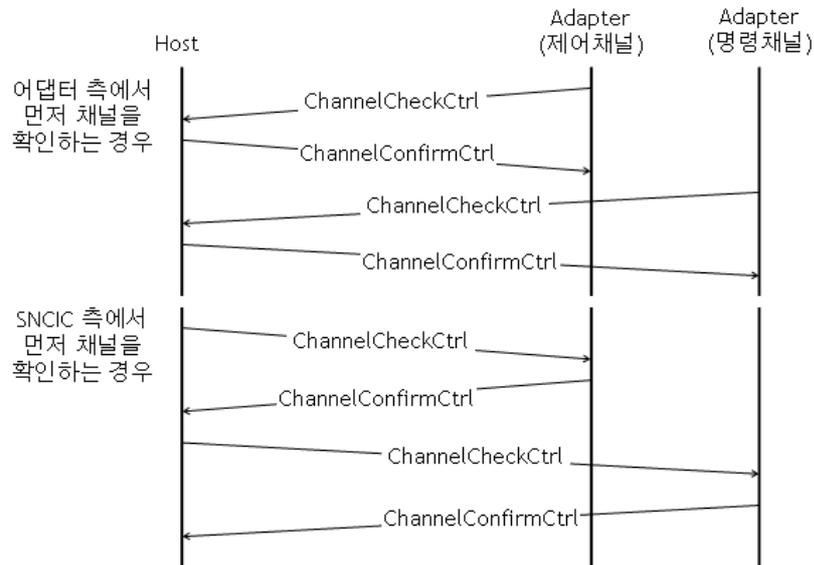


그림 4-9 연결 제어 전체 시나리오

위의 시나리오는 어댑터 측의 연결설정 정보 요청자가 먼저 연결 설정 정보를 호스트 측으로 요청하여 채널에 대한 연결설정 정보를 수신 받아 채널에 대한 연결을 맺고 인증 받아 연결을 종료하기까지 일련의 과정을 표현하고 있다.

#### 4.3.2.2 요청/응답 메시지

호스트가 어댑터에 요청 메시지를 전송하면 어댑터는 이를 처리하여 응답 메시지를 전송한다. 이러한 과정은 동기적으로 이루어진다. 즉, 호스트는 하나의 요청 메시지를 전달하고 응답이 도달하기 전에 다른 요청을 어댑터로 전송할 수 없다.

##### 4.3.2.2.1 네트워크 정보 요청, 응답

메시지 이름	메시지유형 값	설명
NetworkInfoReq	0x1000	네트워크 정보 요청
NetworkInfoRes	0x4000	네트워크 정보 요청에 대한 응답

표 4-6 네트워크 정보 요청, 응답 메시지

응용 프로그램에서 질의가 호스트를 통해 센서네트워크까지 전달되기 위해서는 해당 센서네트워크에 대한 정보가 필요하다. 이러한 정보들을 수집하기 위한 명령이 네트워크 정보 요청 메시지(NetworkInfoReq)이다. 이를 수신한 어댑터는 센서네트워크에 대한 정보를 응답 메시지(NetworkInfoRes)에 담아 응답한다.

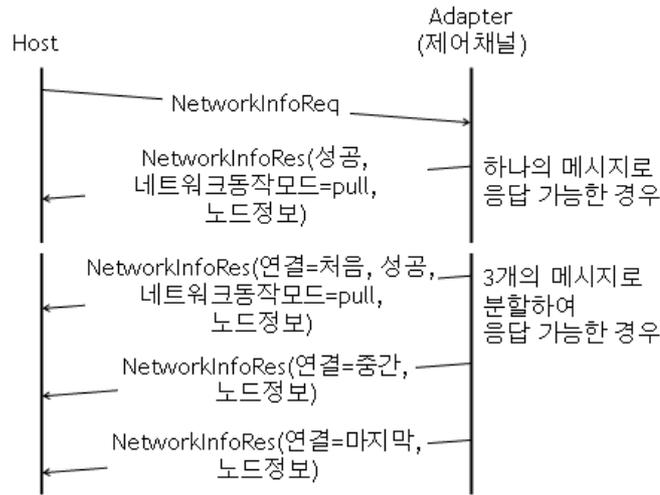


그림 4- 10 네트워크 정보 요청 시나리오

위의 시나리오는 호스트가 네트워크 정보를 요청하고, 어댑터가 이에 대한 응답을 전송하는 경우에 대한 시나리오이다. 어댑터는 센서네트워크에 대한 정보를 하나의 메시지에 모두 담을 수 없을 때 여러 개의 메시지로 분할 할 수 있다. 위의 예에서는 3개의 메시지로 분할되었다.

4.3.2.2.2 버퍼링 메시지 요청, 응답

메시지 이름	메시지유형 값	설명
BufferDataReq	0x1001	버퍼링된 메시지 요청
BufferDataRes	0x4001	버퍼링된 메시지 전달 완료 응답

표 4- 7 버퍼링 정보 요청, 응답 메시지

어댑터는 필요에 따라 보고될 내용들을 버퍼링하고 있다가 이를 한꺼번에 호스트에 전달할 수 있다. 그러나, 연결에 대한 해제 혹은 응용 프로그램의 요청 등으로 인하여 어댑터의 전송 스케줄과는 별도로 버퍼링된 값들이 전송될 수도 있다.

요청에 의한 응답 메시지(xxxRes)는 동기적으로 처리되어야하는 것으로 버퍼링 없이 전달되어야 한다.

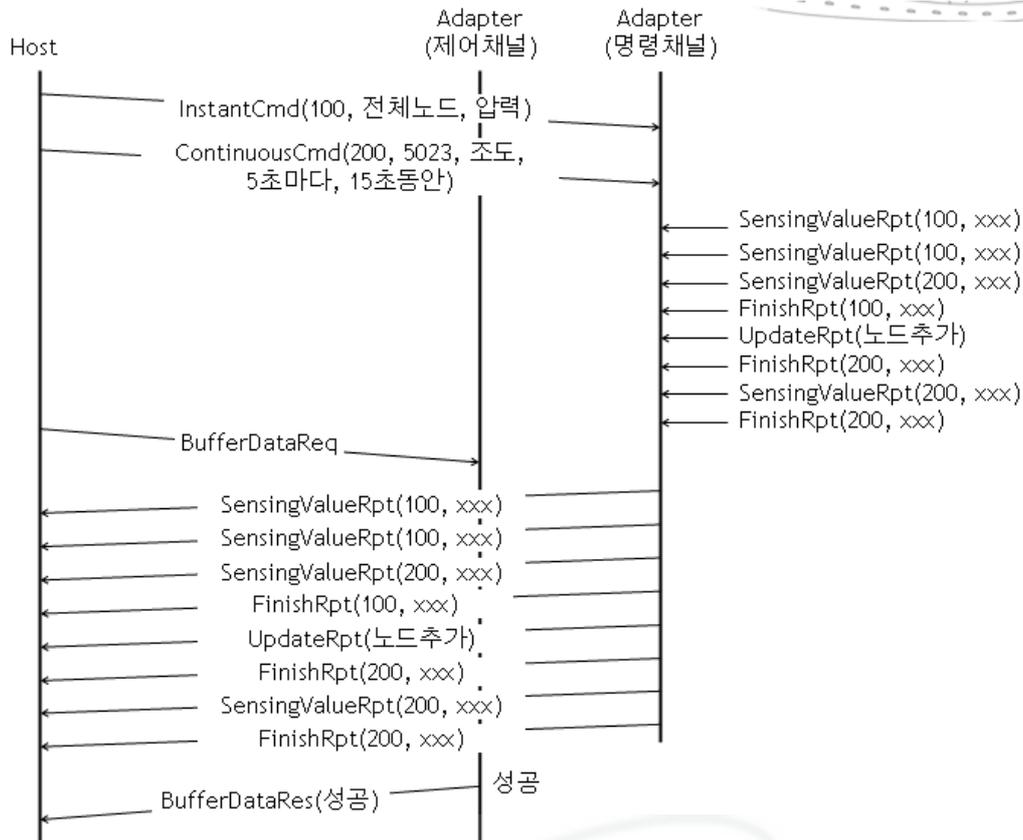


그림 4- 11 버퍼링 정보요청/응답 시나리오

위의 시나리오는 5분간 보고서를 버퍼링하였다가 한꺼번에 전송하는 어댑터가 버퍼링 정보 요청/응답에 대해 동작하는 시나리오이다. 먼저 호스트는 어댑터가 버퍼링하고 있는 보고서들을 수신하기 위해 버퍼링된 메시지 요청메시지(BufferDataReq)를 제어채널을 통해 어댑터로 전송하다. 이를 수신한 어댑터는 명령채널을 통해 버퍼링하고 있는 메시지들을 모두 전송하고, 전송이 완료되었음을 알리기 위해 응답 메시지(BufferDataRes)를 전송한다. 버퍼링된 메시지가 없는 경우에는 단순히 성공하였음을 알리는 응답 메시지(BufferDataRes)만을 전송하면 된다.

어댑터는 지정된 버퍼링시간과는 별도로 네트워크 정보 갱신 메시지(UpdateRpt)나 에러 보고(ErrorRpt), 메시지 확인(NakChk) 메시지 등과 같이 호스트로 즉시 알려야 할 필요가 있는 내용은 즉시 전송할 수 도 있다.

만약, 어댑터 측에서 센싱값에 대한 버퍼링을 지원하지 않는다면 요청에 대해 무조건 성공을 알리는 응답 메시지를 전송하거나 해당 명령을 지원하지 않음을 전달할 수 있다.

4.3.2.2.3 명령 제어 요청, 응답

메시지 이름	메시지유형 값	설명
CmdActionReq	0x2000	실행중인 센싱명령 제어
CmdActionRes	0x5000	실행중인 센싱명령 제어 결과 응답

표 4- 8 명령 제어 요청, 응답 메시지

센서네트워크에 전달된 명령은 명령 제어 요청에 의해 일시중지, 재시작, 중단될 수 있다. 이러한 명령의 제어 요청은 명령제어 요청 메시지(CmdActionReq)에 의해 전달되며, 어댑터는 해당 요청을 수행한 후에 그 결과를 담은 응답메시지(CmdActionRes)를 전송한다.

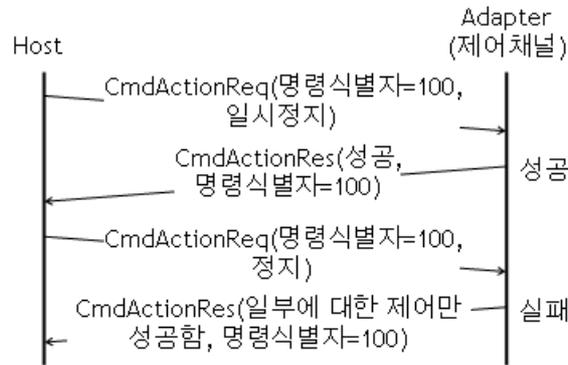


그림 4- 12 명령제어 요청/응답 시나리오

위의 시나리오는 사전에 어댑터에 100번 명령이 전송된 경우를 가정하고 작성되었다. 호스트는 100번 명령의 실행을 잠시 중단하기 위해 센서명령 제어 메시지(CmdActionReq)를 전달한다. 이를 수신한 어댑터는 해당 명령의 수행을 일시 정지하고, 제어에 성공하였음을 센서명령 제어 결과 메시지(CmdActionRes)에 담아 전송하였다. 이후, 100번 명령에 대한 정지를 요청하였고 이는 해당 명령을 수행중인 일부 노드에서만 처리가 완료되어 실패하여, 실패를 응답하였다.

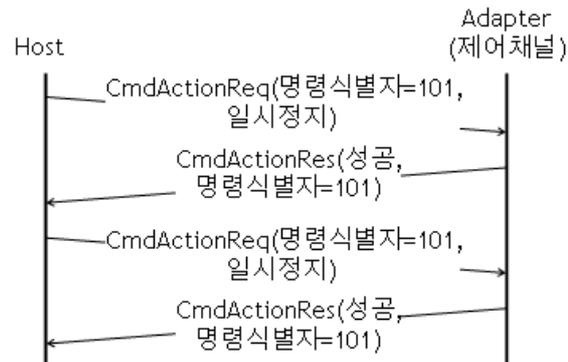


그림 4- 13 명령에 대한 다수 번의 일시정지 제어 처리 시나리오

위의 시나리오에서는 먼저 101번 명령에 대한 일시 정지를 요청하고, 어댑터 측에서는 해당 제어 요청을 성공적으로 수행하였다. 그런데, 잠시 후에 이미 일시 정지된 명령에 대해 다시 일시 정지 요청이 전달된 경우에는 해당 요청을 성공하였음을 알린다.

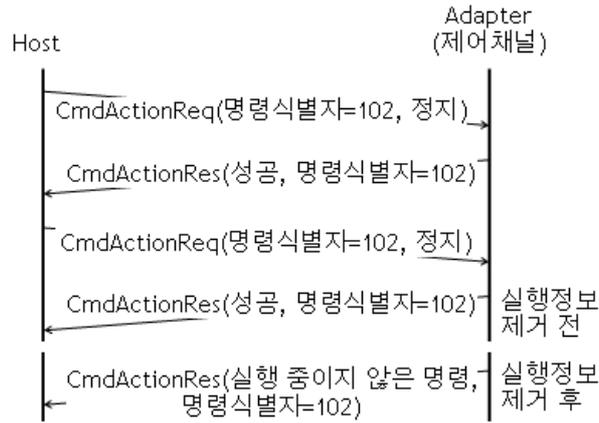


그림 4-14 명령에 대한 다수번의 정지 제어 처리 시나리오

위의 시나리오에서는 먼저 102번 명령에 대한 정지를 요청하고, 어댑터 측에서는 해당 제어 요청을 성공적으로 수행하였다. 그런데, 잠시 후에 이미 정지된 명령에 대해 다시 정지 요청이 전달되었다. 이러한 경우 어댑터 측에서 해당 명령이 현재 정지된 상태라는 사실을 알 수 있다면 제어 요청에 대해 성공을 반환한다. 명령에 대한 정지는 해당 명령의 수행 완료를 의미하므로, 해당 명령에 대한 정보를 모두 삭제한 이후라서 자신이 102번 명령을 수행했었거나 이를 중지시켰다는 정보를 알 수 없다면 실행 중지하지 않은 명령임을 응답한다.

4.3.2.2.4 명령 갱신 요청, 응답

메시지 이름	메시지유형 값	설명
UpdateCmdReq	0x2001	실행중인 센싱명령 갱신
UpdateCmdRes	0x5001	실행중인 센싱명령 갱신 결과 응답

표 4-9 명령 갱신 요청, 응답 메시지

센서네트워크로 전달되어 실행중인 명령은 응용의 요청이나 실행 목적에 따라 실행을 위한 파라미터 값들이 변경될 수 있다. 변경 가능한 항목은 명령의 조건, 시간, 함수이다. 이러한 명령의 실행 파라미터 변경 요청은 명령 갱신 요청 메시지(UpdateCmdReq)에 의해 전달되며, 이에 대한 처리 후 명령갱신 요청 결과를 담은 응답 메시지(UpdateCmdRes)로 응답한다.

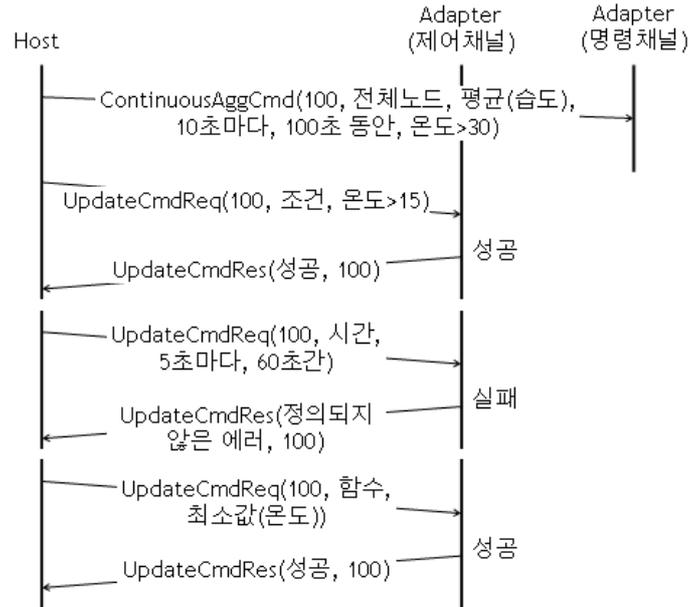


그림 4- 15 명령갱신 요청/응답 시나리오

위의 시나리오에서 어댑터에 전달된 100번 명령은 센서네트워크에 속한 전체 노드에서 100초 동안 매 10초마다 온도 값이 30보다 클 때 습도 값을 센싱하여 그 평균을 보고하라는 명령이다. 이를 수신한 어댑터는 센서네트워크로 명령을 전달하여 수행될 수 있도록 한다.

이 후에 100번 명령에 대한 조건을 “온도가 30보다 클 때”에서 “온도가 15보다 클 때”로 변경하라는 요청이 전달되었다. 어댑터는 이를 수신하여 성공적으로 파라미터를 변경하고, 이를 호스트에 응답한다.

두 번째로 센싱시간 정보 갱신을 위한 요청이 전달되었는데, 해당 파라미터에 대한 갱신은 실패하였다. 이러한 경우 어댑터는 실패원인을 담은 실패코드가 반환하여야 하며, 명령은 처음 전달된 상태로 계속 진행된다.

세 번째로 집계 정보를 “평균(습도)”에서 “최소값(온도)”로 변경하라는 요청이 전달되었다. 이 요청은 성공적으로 처리되고, 어댑터는 제어채널을 통해 성공하였음을 응답한다.

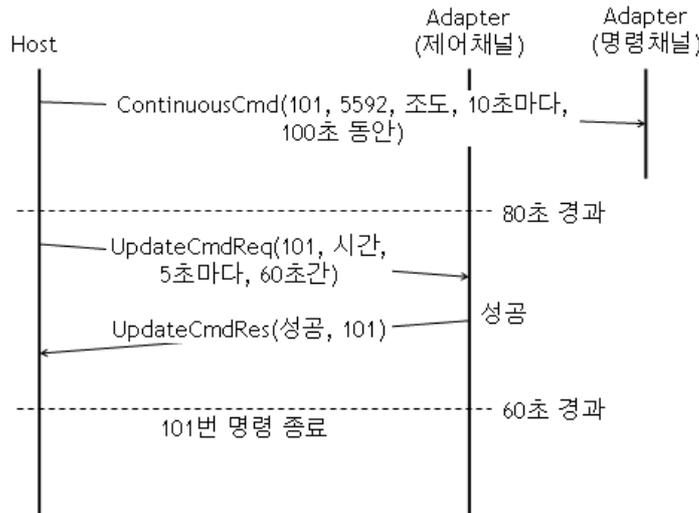


그림 4- 16 시간정보 갱신 요청/응답 시나리오

위의 시나리오에서 호스트는 어댑터에 10초마다 100동안 센싱하라는 101번 명령을 전달하였다. 이후 매 10초마다 값이 센싱되고, 이 값을 보고하는 과정이 80초간 진행되었다. 이 시점에서 해당 명령에 대한 시간 정보가 5초마다 60초간 센싱하는 것으로 변경되는 경우 101번 명령의 종료시점은 명령갱신 시간에서 갱신되는 시간의 명령 완료시간만큼 이른 시간이다.

#### 4.3.2.2.5 센서네트워크 제어 요청, 응답

메시지 이름	메시지유형 값	설명
ControlNetworkReq	0x3000	네트워크에 대한 제어 요청
ControlNetworkRes	0x6000	네트워크에 대한 제어 결과 응답

표 4- 10 센서네트워크 제어 요청, 응답 메시지

동작 중인 센서네트워크는 요청에 따라 제어될 수 있다. 제어의 유형에는 네트워크 상태 초기화를 위한 리셋, 통신채널 변경, 동작모드 변경이 있다. 센서네트워크를 제어하기 위해서 호스트는 센서네트워크 제어 요청 메시지(ControlNetworkReq)에 제어 유형과 필요한 경우 제어에 필요한 값을 담아 전달한다. 이를 수신한 어댑터는 해당 메시지를 처리하고, 그 결과를 센서네트워크 제어 응답 메시지(ControlNetworkRes)에 담아 응답한다.

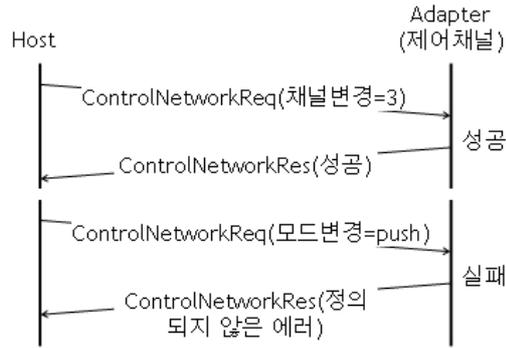


그림 4- 17 센서네트워크 제어 요청/응답 시나리오

위의 시나리오는 센서네트워크의 채널과 모드를 변경 하는 경우를 표현하고 있다. 먼저 호스트는 센서네트워크의 통신채널 변경을 요청한다. 이를 수신한 어댑터는 센서네트워크의 통신 채널을 변경하고, 그 결과를 응답한다. 두번째로 호스트는 센서네트워크의 동작 모드 변경을 요청하였다. 모드 변경에 실패한 어댑터는 실패코드를 담아 응답한다.

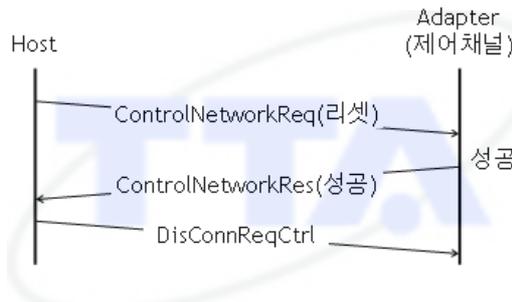


그림 4- 18 센서네트워크 리셋 요청/응답 시나리오

위의 시나리오는 센서네트워크를 리셋하는 경우를 표현하고 있다. 먼저 호스트는 센서네트워크의 리셋을 요청하고, 어댑터는 해당 요청을 수행할 수 있는 경우 수행할 수 있음을 의미하는 성공을 반환한다. 이후 센서네트워크 리셋을 감지한 호스트는 연결설정 종료 메시지(DisConnReqCtrl)를 전달한다.

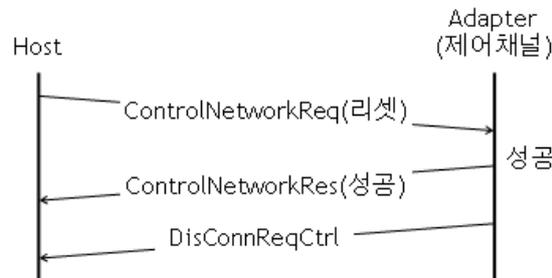


그림 4- 19 센서네트워크 리셋 요청/응답 시나리오

위의 시나리오에서는 센서네트워크 리셋 성공을 응답한 어댑터가 먼저 연결설정 종

료 메시지(DisConnReqCtrl)를 호스트에 전달하였다.

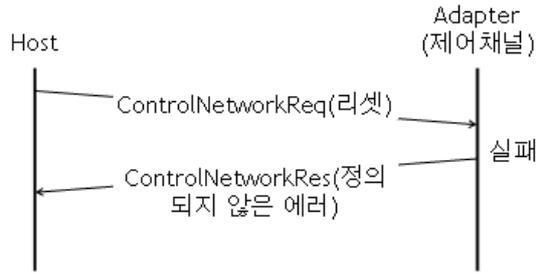


그림 4- 20 센서네트워크 리셋 요청/응답 실패 시나리오

위의 시나리오는 호스트의 센서네트워크 리셋 요청을 어댑터가 수행할 수 없어서 실패 코드를 반환한 경우이다. 리셋을 향 후 수행한다는 성공 응답에 반해, 이 경우에는 연결설정이 종료되지 않는다.

4.3.2.2.6 노드 제어 요청, 응답

메시지 이름	메시지유형 값	설명
ControlNodeReq	0x3001	노드에 대한 제어 요청
ControlNodeRes	0x6001	노드에 대한 제어 결과 응답

표 4- 11 노드 제어 요청, 응답 메시지

동작중인 노드는 요청에 따라 제어될 수 있다. 제어의 유형에는 리셋, 켜기/끄기, 부모노드 설정이 있다. 노드에 대한 제어가 필요한 경우 호스트는 노드제어 요청 메시지(ControlNodeReq)에 제어 대상이 되는 노드 식별자와 제어유형, 제어에 필요한 값을 담아 전달한다. 어댑터는 제어 요청을 처리하고, 그 결과를 노드제어 응답 메시지(ControlNodeRes)에 담아 응답한다.

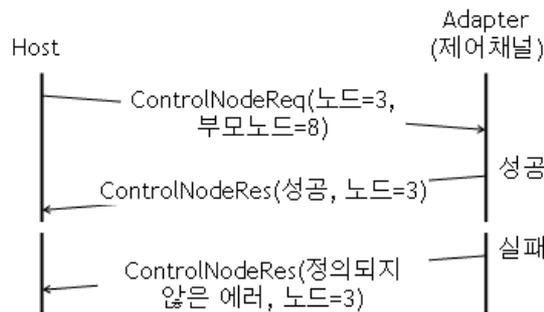


그림 4- 21 노드 제어 요청/응답 시나리오

위의 시나리오는 3번 노드에 대한 부모노드를 8번 노드로 설정하도록 하는 노드 제어 요청이다. 이를 수신한 어댑터는 해당 노드에 대한 부모노드 변경을 수행하고, 그 결과를 반환한다.

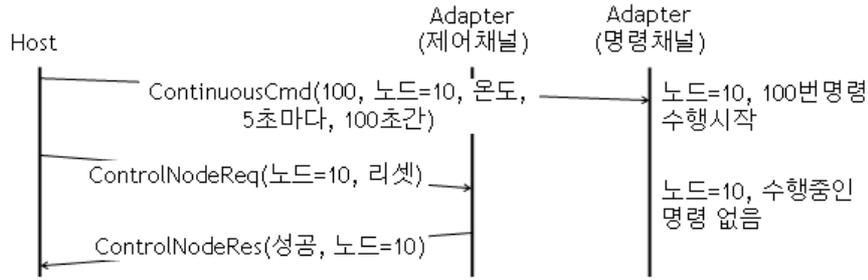


그림 4- 22 명령 수행과 노드 제어 시나리오

위의 시나리오에서는 노드 10번에 5초마다 100초간 온도를 센싱하는 100번 명령이 전달되고, 노드가 이 명령을 수행한다. 이후 10번 노드를 리셋시키라는 명령이 전달되었고, 이 요청은 성공적으로 수행되었다. 리셋된 노드 10번은 수행 중이던 명령을 잃어버리고, 초기 상태가 된다.

4.3.2.2.7 요청에 대한 응답 코드

동기적으로 동작되는 요청에 대한 모든 응답은 요청의 처리 결과가 성공적으로 이루어졌는지 여부를 표현하는 코드 값을 가진다. 결과를 표현하는데 공통적으로 사용되는 응답 코드는 다음과 같다.

응답코드	설 명
0x00 (1) <sub>10</sub>	성공
0x11 (17) <sub>10</sub>	동일한 어댑터가 이미 연결되어 있음 연결설정 정보 요청 응답 메시지(ConnResCtrl), 채널인증 응답 메시지(AuthResCtrl)에서 유효
0x12 (18) <sub>10</sub>	등록되지 않은 센서네트워크 식별자 연결설정 정보 요청 응답 메시지(ConnResCtrl)에서 유효
0x13 (19) <sub>10</sub>	호스트의 연결 자원이 부족함 연결설정 정보 요청 응답 메시지(ConnResCtrl)에서 유효
0x21 (33) <sub>10</sub>	연결설정 요청과정이 이루어지지 않았음 채널 인증 응답 메시지(AuthResCtrl)에서 유효
0x22 (34) <sub>10</sub>	유효하지 않은 채널 식별자 채널 인증 응답 메시지(AuthResCtrl)에서 유효
0x23 (35) <sub>10</sub>	유효하지 않은 credential 채널 인증 응답 메시지(AuthResCtrl)에서 유효
0x31~0x4F	예약
0x41~0x4F	예약
0x51 (81) <sub>10</sub>	실행되고 있지 않은 명령 명령 제어 응답 메시지(CmdActionRes, UpdateCmdRes)에서 유효

0x52 (82) <sub>10</sub>	중단되지 않은 명령 명령 제어 응답 메시지(CmdActionRes)에서 유효
0x53 (83) <sub>10</sub>	일부에 대한 제어만 성공함 명령 제어 응답 메시지(CmdActionRes, ControlNetworkRes)에서 유효
0x61~0x6F	예약
0x71~0x7F	예약
0x81~0x8F	예약
0x91~0xFC	예약
0xFD (253) <sub>10</sub>	처리 대상이 없는 명령. 유효한 명령이고, 명령 내에 포함된 정보의 의미를 모두 알 수 있지만, 해당 명령을 수행할 대상이 존재하지 않음.
0xFE (254) <sub>10</sub>	지원하지 않는 명령. 유효한 명령이지만 명령 내에 포함된 정보의 의미를 모두 알 수 없음
0xFF (255) <sub>10</sub>	정의되지 않은 에러. 명령을 분석하여 처리를 시도하였지만, 정의되지 않은 원인으로 인하여 처리에 실패하였음

표 4- 12 요청에 대한 응답 코드

4.3.2.3 명령 및 보고 메시지

호스트는 어댑터에 센싱 혹은 모니터링 명령을 전송하고, 어댑터는 수행한 결과를 보고한다. 하나의 명령에 대한 결과보고는 존재하지 않거나 다수 개일 수 있다. 수신된 센싱명령이나 모니터링 명령에 대한 에러 상황 혹은 센서네트워크에 대한 에러 상황은 에러 보고 메시지(ErrorRpt)를 통해 전송되며, 네트워크에 대한 노드, 트랜스듀서에 대한 정보 변동 사항은 갱신 메시지(UpdateRpt)를 통해 전송된다.

4.3.2.3.1 센싱 명령/보고 메시지

메시지 이름	메시지유형 값	설명
InstantCmd	0x7000	일회성 명령
ContinuousCmd	0x7001	연속성 명령
InstantEventCmd	0x7002	일회성 이벤트 명령
InstantAggCmd	0x7004	일회성 집계 명령
ContinuousAggCmd	0x7005	연속성 집계 명령
SensingValueRpt	0xA000	센싱값 보고

표 4- 13 센싱 명령, 보고 메시지

센싱명령 수행 방법을 확인하기 위하여 다음 그림과 같이 구성된 센서네트워크를 가정하자. 센서네트워크에는 식별자가 5952, 5023, 3322인 3개의 노드가 존재하고 각 노드에는 조도, 압력, 온도, 습도센서가 장착되어 있다.



그림 4- 23 센싱명령 처리를 위한 센서네트워크 구성 예

위와 같이 구성된 센서네트워크에서 명령에 대한 처리 결과는 다음과 같이 보고되어야 한다. 가장 기본적인 명령은 일회성 명령(InstantCmd)이며, 연속성 명령(ContinuousCmd)은 주어진 시간 동안 매 주기마다 일회성 명령을 수행하는 명령이다. 일회성 이벤트 명령(InstantEventCmd)은 주어진 이벤트가 발생할 때 마다 주어진 일회성 명령을 수행한다. 일회성 집계명령(InstantAggCmd)은 주어진 집계명령을 한번 수행하고, 연속성 집계명령(ContinuousAggCmd)은 주어진 주기마다 집계명령을 수행한다. 명령에 대한 처리 결과는 센싱값 보고서(SensingValueRpt)를 통해 전달되고, 센싱값에 대한 전달이 완료되었음을 완료 보고 메시지(FinishRpt)를 통해 전달한다. 완료 보고 메시지(FinishRpt)는 어댑터의 성능에 따라 지원하거나 지원하지 않을 수 있다. 지원여부는 호스트가 사전에 알 수 있어야 하며, 모든 명령에 대해 일괄적으로 지원하거나 지원하지 않아야 한다.

4.3.2.3.1.1 일회성 명령과 보고

일회성 명령은 명령이 전달되면 한번 값을 센싱하여 조건에 맞는 값을 보고하는 명령이다. 조건에 맞는 값이 존재하지 않는 경우에는 센싱값 보고 또한 이루어지지 않게 된다. 일회성 명령은 그 특성상 명령에 대한 갱신이 이루어질 수 없다.

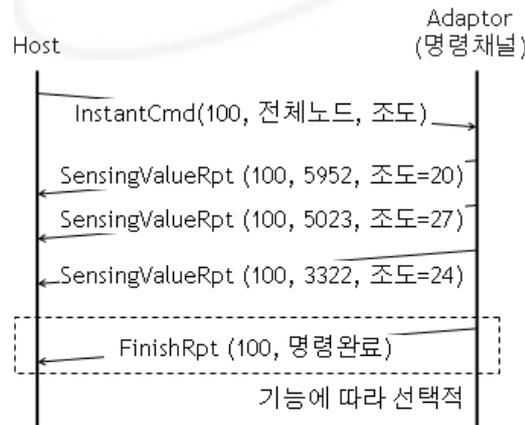


그림 4- 24 일회성 명령의 처리 시나리오 - 전체 노드에 존재하는 센서에서 센싱

위의 시나리오에서는 전체 노드에서 조도를 센싱하는 100번 명령이 완료보고 메시지(Finishrpt)를 처리할 수 있는 어댑터로 전달되었다. 어댑터는 각 노드에서 센싱된 조도 값을 센싱하여 명령식별자, 노드 식별자 등을 첨부하여 센싱값 보고 메시지(SensingValueRpt)에 담아 보고하고, 마지막으로 명령에 대한 처리가 완료되었음을 완료보고 메시지(FinishRpt)를 통해 전달하였다.

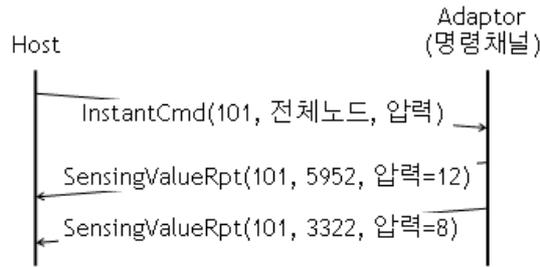


그림 4- 25 일회성 명령의 처리 시나리오 - 일부 노드에는 존재하지 않는 센서에서 센싱

위의 시나리오에서는 전체 노드에서 압력을 센싱하는 101번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전달되었다. 어댑터는 압력센서가 있는 5952, 3322번 노드에서 센싱된 압력 값을 센싱하여 명령식별자, 노드 식별자 등을 첨부하여 보고한다. 압력센서가 존재하지 않는 5023번 노드는 명령을 수행하지 않는다. 명령 수행에 따른 결과는 센싱값 보고 메시지(SensingValueRpt)로 전달되었다.

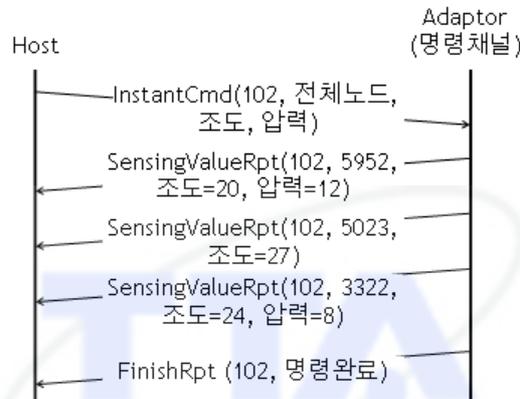


그림 4- 26 일회성 명령의 처리 시나리오 - 다수의 센서에서 센싱

위의 시나리오에서는 전체노드에서 조도와 압력을 센싱하는 102번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 있는 어댑터로 전달되었다. 조도와 압력센서가 모두 장착된 5952번, 3322번 노드는 해당 값을 모두 전달하였고, 5023번 노드는 압력센서가 존재하지 않기 때문에 조도 값만을 보고하였다.

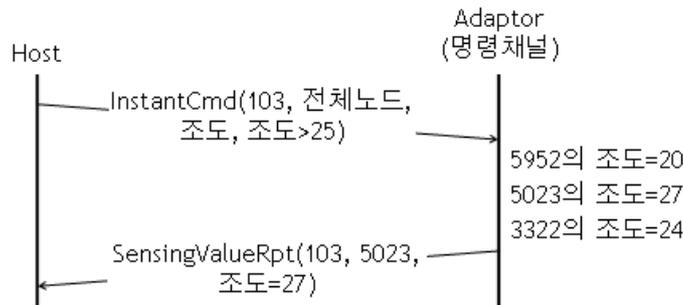


그림 4- 27 일회성 명령의 처리 시나리오 - 센싱 후 조건 판단

위의 시나리오에서는 전체 노드에서 센싱된 조도 값이 25보다 큰 경우 이를 보고하는 103번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전달되

었다. 각 노드에서 센싱된 조도 값이 각각 20, 27, 24일 때, 어댑터는 조건에 맞는 값을 센싱한 5023번 노드의 값에 명령 식별자, 노드 식별자 등의 정보를 첨부하여 보고한다. 5952, 3322번 노드에서 센싱된 값은 조건에 맞지 않으므로 보고되지 않았다.

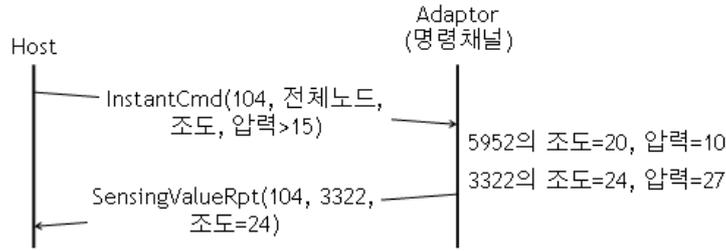


그림 4- 28 일회성 명령의 처리 시나리오 - 보고와 조건판단을 위한 센싱값 유형이 다른 경우

위의 시나리오에서는 전체노드에서 조도와 압력 값을 센싱하여 압력 값이 15보다 크다면 조도 값을 보고하는 104번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전달된 경우에 대한 시나리오이다. 명령은 압력 값을 판단하여야 하므로 압력 센서가 존재하지 않는 5023번 노드는 명령 처리 대상에서 제외된다. 나머지 노드 중에서 주어진 조건을 만족하는 값이 센싱된 3322번 노드의 조도 값만이 보고되었다.

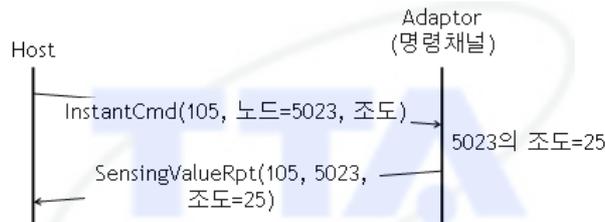


그림 4- 29 일회성 명령의 처리 시나리오 - 노드를 지정하여 센싱

위의 시나리오에서는 5023번 노드에서 조도 값을 센싱하여 보고하는 105번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전달된 경우에 대한 시나리오이다. 어댑터는 5023번 노드에서 센싱된 조도 값을 보고한다.

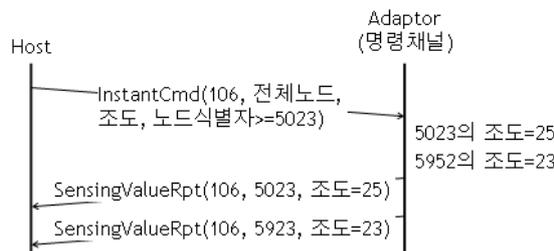


그림 4- 30 일회성 명령의 처리 시나리오 - 조건에 노드 식별자가 존재하는 경우

위의 시나리오에서는 전체 노드에서 노드의 식별자가 5023보다 크거나 같을 때 조도 값을 보고하는 106번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전달되었다. 주어진 조건을 만족하지 않는 3322번 노드는 센싱값 보고 대상에서 제외되고, 5023, 5952번 노드에서 센싱된 값만이 보고된다.

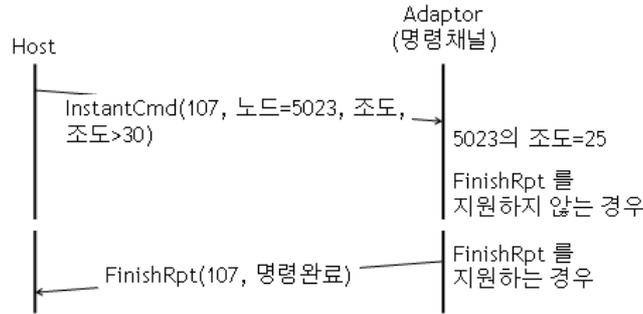


그림 4- 31 일회성 명령의 처리 시나리오 - 완료 보고 메시지 처리

위의 시나리오에서는 5023번 노드에서 조도의 값을 센싱하여 이 값이 30보다 큰 경우 조도 값을 보고하는 107번 명령이 어댑터로 전달되었다. 명령 전달 후에 센싱된 조도 값은 25이고, 이는 조건을 만족하지 않아 보고될 센싱값이 존재하지 않게 된다. 이러한 경우 완료 보고 메시지(FinishRpt)를 지원하지 않는 경우에는 그대로 명령이 완료되며, 완료 보고 메시지(FinishRpt)를 지원하는 경우에는 해당 명령에 대한 처리가 완료되었음을 알리는 메시지가 전달되고 명령이 완료된다.

4.3.2.3.1.2 연속성 명령과 보고

연속성 명령은 명령이 수신된 순간부터 주어진 주기마다 값을 센싱하여 보고하는데 이는 해당 명령의 유효시간까지 계속된다. 센싱에 대한 규칙은 일회성 명령과 동일하므로, 이 절에서는 시간에 대한 부분만을 기술한다.

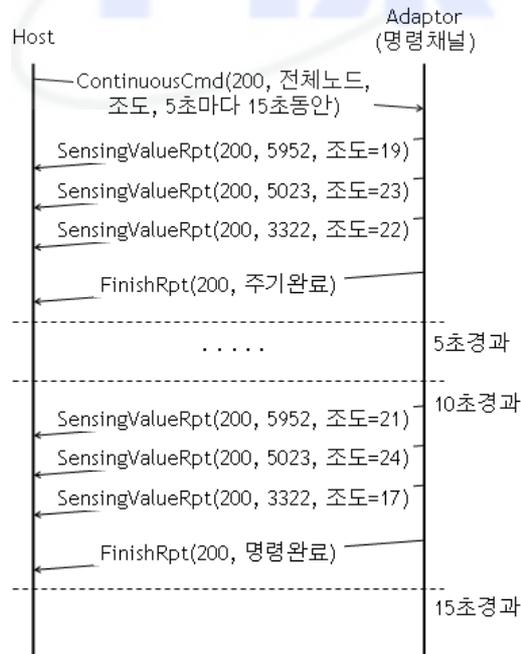


그림 4- 32 연속성 명령의 처리 시나리오 - 주기와 완료시간 처리

위의 시나리오에서는 전체노드에서 5초마다 15초동안 조도를 센싱하여 보고하는 200번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 있는 어댑터로 전달되었다.

일회성 명령과 마찬가지로 명령이 수신되면 어댑터는 각 노드의 조도 센싱값을 보고하고, 5초가 경과한 다음 다시 전체노드에서 조도 값을 센싱하여 보고한다. 이러한 과정을 15초 동안 반복한다. 완료 보고 메시지(FinishRpt)는 주기가 완료된 경우에는 주기완료 임을 가장 마지막 주기에서는 명령완료 임을 알려주어야 한다.

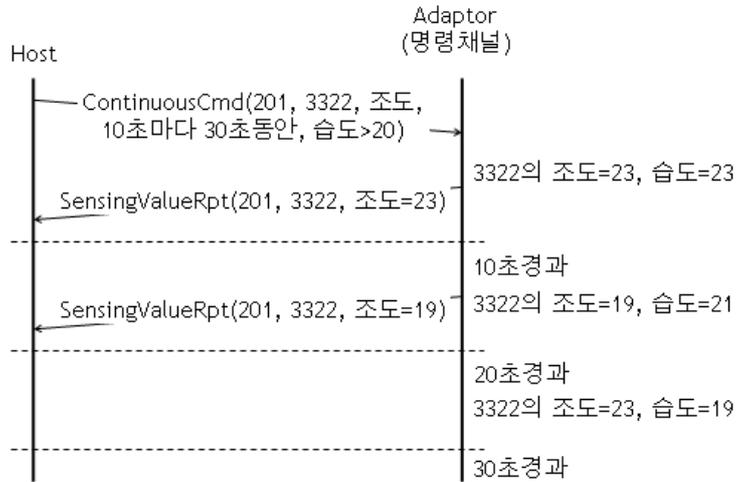


그림 4- 33 연속성 명령의 처리 시나리오 - 주기별 조건처리

위의 시나리오에서는 전체노드에서 습도가 20보다 큰 경우 조도 값을 보고하는 것을 매 10초마다 30초동안 수행하는 201번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전송되었다. 첫 번째와 두번째 주기에서 센싱된 습도 값은 주어진 조건을 만족하므로 조도 값이 보고되었으며, 마지막 주기에서는 센싱된 습도 값이 조건을 만족하지 못하므로 센싱값에 대한 보고가 존재하지 않는다.

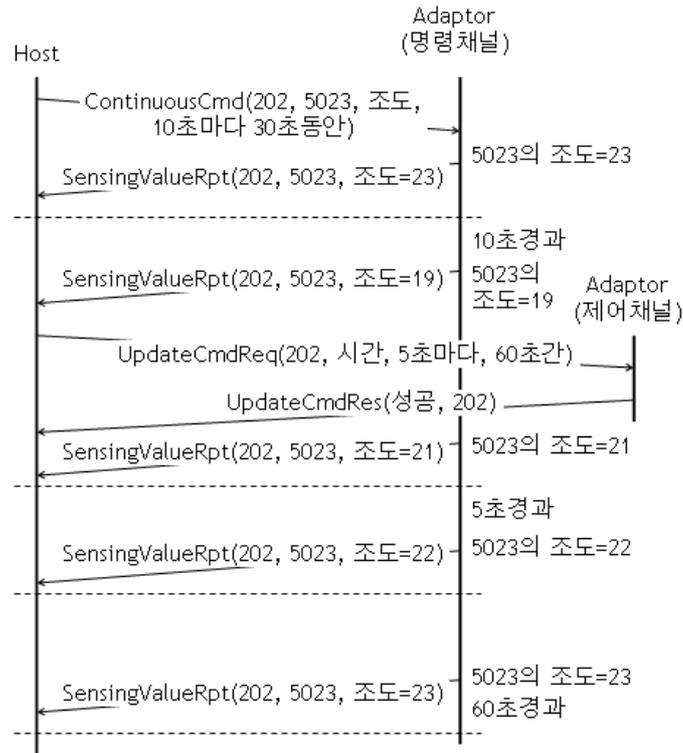


그림 오류! 지정한 스타일은 사용되지 않습니다. 4-1 연속성 명령과 시간 정보 갱신 처리 시나리오  
- 시간 정보 갱신

위의 시나리오에서는 5023번 노드에서 10초마다 30초동안 조도 값을 센싱하여 보고하는 202번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전송되고 이에 대한 시간 정보를 갱신하는 경우에 대한 시나리오이다. 두번째 주기에서 5023번 노드의 조도 값은 19이고, 이를 보고한 다음 주기와 명령에 대한 유효시간이 “5초마다 60초간”으로 성공적으로 갱신되었다. 갱신된 시점을 기준으로 주어진 202번 명령의 주기는 5초마다로, 명령에 대한 완료시간은 갱신 시점으로부터 60초 간으로 계산된다.

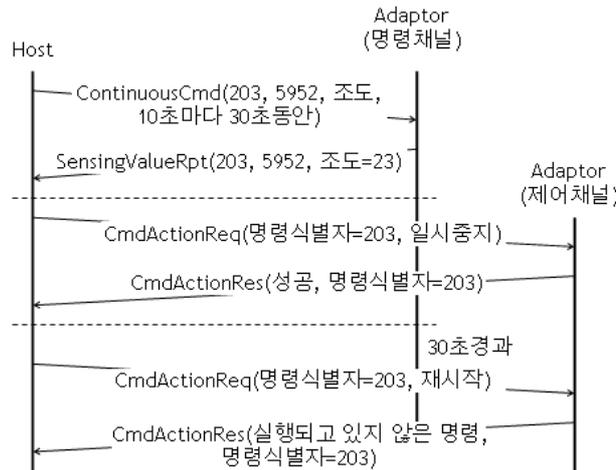


그림 4- 34 연속성 명령과 명령 제어 처리 시나리오 - 완료 시간 이후의 제어요청

위의 시나리오에서는 5952 노드에서 10초마다 30초동안 조도 값을 센싱하여 보고하는 203번 명령이 어댑터에 전송되고 이에 대한 제어 명령이 전달되는 경우에 대한 시나리오이다. 두번째 주기에서 명령 처리에 대한 일시중지가 성공적으로 이루어졌고, 203번 명령에 대한 유효시간이 만료된 이후에 재시작 요청이 전달되었다. 이러한 경우에는 명령이 실행되고 있지 않음을 응답한다.

#### 4.3.2.3.1.3 일회성 이벤트 명령

일회성 이벤트 명령은 주어진 이벤트 조건을 만족할 때 즉, 이벤트가 발생하였을 때 일회성 명령을 수행하는 형태의 명령이다. 이벤트는 연속성 명령(ContinuousCmd)과는 달리 발생주기가 완료 시간이 존재하지 않는다. 또한 어댑터의 완료보고 메시지(FinishRpt) 지원 여부에 상관없이 일회성 이벤트 명령에 대해서는 완료 보고 메시지(FinishRpt)는 전달되지 않는다.

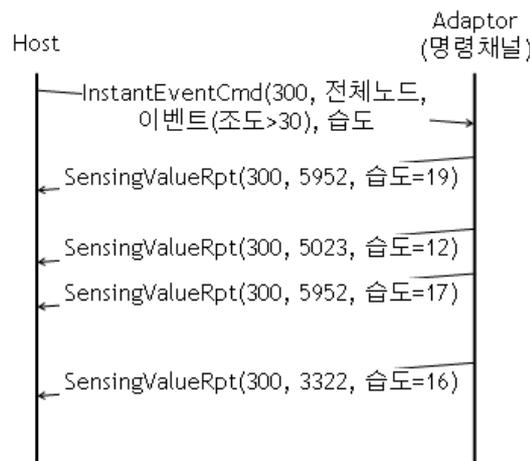


그림 4- 35 이벤트 명령의 처리 시나리오

위의 시나리오에서는 전체노드에서 조도가 30보다 큰 이벤트가 발생하였을 때 습도를 센싱하여 보고하는 300번 명령이 어댑터로 전송되는 경우에 대한 시나리오이다.

시간에 대한 제약조건 없이 각 노드는 언제라도 조도 값이 30이상이면 습도값을 센싱하여 보고한다. 다음 시나리오에서는 5952, 5023, 5952, 3322번 노드에서 순차적으로 이벤트가 발생되었고, 각 노드에 대한 습도 값을 보고하였다. 만약 주어진 이벤트가 발생하지 않는다면, 센싱값 보고서 또한 보고되지 않게된다.

4.3.2.3.1.4 일회성 집계명령과 보고

일회성 집계 명령은 한번 센싱된 결과 값을 수집하여 최소값, 최대값, 평균 등의 결과를 계산하여 보고하는 형태의 명령이다. 그러므로, 조건에 따라 결과는 하나이거나 없을 수 있다.

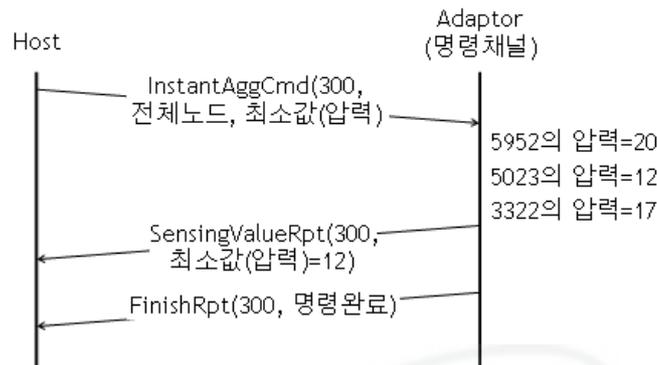


그림 4- 36 일회성 집계 명령의 처리 시나리오

위의 시나리오에서는 전체노드에서 센싱된 압력 값 중에서 최소값을 보고하는 300번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 있는 어댑터로 전송된 경우에 대한 시나리오이다. 어댑터는 각 노드에서 센싱된 압력값 중에서 가장 작은 압력 값을 보고하고, 명령 완료를 알린다.

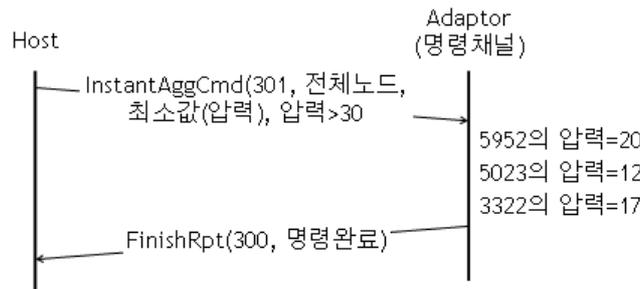


그림 4- 37 일회성 집계 명령의 처리 시나리오 - 조건 처리

위의 시나리오에서는 전체노드에서 센싱된 압력 값 중에서 30보다 큰 값들의 최소값을 보고하는 301번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 있는 어댑터로 전송된 경우이다. 전체노드에서 센싱된 압력값은 모두 30을 넘지 않으므로 결과 또한 보고되지 않았으며, 명령 완료를 알리는 완료 보고 메시지(FinishRpt)만이 전달되었다.

4.3.2.3.1.5 연속성 집계명령과 보고

연속성 집계 명령은 주어진 시간동안 주기적으로 센싱된 결과 값을 수집하여 최소값, 최대값, 평균 등의 결과를 계산하여 보고하는 형태의 명령이다. 그러므로, 조건에 따라 결과는 하나이거나 없을 수 있다.

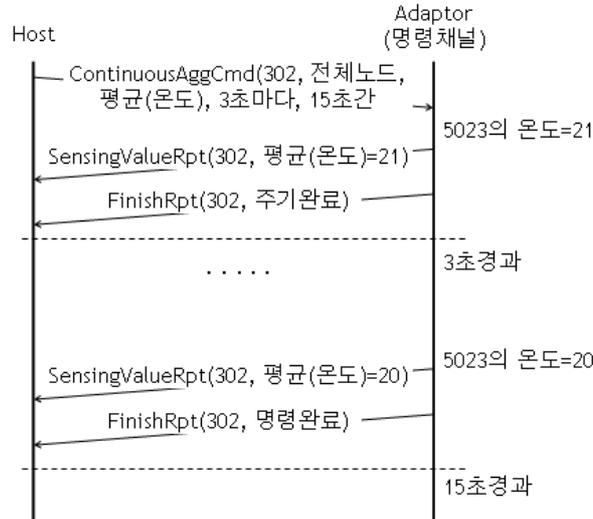


그림 4- 38 연속성 집계 명령의 처리 시나리오 - 주기처리

위의 시나리오에서는 전체노드에서 센싱된 온도값의 평균을 3초마다 15초간 보고하는 302번 명령이 어댑터로 전송된 경우이다. 전체노드에서 온도센서가 존재하는 노드는 5023번 노드만 존재하므로 해당 노드의 값을 수신받아 평균값을 전달하게 된다. 매 주기마다 센싱값을 전달한 다음에 주기완료임을 전달하고, 마지막 주기에는 명령이 완료되었음을 알린다.

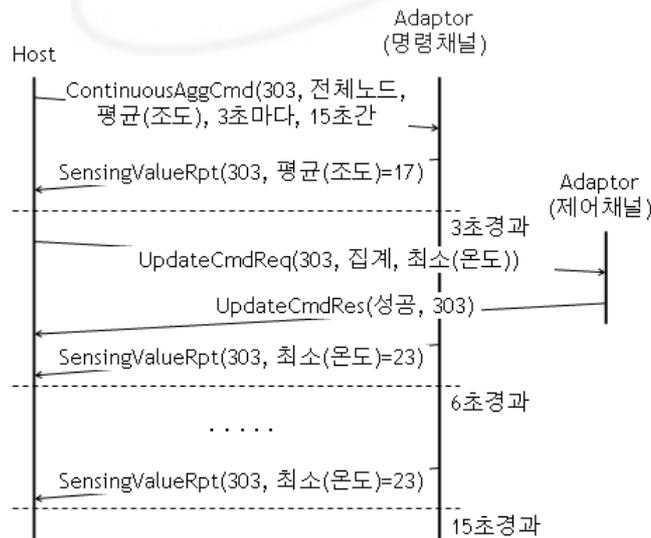


그림 4- 39 연속성 집계 명령과 집계정보 갱신 시나리오

위의 시나리오는 전체노드에서 센싱된 조도값의 평균을 3초마다 15초간 보고하는 303번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전송되고, 이후에 집계정보 갱신 요청이 전송된 경우에 대한 시나리오이다. 두번째 주기에서 집계

정보가 최소의 “평균(조도)”에서 “최소(온도)”로 변경 요청이 전달되고, 이는 성공적으로 처리되었다. 이후로는 온도에 대한 최소값이 계산되어 전달된다.

#### 4.3.2.3.2 구동기 명령/보고 메시지

메시지 이름	메시지유형 값	설명
RunActuatorCmd	0x7A00	구동기 동작 명령
RunActuatorRpt	0xA001	구동기 실행 결과 보고

표 4- 14 구동기 명령, 보고 메시지

구동기 동작 명령은 주어진 노드에 장착된 구동기를 작동시키고 그 결과를 보고하여야 한다. 센싱 명령과 마찬가지로 어댑터가 완료 보고 메시지(FinishRpt)를 처리할 수 있다면, 이를 전송하여야 한다.

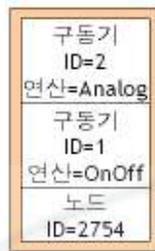


그림 4- 40 구동기 명령 처리 예제를 위한 센서네트워크 구성 예

위의 그림과 같이 OnOff 연산 및 아날로그 연산이 가능한 각각의 구동기가 장착된 2754번 노드가 존재할 때의 동작 시나리오는 다음과 같다.

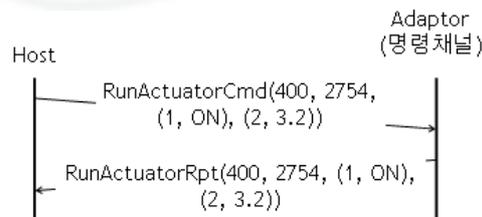


그림 4- 41 구동기 명령의 처리 시나리오

위의 시나리오에서는 2754번 노드의 1번 구동기는 켜고, 2번 구동기는 3.2에 해당하는 값으로 작동시키는 400번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 없는 어댑터로 전달된 경우이다. 각 구동기는 주어진 값에 맞도록 모두 작동되었고 명령에 따라 작동된 상태를 보고하게 된다.

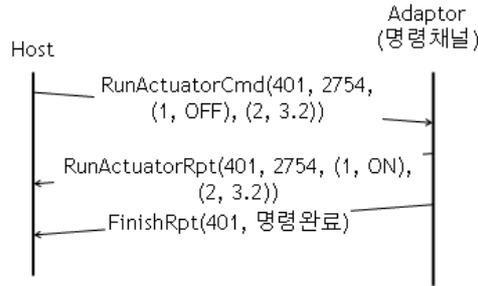


그림 4- 42 구동기 명령의 오류 처리 시나리오

위의 시나리오에서는 2754번 노드의 1번 구동기는 끄고, 2번 구동기는 기존과 같이 3.2에 해당되는 값으로 작동시키는 401번 명령이 완료 보고 메시지(FinishRpt)를 처리할 수 있는 어댑터로 전달된 경우이다. 명령 수행 결과 1번 구동기는 끄기에 실패하여 여전히 켜져있는 상태이고, 2번 구동기는 기존과 동일한 값으로 동작 중임이 보고되고 완료 보고 메시지(FinishRpt)가 전달된다.

아날로그 연산이 가능한 구동기는 주어진 아날로그 값에 의한 동작과 전원의 켜짐/꺼짐에 관한 On/Off 연산이 분리되어 전달된다.

#### 4.3.2.3.3 모니터링 명령/보고 메시지

메시지 이름	메시지유형 값	설명
MonitoringStartCmd	0x8000	모니터링 시작 명령
MonitoringStopCmd	0x8001	모니터링 중단 명령
MonitoringRpt	0xB000	모니터링 결과 보고

표 4- 15 모니터링 명령, 보고 메시지

모니터링은 기본적으로 3가지 방식으로 처리된다. 어떤 방식으로 처리되는가는 센서네트워크의 동작모드에 따라 결정된다. 다음은 모니터링 관련 처리능력이 모두 제공되는 경우로부터 수동적으로 값을 올려주기만하는 센서네트워크 순서로 정리하였다.

- 주기적 모니터링을 수행하는 센서네트워크  
모니터링 시작명령(MonitoringStartCmd)에 주기값(interval)을 전달받고, 해당 주기마다 모니터링값을 모니터링 결과 보고 메시지(MonitoringRpt)를 이용하여 호스트에 보고한다. 이는 모니터링 중단 명령(MonitoringStopCmd)을 수신할때까지 반복한다.
- 주기적 모니터링을 수행하지 못하나 모니터링 시작명령(MonitoringStartCmd)을 처리할 수 있는 센서네트워크  
모니터링 시작 명령(MonitoringStartCmd)을 수신하면 해당 명령에 기술된 모니터링 정보를 수집하여 모니터링 결과보고 메시지(MonitoringRpt)를 이용하여 한번 보고한다. 모니터링 중단 명령(MonitoringStopCmd)은 전달되지 않는다.

- 수집된 모니터링 정보만을 보고할 수 있는 센서네트워크  
 수집된 모니터링정보를 모니터링 결과 보고 메시지(MonitoringRpt)를 이용해서 호스트에 보고한다. 모니터링 시작명령(MonitoringStartCmd)이나 모니터링 종료명령(MonitoringStopCmd)은 사용되지 않는다.

모니터링 명령 수행 방법을 확인하기 위하여 다음 그림과 같이 구성된 센서네트워크를 가정하자. 센서네트워크에는 식별자가 5952, 5023, 3322인 3개의 노드가 존재하고, 각 노드는 모니터링 파라미터 상태, 위치, 부모노드 식별자, 노드간 연결강도를 모니터링 할 수 있다.



그림 4- 43 모니터링 명령 처리 예제를 위한 센서네트워크 구성 예

위와 같이 구성된 센서네트워크에서 모니터링 처리 결과는 다음과 같이 보고되어야 한다. 주기적인 모니터링을 수행할 수 있는 센서네트워크의 경우에는 모니터링 시작명령(MonitoringStartCmd)을 수신하였을 때, 해당 명령에 기술된 주기마다 모니터링 파라미터를 수집하여 보고한다. 매 주기마다 수집된 값을 보고하다가 모니터링 종료명령(MonitoringStopCmd)이 도달하면 수집 및 보고를 중단한다.

주기적인 모니터링을 수행하지 못하는 센서네트워크의 경우에는 모니터링 시작명령(MonitoringStartCmd)을 수신하면 모니터링 주기를 무시하고 한번만 모니터링 파라미터를 수집하여 보고한다. 이에 대한 모니터링 종료 명령(MonitoringStopCmd)은 별도로 전달되지 않는다.

수집된 모니터링 파라미터의 값은 모니터링 보고 메시지(MonitoringRrpt)를 통해 호스트로 전달된다. 모니터링 보고는 센싱값 보고 메시지(SensingValueRpt)와는 달리 완료 메시지(FinishRpt)가 존재하지 않는다.

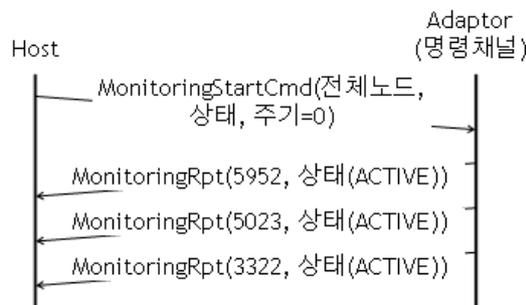


그림 4- 44 모니터링 시나리오 - 주기적인 처리가 불가능한 경우

위의 시나리오에서는 주기적인 모니터링을 수행하지 못하는 센서네트워크의 전체노드 상태를 모니터링하기 위한 명령이 전달되었다. 각 노드는 현재상태를 보고하고,

모니터링 명령은 종료된다. 주기적인 모니터링을 수행하지 못하는 센서네트워크이므로 별도의 모니터링 종료 메시지(MonitoringStopCmd)는 존재하지 않는다.

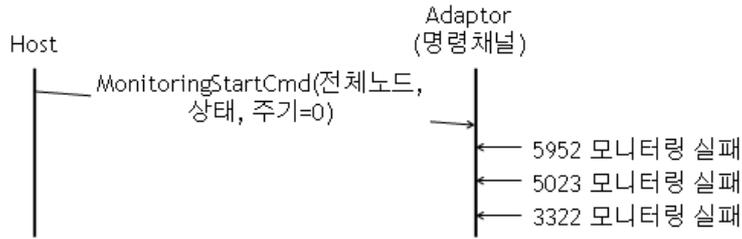


그림 4- 45 모니터링 시나리오 - 처리에 실패한 경우

위의 시나리오에서는 주기적인 모니터링을 지원하지 못하는 센서네트워크의 전체노드 상태를 모니터링하기 위한 명령이 전달되었는데, 모든 노드는 모니터링 수행에 실패하였다. 이러한 경우 어댑터는 호스트로 어떠한 내용도 보고하지 않는다. 만약, 하나의 노드에서만 모니터링에 성공하였다면, 결과는 하나만 전달되게 된다.

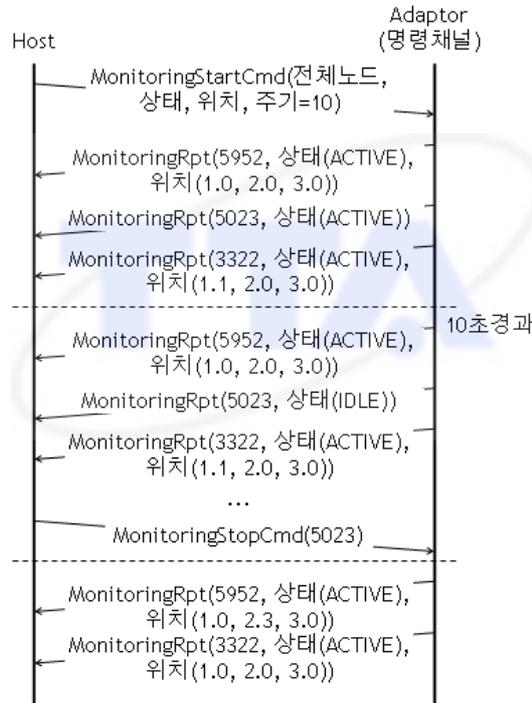


그림 4- 46 모니터링 시나리오 - 주기적인 처리가 가능한 경우

위의 시나리오에서는 주기적인 모니터링을 수행할 수 있는 센서네트워크의 전체노드의 상태와 위치를 10초간격으로 모니터링하기 위한 명령이 전달되었다. 각 노드는 매 10초마다 파라미터에 해당되는 모니터링 값을 수집하여 전달한다. 그런데, 5023번 노드는 위치정보를 모니터링 할 수 없는 노드이다. 이러한 경우 노드는 수집가능한 모니터링 파라미터의 값만을 수집하여 전달한다. 만약, 수집할 수 있는 항목이 존재하지 않는다면 아무런 내용도 보고하지 않을 것이다.

수집되는 모니터링 파라미터를 보고하는 중에 5023번 노드에 대한 모니터링 중단 명령(MonitoringStopCmd)이 전달되었다. 이러한 경우 5023번 노드는 모니터링을 즉

시 중단하여야 한다. 다른 노드들(5952, 3322번 노드)은 모니터링을 계속 수행한다.

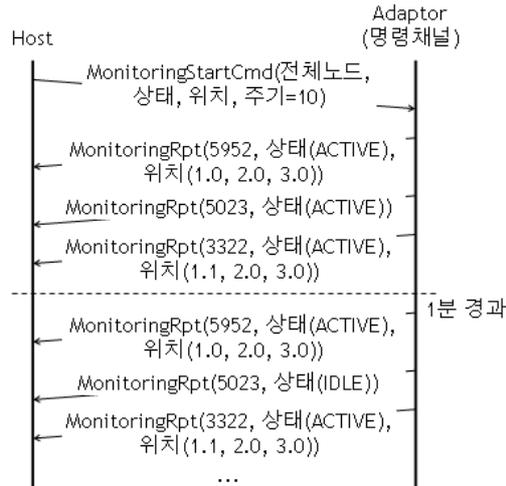


그림 4- 47 모니터링 시나리오 - 고정된 모니터링 주기의 경우

위의 시나리오에서는 모니터링 주기가 1분으로 고정되어있는 센서네트워크에 10초마다 상태를 보고하라는 모니터링 명령이 전달되었다. 이러한 경우 모니터링 시작 명령(MonitoringStartCmd)의 주기는 무시되며, 어댑터는 미리 고정되어 있던 1분 간격으로 모니터링을 수행하여 보고한다.

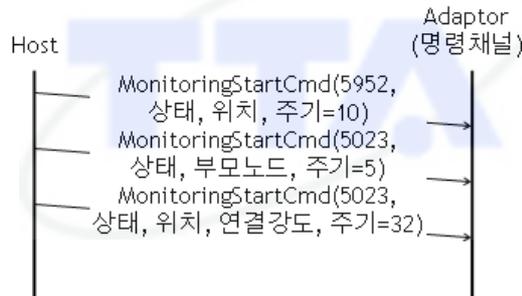


그림 4- 48 모니터링 시나리오 - 노드별 모니터링

위의 시나리오에서는 각 노드에 대해 서로 다른 모니터링 주기로 모니터링 명령(MonitoringStartCmd)이 전달되었다. 이러한 경우 해당 노드는 수집된 정보를 해당 주기마다 보고하면 된다. 모니터링 명령(MonitoringStartCmd)을 시작할 수 없거나 수집된 파라미터가 하나도 존재하지 않는 경우에는 모니터링 보고 메시지(MonitoringRpt)를 전달하지 않는다.

4.3.2.3.4 에러보고 메시지

메시지 이름	메시지유형 값	설명
ErrorRpt	0xC000	에러 보고

표 4- 16 에러보고 메시지

센싱/모니터링 명령등 호스트가 어댑터 측으로 전달되는 명령이나 센서네트워크는 동작 중에 에러가 발생할 수 있다. 어댑터 측은 이러한 에러 상황을 에러보고 메시지(ErrorRpt)를 통해 호스트에 보고한다. 에러 보고 메시지(ErrorRpt)에 포함되는 정보에는 에러의 유형과 각 유형에 따른 에러 코드값 그리고 확인을 위한 부가정보가 있다.

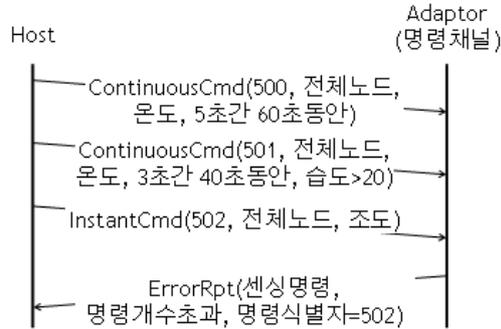


그림 4- 49 에러보고 시나리오 - 수행할 수 있는 명령 개수가 초과된 경우

위의 시나리오에서는 동시에 수행 가능한 명령의 개수가 2개인 센서네트워크에 세 번째 명령이 전달되었다. 센서네트워크는 가장 마지막에 도달한 503번 명령에 대한 처리를 포기하고, 이를 에러 보고 메시지(ErrorRpt)를 통해 전달 하였다. 처리를 포기 할 명령은 어댑터 구현에 따른다.

명령개수 초과에 대한 에러 보고는 센싱값을 버퍼링하고 있거나 혹은 수행중인 명령의 제어처리에 실패한 경우 등 호스트에서는 해당 명령에 대한 처리를 예상하고 새로운 명령을 전달하였지만, 실제로는 명령에 대한 처리가 종료되지 않은 경우를 명시적으로 판별하기 위하여 존재한다.

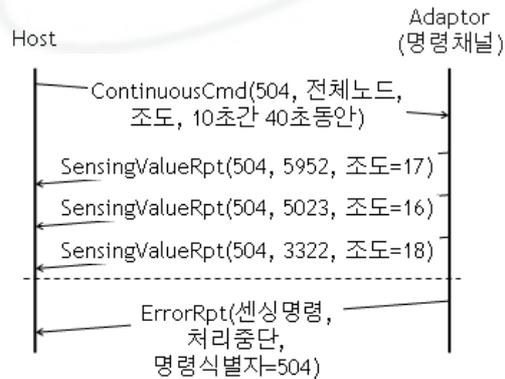


그림 4- 50 에러보고 시나리오 - 명령처리 중단

위의 시나리오에서는 수행중인 명령에 대한 처리가 중단된 경우에 대한 에러 보고 예제이다. 센싱명령에 대한 처리가 중단되었다는 것은 주어진 504번 명령에 대한 센싱값에 대한 보고가 더 이상 이루어지지 않음을 의미한다.

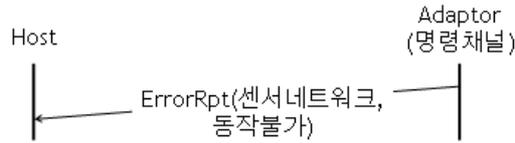


그림 4- 51 에러보고 시나리오 - 센서네트워크 동작 불가

위의 시나리오에서는 센서네트워크의 동작이 중단된 경우에 대한 에러 보고 예제이다. 이러한 경우 센서네트워크 내에서 수행 중이던 센싱 혹은 모니터링 명령은 모두 사라지게 된다. 후속 처리는 어댑터 구현 방법에 따른다.

4.3.2.3.5 에러보고 유형과 코드 값

- 에러보고 유형 (ErrorType)
  - 0x00: 센싱/구동기 명령
    - ◆ 모든 에러보고 코드 값이 유효
  - 0x01: 센서네트워크
    - ◆ 에러코드 0x01, 0xFF 만이 유효
  
- 에러보고 코드 (ErrorCode)
  - 0x00: 명령 수행 개수를 초과하여 더 이상 수행할 수 없음
  - 0x01: 명령 수행이 중단 되었음 혹은 센서네트워크 동작이 중단되었음
  - 0xFE: 명령은 분석되었으나, 내부 요소들을 지원하지 않음
  - 0xFF: 명령 처리를 시도하였으나 처리에 실패하였음 혹은 센서네트워크에서 알 수 없는 문제가 발생하였음

4.3.2.3.6 네트워크 갱신 보고 메시지

메시지 이름	메시지유형 값	설명
UpdateRpt	0xC001	상태 변경 보고

표 4- 17 네트워크 갱신 보고 메시지

센서네트워크에서 발생된 노드나 트랜스듀서 추가, 삭제 그리고 어댑터와 센서네트워크 사이의 연결 상태에 대한 정보는 갱신 보고 메시지(UpdateRpt)를 통해 전달된다.

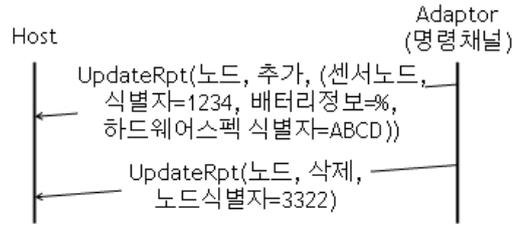


그림 4- 52 네트워크 갱신 보고 시나리오 - 노드 추가, 삭제

위의 시나리오에서는 센서네트워크에 새로운 노드 1234번이 추가되고, 기존에 존재하는 3322번 노드가 삭제되었다. 노드에 대한 추가는 최소한의 정적 메타 정보를 포함하며, 삭제는 식별자 정보만을 포함한다.

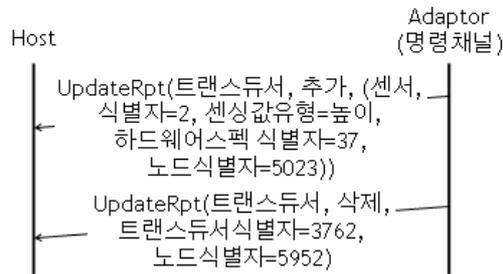


그림 4- 53 네트워크 갱신 보고 시나리오 - 트랜스듀서 추가, 삭제

위의 시나리오에서는 노드 식별자 5023에 새로운 센서 2번이 추가되고, 5952노드에서 3762번 센서가 삭제되었다. 노드와는 달리 트랜스듀서에 대한 추가 삭제는 해당 트랜스듀서가 어떤 노드에 포함되는 것인지에 대한 정보도 함께 전달된다.

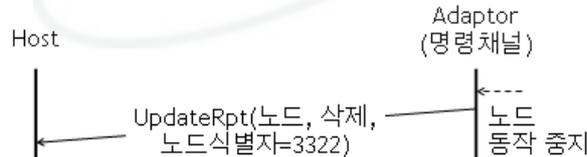


그림 4- 54 네트워크 갱신 보고 시나리오 - 동작 중지된 노드

위의 시나리오에서는 동작 중지된 노드에 대한 처리 방법을 표현하고 있다. 노드에 대한 동작이 불가능한 경우 별도의 에러 보고는 없으며 단지 해당 노드가 네트워크에서 제거되었음을 알린다. 이후 호스트는 해당 노드에 더 이상의 명령을 전달하지 않으며, 이미 수행 중인 명령이 존재하였다면 해당 노드에서는 센싱값 보고가 더 이상 전달되지 않을 것임을 판단할 수 있다. 트랜스듀서 또한 동일한 방법으로 동작된다.

#### 4.3.2.4 확인

##### 4.3.2.4.1 채널확인

메시지 이름	메시지유형 값	설명
--------	---------	----

ChannelCheckCtrl	0xD000	채널 확인
ChannelConfirmCtrl	0xD001	채널 확인 응답

표 4- 18 채널 확인 메시지

어댑터나 호스트는 연결 설정 후에 다양한 메시지를 주고 받을 수 있다. 그러나, 어댑터가 센싱된 값을 버퍼링 하고 있다가 한꺼번에 이를 호스트로 전송하는 경우 혹은 응용 프로그램에서 호스트로 질의를 전달하기 전에는 채널을 통해 주고 받을 메시지가 없을 수 있다. 이러한 경우 시스템에 의해 강제적으로 채널 연결이 종료될 수 있는데, 이를 방지하고 또한 호스트와 어댑터간의 채널 연결이 정상적으로 이루어져 있음을 확인하기 위해 채널 검사를 수행할 수 있다.

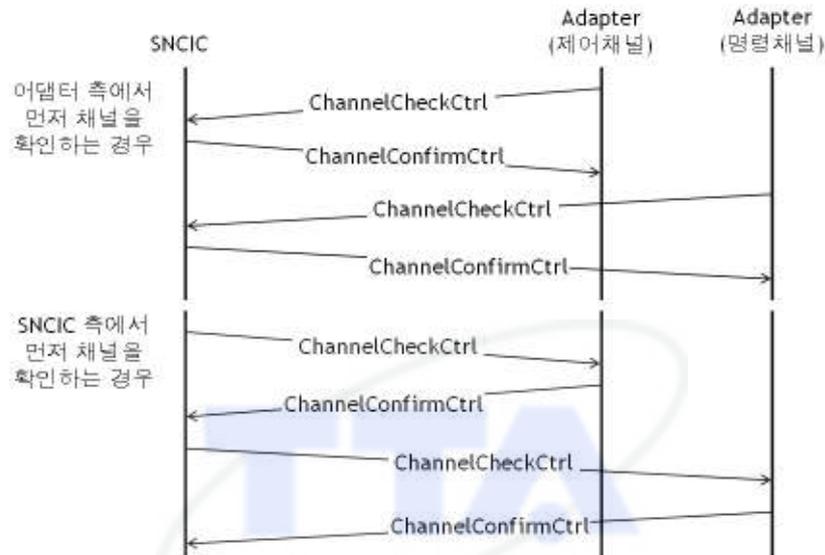


그림 4- 55 채널 확인 시나리오

채널검사는 어댑터나 호스트가 상대방의 각 채널에 채널 확인 메시지 (ChannelCheckCtrl)를 전달하고, 이를 수신한 측에서 채널 확인 응답 메시지 (ChannelConfirmCtrl)를 전달 함으로써 이루어진다. 채널 확인에 대한 응답 메시지가 수신되지 않은 경우에 대한 처리는 구현에 따른다.

4.3.2.4.2 오류 확인

메시지 이름	메시지유형 값	설명
NakChk	0xDFFF	메시지 오류 전달

표 4- 19 오류확인 메시지

호스트와 어댑터는 상대방으로부터 전달받은 메시지의 프로토콜 버전과 메시지 타입을 확인하여 이상이 확인되면, 이를 상대방에 알린다. 프로토콜버전은 연결설정 정보 요청자에 의해 최초로 전송되는 연결설정 정보 요청 메시지(ConnReqCtrl )에 기술된 프로토콜 버전을 기준으로 한다.

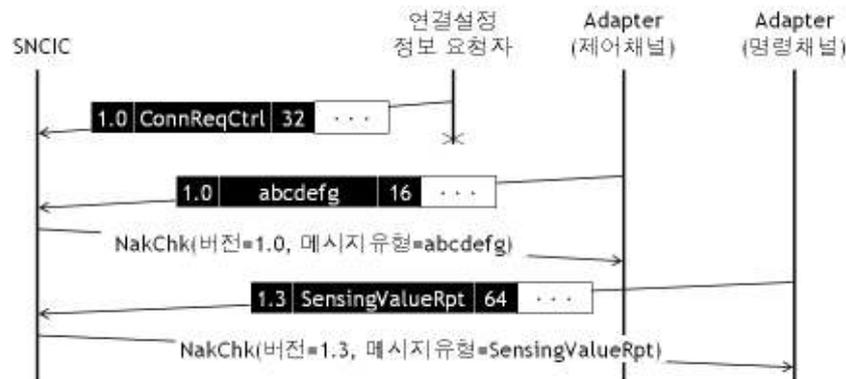


그림 4- 56 오류확인 메시지 전송 시나리오

위의 시나리오는 오류확인 메시지(NakChk) 전송 시나리오이다. 연결설정 정보 요청자에 의해 최초로 전송된 메시지는 1.0 버전을 따르고 있다. 이후 제어채널에서 메시지 규격에 존재하지 않는 형태의 메시지를 전달하였다. 이를 감지한 호스트는 해당 메시지의 헤더 정보를 오류확인 메시지(NakChk)에 담아 전송하였다.

명령채널에서는 최초 전송된 메시지 버전 1.0과는 다른 1.3 버전의 메시지를 전송하였다. 이러한 경우도 해당 메시지의 헤더를 오류확인 메시지(NakChk)에 담아 상대방에 전송한다. 오류감지에 따른 이후 메시지 처리는 해당 채널로 수신된 메시지를 버리고 이후로 수신된 메시지부터 처리하도록 한다.

### 4.3.3 예약된 값

#### 4.3.3.1 노드/트랜스듀서 식별자 (8bytes)

- 전체: 0x8000 0000 0000 0000

#### 4.3.3.2 명령 식별자

- 의미없는 명령 식별자: 0x7FFF FFFF FFFF FFFE

#### 4.3.3.3 시간의 표현

- 연속질의의 lifetime 이 무한대인 경우의 값: 0x7FFFFFFF
- 시간 정보를 알 수 없는 경우의 값: 0

## 5. 바이트 스트림 메시지

호스트와 어댑터 간의 메시지 전송은 다양한 방법으로 이루어질 수 있는데, 이 절에서는 TCP/IP를 이용하여 메시지를 교환하기 위해 필요한 바이트 스트림 형태의 메시지 프로토콜에 대해 기술한다.

5장의 필드중 다음 필드는 연관된 항목을 통해서 그 내용을 확인할 수 있다.

SensorNetworkID, NodeID: 향후, 표준으로 정의될 센서네트워크 id 식별자 체계와 센서노드 id 식별자 체계에 따른 센서네트워크 id와 센서 노드 id를 포함하는 필드임.

SensingTypeID: 센서네트워크 공통 인터페이스는 센서 네트워크의 메타데이터를 저장/검색/수정/관리하는 모듈을 사용함을 가정한다. 센서노드 혹은 어댑터의 처리 수준이 매우 다양하므로 공통메시지를 이용하는 경우, 해당 도메인별로 부록에 예시된 바와 같이 센서 데이터 유형 정의표를 정의하여 메타데이터 관리 모듈을 통해서 통일되게 이용할 수 있도록한다.

### 5.1 메시지 구조

메시지는 다음과 같은 구조를 갖는다.

메시지 헤더	프로토콜 버전 (2 bytes)	메시지 유형 (2 bytes)	몸체길이 (2bytes)
메시지 바디	Body ( n bytes)		

그림 5- 1 바이트 스트림 메시지 구조

#### □ 메시지 헤더

##### ◆ 프로토콜 버전

- 호스트와 어댑터간의 메시지 전송규약에 대한 버전을 표현
- 상위 1 바이트는 Major 버전을, 하위 1 바이트는 Minor 버전을 표현
  - Major 버전: 1 ~ 255 사이의 정수 값
  - Minor 버전: 0 ~ 255 사이의 정수 값

##### ◆ 메시지 유형

- 호스트와 어댑터 간에 교환되는 메시지의 종류를 표현

##### ◆ 몸체 길이

- 메시지 헤더를 제외한 메시지 몸체의 길이를 표현. 메시지 바디가 가질 수 있는 값은 0 ~ 65535 임

#### □ 메시지 바디

- ◆ 메시지 유형에 따른 몸체를 표현

### 5.2 공통 상수

### 5.2.1 노드 유형

- 길이: 1 바이트
- 값
  - 0x00: 센서노드, 0x01: RFID 리더, 0x02: IP-USN 노드

### 5.2.2 트랜스듀서 유형

- 길이: 1 바이트
- 값
  - 0x01: 센서, 0x02: 구동기

### 5.2.3 트랜스듀서 연산 유형

- 길이: 2 바이트
- 값
  - 트랜스듀서가 센서인 경우: 센싱값 유형 식별자
  - 트랜스듀서가 구동기인 경우
    - ◆ 0x00: OnOff 연산이 가능한 구동기
    - ◆ 0x01: 아날로그 값에 기반하여 동작하는 구동기

### 5.2.4 구동기 동작 유형

- 길이: 1 바이트
- 값
  - OnOff 연산 : 0x00
  - Analog 연산 : 0x01

### 5.2.5 구동기 동작 값

- 길이 : n 바이트
  - 구동기 동작 유형이 OnOff 연산인 경우 : 1 바이트
    - ◆ 0x00 : 꺼짐(off), 0x01 : 켜짐(on)
  - 구동기 동작 유형이 아날로그 연산인 경우 : 4 바이트 실수
    - ◆ 구동될 값을 표현

### 5.2.6 연산자

#### 5.2.6.1 논리연산자

- 길이: 1 바이트
- 값
  - 0x00: 없음, 0x01: and, 0x02: or
- 논리 연산자의 우선순위는 왼쪽에서 오른쪽이다.
  - 일반적인 NOT, AND, OR 순의 우선순위를 사용하지 않는다.

### 5.2.6.2 관계연산자

- 길이: 1 바이트
- 값
  - 0x00: !=, 0x01: ==, 0x02: <, 0x03: <=, 0x04: >, 0x05: >=, 0xFF: 없음

### 5.2.7 함수

- 길이: 1 바이트
- 값
  - 0x00: 최소값, 0x01: 최대값, 0x02: 합계, 0x03: 평균, 0x04: 개수, 0xFF: 없음

### 5.2.8 네트워크 동작유형

- 길이: 1 바이트
- 값
  - 0x00: push, 0x01: pull

### 5.2.9 배터리 정보 표현 유형

- 길이: 1 바이트
- 값
  - 0x00: 전원 정보를 지원하지 않음
  - 0x01: 영구전원이 공급됨
  - 0x02: 현재의 전압 값(V)을 실수로 표현
  - 0x03: 남아있는 전원을 백분율(% , 실수)로 표현 (0%~100%)
  - 0x04: 향후 동작 가능한 시간(단위: 초, 정수)을 표현
  - 0x05: 남아있는 전류의 양(단위: mA, 정수)을 표현
  - 0x06: 전원 부족 여부만을 표현 (정수)
    - ◆ 0x00: 전원부족, 0x01: 부족하지 않음

### 5.2.10 모니터링 파라미터

- 길이: 2 바이트
- 값
  - 0x0001: 상태, 0x0002: 위치, 0x0004: 부모노드 식별자, 0x0008: 노드간 연결강도, 0x0010: 잔존 배터리 양, 0x11: 노드 sw 버전, 0x12: DSP 상태

## 5.3 연결제어 메시지

### 5.3.1 ReqConnCtrl

호스트가 어댑터에 호스트에 연결설정 정보 요청 메시지(ConnReqCtrl)를 전송할 것을 요청한다.

(1) MessageType: 0x0001

(2) MessageBody

이름	크기	설명
SensorNetworkID	8	센서네트워크 식별자

표 5- 1 ReqConnCtrl 메시지의 바디

### 5.3.2 ConnReqCtrl

연결설정 요청자가 호스트에 연결설정을 요청하기 위한 메시지이다.

(1) MessageType: 0x0002

(2) MessageBody

이름	크기	설명
SensorNetworkID	8	어댑터가 표현하는 센서네트워크 식별자

표 5- 2 ConnReqCtrl 메시지의 바디

- SensorNetworkID: 센서네트워크를 구분하기 위한 식별자

### 5.3.3 ConnResCtrl

호스트가 연결 요청자에게 접속에 관한 정보를 전달하기 위한 메시지이다.

(1) MessageType: 0x0003

(2) MessageBody

이름	크기	설명
RetCode	1	연결설정요청에 대한 응답
EncryptMode	1	보안 설정 등급 표현 RetCode가 성공인 경우에만 표현됨
ConnString	n	연결이 허가 되었을 경우 어댑터가 연결할 서버주소 및 제어채널, 명령채널의 포트번호를 문자열로 표현 RetCode가 성공인 경우에만 표현됨

표 5- 3 ConnInfoCtrl 메시지의 바디

- RetCode 의 값
  - 표 4- 12 요청에 대한 응답 코드 참고
- EncryptMode 의 값
  - 0x00 : 기밀성 비적용
  - 0x01 : 메시지 보안 - 교환되는 메시지단위 선택적 보안 적용
  - 0x02 : 통신 보안 - TLS 를 통해 통신 채널 전체에 보안 적용
  - 0x03 : 메시지 보안 + 통신보안 - TLS 를 통해 통신 채널 전체에 보안이 적용되  
면서 각각의 메시지는 암호화
- ConnString 의 값

연결정보 문자열은 다음과 같은 형태로 표현되며, 각 채널은 동일한 포트를 사용할 수 있다.

[tcpip|http]://서버주소/info?ControlPort=XXXX&CommandPort=XXXX

- 서버주소: 연결 설정할 서버의 주소
- ControlPort=XXXX: 제어채널의 포트번호
- CommandPort=XXXX: 명령채널의 포트번호

### 5.3.4 DisConnReqCtrl

어댑터나 호스트에서 상대방에 연결종료를 요청하기 위한 메시지이다.

- (1) MessageType: 0x000F
- (2) MessageBody: 없음

### 5.3.5 AuthReqCtrl

어댑터의 각 채널이 채널에 대한 인증을 요청하는 메시지이다.

- (1) MessageType: 0x0100
- (2) MessageBody

이름	크기	설명
SensorNetworkID	8	센서네트워크 식별자
ChannelType	1	채널 유형
CredentialLength	2	Credential의 길이
Credential	n	채널이 자신을 입증하기 위한 정보 채널 인증하지 않는 경우에는 의미없음
AddressLength	1	명령/제어 채널에 대한 주소 길이 (Binary/TCP 통신의 경우 항상 0)
Address	n	명령/제어 채널 주소 (통신방식이 비연결성을 가지는 경우사용)

표 5- 4 AuthReqCtrl 메시지 의 바디

- ChannelType: 채널 유형
  - 0x02: 제어채널, 0x04: 명령채널
  - 0x06: 제어채널이면서 동시에 명령채널인 경우 (0x02 | 0x04)
- Credential: 채널이 자신을 입증하기 위한 정보
  - USSC 상세 설계서 참고
- Address : 채널에 대한 주소  
연결정보 문자열은 다음과 같은 형태로 표현되며, 각 채널은 동일한 포트를 사용할 수 있다.

[tcpip|http]://서버주소/info?ControlPort=XXXX&CommandPort=XXXX

- 서버주소: 연결 설정할 서버의 주소

- ControlPort=XXXX: 제어채널의 포트번호
- CommandPort=XXXX: 명령채널의 포트번호

### 5.3.6 AuthResCtrl

어댑터의 각 채널에 대한 인증 결과를 전달하기 위한 메시지이다.

- (1) MessageType: 0x0101
- (2) MessageBody

이름	크기	설명
RetCode	1	인증 성공/실패 코드
Result	N	향후 메시지에 대한 보안 적용에 사용될 정보 인증에 성공하고, 메시지에 대한 암호/복호화를 수행하는 경우에만 표현됨

표 5- 5AuthResCtrl 메시지의 바디

- RetCode 의 값
  - 표 4- 12 요청에 대한 응답 코드 참고
- Result: 메시지 보안이 적용되는 암호/복호화 키 및 세션정보
  - USSC 상세 설계서 참고



## 5.4 요청/응답 메시지

### 5.4.1 NetworkInfoReq

호스트에서 어댑터에 센서네트워크에 대한 정보를 요청하기 위한 메시지이다.

- (1) MessageType: 0x1000
- (2) MessageBody: 없음

### 5.4.2 NetworkInfoRes

어댑터에서 호스트로부터 전달받은 네트워크 정보 요청 메시지(NetworkInfoReq)에 응답하기 위한 메시지로, 요청에 대한 처리결과 및 센서네트워크 정보를 전달하기 위한 메시지이다.

- (1) MessageType: 0x4000
- (2) MessageBody

이름		크기	설명	
LinkType		1	메시지 응답 유형	
RetCode		1	요청에 대한 처리 결과 응답유형이 처음인 경우 유효	
Mode		1	네트워크 동작 모드 응답유형이 처음인 경우 유효	
NodeNum		1	노드 정보 개수	
반 복	NodeType	1	노드의 유형	
	NodeID	8	노드 식별자	
	BatteryType	1	노드 잔존 배터리 정보 표현 유형	
	NodeHWSpecID	8	노드의 하드웨어 스펙 식별자	
	TransducerNum	1	노드에 연결된 트랜스듀서의 수를 표현	
	반 복	TransducerType	1	트랜스듀서의 유형
		TransducerTypeID	8	트랜스듀서 식별자
		TransducerOpType	2	트랜스듀서 연산 유형
TransducerHWSpecID		8	트랜스듀서의 하드웨어 스펙 ID를 표현	

표 5- 6 NetworkInfoRes의 메시지 바디

- LinkType: 요청에 응답 메시지가 처음인지 마지막인지 혹은 응답 중인지 표현
  - 0x01: 처음, 0x02: 중간, 0x04: 마지막
- RetCode 의 값
  - 표 4- 12 요청에 대한 응답 코드 참고
- Mode: 센서네트워크 동작 유형
  - 0 5.2.8 네트워크 동작유형 참고
- NodeType: 노드의 유형
  - 0 5.2.1 노드 유형 참고

- BatteryType: 노드의 배터리 정보 표현 유형
  - 0 5.2.9 배터리 정보 표현 유형 참고
- TransducerType: 트랜스듀서의 유형
  - 0 5.2.2 트랜스듀서 유형 참고
- TransducerOpType: 트랜스듀서의 연산유형 혹은 센싱값 유형 식별자
  - 0 5.2.3 트랜스듀서 연산 유형 참고

### 5.4.3 BufferDataReq

호스트가 어댑터에 버퍼링된 데이터의 전송을 요청하기 위한 메시지이다.

- (1) MessageType: 0x1001
- (2) MessageBody: 없음

### 5.4.4 BufferDataRes

버퍼링된 데이터의 전송 요청을 받아 처리하고, 처리 결과를 알려주기 위한 응답 메시지이다.

- (1) MessageType: 0x4001
- (2) MessageBody

이름	크기	설명
RetCode	1	요청에 대한 처리 결과

표 5- 7 BufferDataRes의 메시지 바디

- RetCode 값
  - 표 4- 12 요청에 대한 응답 코드 참고

### 5.4.5 CmdActionReq

실행중인 명령에 대한 제어를 요청하기 위한 메시지이다.

- (1) MessageType: 0x2000
- (2) MessageBody

이름	크기	설명
CommandID	4	제어의 대상이 되는 명령 식별자
Operation	1	제어 방법

표 5- 8 CmdActionReq의 메시지 바디

- Operation 값
  - 0x00: 중지, 0x01: 일시중지, 0x02: 재시작

### 5.4.6 CmdActionRes

명령제어 요청에 대한 응답을 위한 메시지이다.

- (1) MessageType: 0x5000
- (2) MessageBody

이름	크기	설명
RetCode	1	요청에 대한 처리 결과
CommandID	4	제어의 대상이된 명령 식별자

표 5-9 CmdActionRes의 메시지 바디

- RetCode 값
  - 표 4- 12 요청에 대한 응답 코드 참고

### 5.4.7 UpdateCmdReq

실행중인 명령의 파라미터를 변경하기 위한 메시지이다.

- (1) MessageType: 0x2001
- (2) MessageBody

이름	크기	설명
CommandID	4	갱신할 명령 식별자
UpdateType	1	갱신할 파라미터 항목
Parameter	n	갱신할 파라미터 정보

표 5- 10 UpdateCmdReq 메시지 바디

- UpdateType 값
  - 0x01: 센싱조건, 0x02: 시간, 0x04: 함수
- Parameter 값
  - UpdateType 의 값이 센싱조건(0x00) 혹은 이벤트 발생 조건(0x01)인 경우

이름		크기	설명
Num		1	조건의 개수
반 복	SensingTypeID	2	조건이 주어지는 센싱값유형 식별자
	OpLogical	1	다음 조건과의 논리연산자
	OpRelational	1	SensingTypeID와 Value의 관계연산자
	Value	n	비교할 값을 표현

표 5- 11 UpdateCmdReq 메시지의 조건 갱신

- ◆ OpLogical: 논리 연산자
  - 0 5.2.6.1 논리연산자 참고
- ◆ OpRelational: 관계 연산자
  - 0 5.2.6.2 관계연산자 참고
- UpdateType 의 값이 시간(0x02)인 경우

이름	크기	설명
----	----	----

TimeInterval	4	센싱할 시간 간격
LifeTime	4	명령이 유효한 시간

표 5- 12 UpdateCmdReq 메시지의 시간 갱신

- UpdateType 의 값이 함수(0x04)인 경우

이름		크기	설명
Num		1	집계정보 개수
반	FunctionType	1	함수유형
복	SensingTypeID	2	센싱값 유형 식별자

표 5- 13 UpdateCmdReq 메시지의 함수 갱신

- ◆ Function: 집계 함수
  - 0 5.2.7 함수 참고

### 5.4.8 UpdateCmdRes

실행중인 명령의 파라미터 변경 결과를 응답하기 위한 메시지이다.

- (1) MessageType: 0x5001
- (2) MessageBody

이름	크기	설명
RetCode	1	요청에 대한 처리 결과
CommandID	4	갱신된 명령 식별자

표 5- 14 UpdateCmdRes 메시지의 바디

- RetCode 값
  - 표 4- 12 요청에 대한 응답 코드 참고

### 5.4.9 ControlNetworkReq

센서네트워크를 제어하기 위한 메시지이다.

- (1) MessageType: 0x3000
- (2) MessageBody

이름	크기	설명
ControlType	1	제어유형
ControlValue	4	제어에 필요한 값

표 5- 15 ControlNetworkReq 메시지의 바디

- ControlType 과 ControlValue 값

ControlType의 값	Scope	ControlValue값의 의미
----------------	-------	-------------------

리셋(0x00)	Network or node 단위	의미없음
켜기/끄기(0x01)	Network or node 단위	0x00: 끄기, 0x01: 켜기
시간동기(0x02)	Network or node 단위	의미없음
송신파워설정(0x03)	Network or node 단위	송신 출력
DSP초기화(0x04)	Network or node 단위	센서노드의 DSP초기화
채널변경(0x06)	Network 단위	변경될 채널 번호 (정수)
모드변경(0x07)	Network 단위	0x00: push 방식 동작, 0x01: pull 방식 동작
센서전원제어(0x10)	Node 단위	개별 센서에대한 on/off/sleep
센서gain제어(0x11)	Node 단위	개별 센서 gain값
센서샘플링rate설정 (0x12)	Node 단위	개별 센서 샘플링 rate값
부모노드 설정(0x13)	Node 단위	부모노드 식별자

표 5- 16 ControlNetworkReq의 제어 유형과 제어에 필요한 값

#### 5.4.10 ControlNetworkRes

센서네트워크 제어 결과를 응답하기 위한 메시지이다.

- (1) MessageType: 0x6000
- (2) MessageBody

이름	크기	설명
RetCode	1	요청에 대한 처리결과

표 5- 17 ControlNetworkRes 메시지의 바디

- RetCode 값
  - 표 4- 12 요청에 대한 응답 코드 참고
  - 단, 센서네트워크 리셋의 경우 “성공”은 해당 메시지를 수행할 것임을 의미하며 반드시 성공한다는 보장은 하지 않는다.

#### 5.4.11 ControlNodeReq

노드를 제어하기 위한 메시지이다.

- (1) MessageType: 0x3001
- (2) MessageBody

이름	크기	설명
NodeID	8	제어의 대상이 되는 노드 식별자
ControlType	1	제어의 유형
ControlValue	8	제어에 필요한 값

표 5- 18 ControlNodeReq 메시지의 바디

○ ControlType 과 ControlValue 값

ControlType의 값	ControlValue값의 의미
리셋(0x00)	의미없음
켜기/끄기(0x01)	0x00: 끄기, 0x01: 켜기
부모노드 설정(0x02)	부모노드 식별자

표 5- 19 ControlNodeReq의 제어 유형과 제어에 필요한 값

5.4.12 ControlNodeRes

노드 제어 결과를 응답하기 위한 메시지이다.

- (1) MessageType: 0x6001
- (2) MessageBody

이름	크기	설명
RetCode	1	요청에 대한 처리 결과
NodeID	8	제어의 대상이된 노드 식별자

표 5- 20 ControlNodeRes 메시지의 바디

○ RetCode 값

- 표 4- 12 요청에 대한 응답 코드 참고

5.5 명령 메시지

5.5.1 InstantCmd

한번 값을 센싱하여 보고하는 일회성 명령을 표현하기 위한 메시지이다.

- (1) MessageType: 0x7000
- (2) MessageBody

이름	크기	설명	
CommandID	4	명령 식별자	
NodeNum	2	명령이 전달될 노드 식별자의 개수	
반복	NodeID	8	명령이 전달될 노드 식별자
SensingTypeIDNum	1	센싱값 유형의 개수	
반복	SensingTypeID	2	센싱값 유형 식별자
ConditionNum	1	조건의 개수	
조건	SensingTypeID	2	조건이 주어지는 센싱값 유형 식별자
	OpLogical	1	다음 조건과의 논리연산자
반복	Operator	1	SensingTypeID와 Value간의 연산자
	Value	n	비교될 값

표 5- 21 InstantCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드로 명령이 전달됨을 의미한다.
- ConditionNum 의 값이 0 이라면 “조건반복”부분이 존재하지 않는다.
- OpLogical: 논리 연산자
  - 0 5.2.6.1 논리연산자 참고
- Operator: 관계 연산자
  - 0 5.2.6.2 관계연산자 참고

### 5.5.2 ContinuousCmd

주어진 시간 동안 값을 센싱하여 보고하는 연속성 명령을 표현하기 위한 메시지이다.

(1) MessageType: 0x7001

(2) MessageBody

이름		크기	설명
CommandID		4	명령 식별자
NodeNum		2	명령이 전달될 노드 식별자의 개수
반복	NodeID	8	명령이 전달될 노드 식별자
SensingTypeIDNum		1	센싱값 유형의 개수
반복	SensingTypeID	2	센싱값 유형 식별자
TimeInterval		2	센싱할 시간 간격을 초단위로 표현
LifeTime		4	명령의 종료시점을 초단위로 표현
ConditionNum		1	조건의 개수
조건	SensingTypeID	2	조건이 주어지는 센싱값 유형 식별자
	OpLogical	1	다음 조건과의 논리연산자
반복	Operator	1	SensingTypeID와 Value간의 연산자
	Value	n	비교될 값

표 5- 22 ContinuousCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드로 명령이 전달됨을 의미한다.
- ConditionNum 의 값이 0 이라면 “조건반복”부분이 존재하지 않는다.
- OpLogical: 논리 연산자
  - 0 5.2.6.1 논리연산자 참고
- Operator: 관계 연산자
  - 0 5.2.6.2 관계연산자 참고
- TimeInterval: 단위는 초(sec)이다.
- LifeTime : 각 노드에서 명령을 전달 받은 시점으로부터 LifeTime(단위: 초)에 기술된 시간 후에 명령이 종료된다.

### 5.5.3 InstantEventCmd

이벤트가 발생하였을 때 한번 값을 센싱하여 보고하고, 다음 이벤트 발생을 대기한다.

- (1) MessageType: 0x7002
- (2) MessageBody

이름		크기	설명
CommandID		4	명령 식별자
NodeNum		2	명령이 전달될 노드 식별자의 개수
반복	NodeID	8	명령이 전달될 노드 식별자
EventConditionNum		1	이벤트 발생 조건 개수
반복	SensingTypeID	2	조건이 주어지는 센싱값 유형 식별자
	OpLogical	1	다음 조건과의 논리연산자
	Operator	1	SensingTypeID와 Value간의 연산자
	Value	n	비교될 값
SensingTypeIDNum		1	센싱값 유형의 개수
반복	SensingTypeID	2	센싱값 유형 식별자

표 5- 23 InstantEventCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드로 명령이 전달됨을 의미한다.
- EventConditionNum 은 0 일 수 없다.
- ConditionNum 의 값이 0 이라면 “조건반복”부분이 존재하지 않는다.
- OpLogical: 논리 연산자
  - 0 5.2.6.1 논리연산자 참고
- Operator: 관계 연산자
  - 0 5.2.6.2 관계연산자 참고

### 5.5.4 InstantAggCmd

센싱값을 한번 수집하여 집계하여 보고한다.

- (1) MessageType: 0x7004
- (2) MessageBody

이름		크기	설명
CommandID		4	명령 식별자
NodeNum		2	명령이 전달될 노드 식별자의 개수
반복	NodeID	8	명령이 전달될 노드 식별자
FunctionNum		1	함수의 개수
반복	Function	1	함수
	SensingTypeID	2	함수가 적용될 센싱값 유형 식별자
ConditionNum		1	조건의 개수

조건	SensingTypeID	2	조건이 주어지는 센싱값 유형 식별자
	OpLogical	1	다음 조건과의 논리연산자
반복	Operator	1	SensingTypeID와 Value간의 연산자
	Value	n	비교될 값

표 5- 24 InstantAggCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드로 명령이 전달됨을 의미한다.
- FunctionNum 은 0 일 수 없다.
- ConditionNum 의 값이 0 이라면 “조건반복”부분이 존재하지 않는다.
- Function: 집계함수
  - 0 5.2.7 함수 참고
- OpLogical: 논리 연산자
  - 0 5.2.6.1 논리연산자 참고
- Operator: 관계 연산자
  - 0 5.2.6.2 관계연산자 참고

### 5.5.5 ContinuousAggCmd

주어진 시간 마다 센싱값을 수집하여 집계하고, 보고한다.

- (1) MessageType: 0x7005
- (2) MessageBody

이름		크기	설명
CommandID		4	명령 식별자
NodeNum		2	명령이 전달될 노드 식별자의 개수
반복	NodeID	8	명령이 전달될 노드 식별자
FunctionNum		1	함수의 개수
반복	Function	1	함수
	SensingTypeID	2	함수가 적용될 센싱값 유형 식별자
TimeInterval		2	센싱할 시간 간격을 초단위로 표현
LifeTime		4	명령의 종료시점을 초단위로 표현
ConditionNum		1	조건의 개수
조건	SensingTypeID	2	조건이 주어지는 센싱값 유형 식별자
	OpLogical	1	다음 조건과의 논리연산자
반복	Operator	1	SensingTypeID와 Value간의 연산자
	Value	n	비교될 값

표 5- 25 ContinuousAggCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드로 명령이 전달됨

을 의미한다.

- FunctionNum 은 0 일 수 없다.
- ConditionNum 의 값이 0 이라면 “조건반복”부분이 존재하지 않는다.
- Function: 집계함수
  - 0 5.2.7 함수 참고
- OpLogical: 논리 연산자
  - 0 5.2.6.1 논리연산자 참고
- Operator: 관계 연산자
  - 0 5.2.6.2 관계연산자 참고
- TimeInterval: 단위는 초(sec)이다.
- LifeTime : 각 노드에서 명령을 전달 받은 시점으로부터 LifeTime(단위: 초)에 기술된 시간 후에 명령이 종료된다.

### 5.5.6 RunActuatorCmd

구동기 실행명령을 어댑터에 전달하기 위한 메시지이다.

- (1) MessageType: 0x7A00
- (2) MessageBody

이름		크기	설명
CommandID		4	명령의 식별자
NodeID		8	명령이 전달될 노드 식별자
ActuatorNum		1	동작시킬 구동기의 개수
반 복	ActuatorID	8	작동될 구동기의 식별자
	ActionType	1	구동기 동작 유형
	Value	n	구동기 동작 값

표 5- 26 RunActionCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드임을 의미한다.
- ActionType 값
  - 0 5.2.4 구동기 동작 유형 참고
- Value 값
  - 0 5.2.5 구동기 동작 값 참고

### 5.5.7 MonitoringStartCmd

정해진 주기마다 주어진 모니터링 항목 정보를 수집하여 보고한다.

- (1) MessageType: 0x8000
- (2) MessabeBody

이름	크기	설명
NodeNum	2	노드 식별자 개수

반복	NodeID	8	모니터링 대상이 되는 노드 식별자
	ParamNum	1	파라미터 개수
반복	Parameter	2	모니터링 대상 항목
	TimeInterval	4	모니터링 정보 수집/보고 주기

표 5- 27 MonitoringStartCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드임을 의미한다.
- Parameter: 모니터링 파라미터
  - 0 5.2.10 모니터링 파라미터 참고
- TimeInterval: 단위는 초(sec)이다.
  - TimeInterval 의 값이 0 인 경우, 모니터링 종료 메시지(MonitoringStopCmd)는 전달되지 않으며 주어진 노드는 한번만 모니터링을 수행한다.

### 5.5.8 MonitoringStopCmd

모니터링의 수행을 중단한다.

- (1) MessageType: 0x8001
- (2) MessabeBody

이름		크기	설명
	NodeNum	2	노드 식별자 개수
반복	NodeID	8	모니터링 대상이 되는 노드 식별자

표 5-28 MonitoringStopCmd의 메시지 바디

- NodeID 값이 0xFFFFFFFF 인 경우는 센서네트워크 내의 전체 노드임을 의미한다.

## 5.6 보고 메시지

### 5.6.1 SensingValueRpt

어댑터가 센싱된 값들을 보고하기 위한 메시지이다.

- (1) MessageType: 0xA000
- (2) MessageBody

이름		크기	설명
CommandID		4	어떤 명령에 대한 보고인지 표현하는 명령의 식별자
NodeID		8	센싱 값을 보고하는 노드의 식별자
SendingTime		4	노드가 메시지를 전송한 시간
SensingValueNum		1	센싱된 값의 개수
반 복	Function	1	함수명
	SensingTypeID	2	센싱값 유형 식별자
	SensingValue	n	센싱값
	TimeStamp	4	센싱된 시간

표 5- 29 SensingValueRpt의 메시지 바디

- SendingTime, TimeStamp
  - 1970년 1월 1일 이후로 흐른 시간을 초 단위로 표현한다.
  - 시간 정보를 알 수 없는 경우 0으로 한다.
- Function
  - 0 5.2.7 함수 참고
  - 함수의 종류가 '없음(0xff)'이 아닌 경우 NodeID 값은 의미없다. 즉, 무시된다.
- SensingValue
  - 센싱값 유형 식별자에 따라 그 크기와 각 바이트의 의미가 결정된다.
  - 기본 센싱유형의 경우, 표 5- 42에 따르며, 위치정보와 같이 구조체가 센싱값이 되는 경우는 추후 정의.

### 5.6.2 RunActuatorRpt

구동기 동작 명령을 실행하였음을 보고한다.

- (1) MessageType: 0xA001
- (2) MessageBody

이름		크기	설명
CommandID		4	명령 식별자
NodeID		8	명령이 전달된 노드 식별자
SendingTime		4	노드가 메시지를 전송한 시간
ActuatorNum		1	동작된 구동기의 개수
반	ActuatorID	8	동작된 구동기의 식별자

복	ActionType	1	구동기 동작 유형
	Value	4	구동기 동작 값

표 5- 30 RunActuatorRpt의 메시지 바디

- SendingTime
  - 1970년 1월 1일 이후로 흐른 시간을 초 단위로 표현한다.
- ActionType
  - 0 5.2.4 구동기 동작 유형 참고
- Value 값
  - 0 5.2.5 구동기 동작 값 참고

### 5.6.3 FinishRpt

명령에 대한 센싱값 보고가 완료되었음을 알린다.

- (1) MessageType: 0xAFFF
- (2) MessageBody

이름	크기	설명
FinishType	1	완료의 유형
CommandID	4	보고가 종료된 명령 식별자

표 5- 31 FinishRpt의 메시지 바디

- FinishType의 값
  - 0x00: 주기의 완료, 0x01: 명령의 완료

### 5.6.4 MonitoringRpt

모니터링 수행 결과를 보고하기 위한 메시지이다.

- (1) MessageType: 0xB000
- (2) MessageBody

이름	크기	설명	
NodeID	8	모니터링 결과를 보고하는 노드 식별자	
SendingTime	4	노드가 결과를 전송한 시간	
ValueNum	1	수집된 정보의 개수	
반복	ParameterID	2	파라미터 식별자
	Value	n	수집된 값

표 5- 32 MonitoringRpt 메시지의 바디

- ParameterID: 모니터링 파라미터
  - 0 5.2.10 모니터링 파라미터 참고
- Value 값

- ParameterID의 값이 상태(0x0001)인 경우 1byte로 표현
  - ◆ 0x00: active
    - 센서노드 : 패킷을 주고 받을 수 있는 통신 가능한 상태
    - RFID 리더 : 태그를 읽을 수 있도록 켜진 상태
  - ◆ 0x01: sleep
    - 센서노드 : 미리 정해진 시간동안 네트워크에 참여하지 않는 상태
    - RFID 리더 : 태그를 읽지 않는 상태
- ParameterID의 값이 위치(0x0002)인 경우 12byte로 표현

이름	크기	설명
X	4	X좌표를 실수로 표현
Y	4	Y좌표를 실수로 표현
Z	4	Z좌표를 실수로 표현

표 5- 33 위치정보 표현

- ParameterID의 값이 부모노드 식별자(0x0004)인 경우 8byte로 표현
    - ◆ 부모노드를 알 수 없는 경우: 0xFFFFFFFFFFFFFFFF
    - ◆ 부모노드가 존재하지 않는 경우: 해당 항목에 대한 응답을 전달하지 않음.
  - ParameterID의 값이 노드간 연결강도(0x0008)인 경우 2byte로 표현
    - ◆ 지원하는 경우: 0 ~ 255 사이의 값을 반환
      - 255가 가장 좋은 신호품질을 의미함
    - ◆ 지원하지 않는 경우: 0xffff를 반환
  - ParameterID의 값이 잔존 배터리 양(0x0016)인 경우 4byte로 표현
    - ◆ 전원 정보를 지원하지 않는 경우: 무시됨
    - ◆ 영구전원이 공급되는 경우: 무시됨
    - ◆ 배터리를 이용하는 경우는 다음과 같이 표현
      - 현재의 전압값(V): 실수
      - 남아있는 전원의 백분율(%): 실수 (0% ~ 100%)
      - 향후 동작 가능한 시간(단위: 초): 정수
      - 남아있는 전류의 양(mA): 정수
      - 전원부족
        - 0x00000000: 정상전압
        - 0x00000001: 전원부족
- 잔존 배터리 양 표시는 해당 노드의 배터리 양 표현 방법을 따르는데, 이는 노드 목록 요청에 대한 응답 메시지(NetworkInfoRes)에 표현된 방법과 동일해야 한다.
- SendingTime: 1970년 1월 1일 이후로 흐른 시간을 초 단위로 표현한다.

### 5.6.5 ErrorRpt

센서네트워크의 오류상황을 알려주기 위한 메시지이다.

- (1) MessageType: 0xC000
- (2) MessageBody

이름	크기	설명
ErrorType	1	에러가 발생한 대상을 표현
ErrorCode	1	ErrorType에 대한 에러코드
TimeStamp	4	에러 발생 또는 인지 시간을 초 단위로 표현
ErrorInfo	4	발생된 에러의 부가정보

표 5- 34 ErrorRpt 메시지의 바디

- ErrorType 과 ErrorCode 값
  - 0 4.3.2.3.5 에러보고 유형과 코드 값 참고
- TimeStamp
  - 1970 년 1 월 1 일 이후로 흐른 시간을 초단위로 표현한다.
  - 시간을 알 수 없는 경우에는 0 으로 표현한다.
- ErrorInfo
  - ErrorType 이 센싱/구동기 명령(0x00)인 경우
    - ◆ 명령 식별자 (4byte)

### 5.6.6 UpdateRpt

센서네트워크에 대한 변경 상황을 알려주기 위한 메시지이다.

- (1) MessageType: 0xC001
- (2) MessageBody

이름	크기	설명
ObjectType	1	갱신 대상정보
OpType	1	추가/연결 혹은 삭제/연결해지
UpdateInfo	n	추가/연결 혹은 삭제/연결해지 정보

표 5- 35 UpdateRpt 메시지의 바디

- ObjectType
  - 0x00: 노드, 0x01: 트랜스듀서, 0x02 : 센서네트워크
- OpType
  - 0x00: 추가/연결, 0x01: 삭제/연결해지
- UpdateInfo
  - 노드 추가인 경우 (ObjectType=0x00, OpType=0x00)

이름	크기	설명
Type	1	노드의 유형
NodeID	8	노드 식별자
BatteryType	1	배터리 정보 표현 유형

HWSpecID	8	하드웨어 스펙 식별자
----------	---	-------------

표 5- 36 노드 추가 갱신 메시지의 바디

- Type 의 값
  - 0 5.2.1 노드 유형 참고
- BatteryType 의 값
  - 0 5.2.9 배터리 정보 표현 유형 참고
- 노드 삭제인 경우 (ObjectType=0x00, OpType=0x01)

이름	크기	설명
NodeID	8	노드 식별자

표 5- 37 노드 삭제 갱신 메시지의 바디

- 트랜스듀서 추가인 경우 (ObjectType=0x01, OpType=0x00)

이름	크기	설명
Type	1	트랜스듀서 유형
TransducerID	8	트랜스듀서 식별자
OpType	2	연산유형 혹은 센싱값 유형 식별자
HwSpecID	8	하드웨어 스펙 식별자
NodeID	8	트랜스듀서가 포함된 노드 식별자

표 5- 38 트랜스듀서 추가 갱신 메시지의 바디

- ◆ Type 의 값
  - 0 5.2.2 트랜스듀서 유형 참고
- ◆ OpType 의 값
  - 0 5.2.3 트랜스듀서 연산 유형 참고
- ◆ HwSpecID 의 값
  - 추가되는 트랜스듀서의 하드웨어 스펙 식별자
- 트랜스듀서 삭제인 경우 (ObjectType=0x01, OpType=0x01)

이름	크기	설명
TransducerID	8	트랜스듀서 식별자
NodeID	8	노드 식별자

표 5- 39 트랜스듀서 삭제 갱신 메시지의 바디

- NodeID: 삭제될 트랜스듀서가 포함된 노드의 식별자

## 5.7 확인 메시지

### 5.7.1 ChannelCheckCtrl

각 채널에서 상대방으로의 채널이 유효함을 검사하기 위한 메시지이다.

- (1) MessageType: 0xD000
- (2) MessageBody: 없음

### 5.7.2 ChannelConfirmCtrl

채널검사에 대해 응답하기 위한 메시지이다.

- (1) MessageType: 0xD001
- (2) MessageBody: 없음

### 5.7.3 NakChk

호스트나 어댑터가 상대방으로부터 전달받은 메시지 헤더의 프로토콜 버전과 메시지 유형을 확인하여 문제가 있다면, 해당 메시지의 헤더에 포함된 프로토콜 버전과 메시지 유형을 상대방에 알려주기 위한 메시지이다.

- (1) MessageType: 0xDFFF
- (2) MessageBody

이름	크기	설명
ProtocolVersion	2	수신된 프로토콜 버전
MessageType	2	수신된 메시지 유형

표 5- 40 NakChk 메시지의 바디

## 6. XML 문서 메시지

SNCIC와 어댑터 간의 메시지 전송은 다양한 방법으로 이루어질 수 있는데, 이 절에서는 HTTP를 이용하여 메시지를 교환하기 위해 필요한 XML문서 형태의 메시지 프로토콜에 대해 기술한다.

### 6.1 메시지 구조

메시지는 다음과 같은 구조를 갖는다.

최상위 엘리먼트 시작	<mes:commonMessage sensorNetworkID="xxx" version="xxx" xmlns:mes="http://kr/re/etri/cosmos/abstracttier/sncic/schema/message">
메시지 엘리먼트	메시지
최상위 엘리먼트 종료	</mes:commonMessage>

그림 오류! 지정한 스타일은 사용되지 않습니다.6-2 XML 문서 메시지 구조

#### □ 최상위 엘리먼트

- ◆ 센서네트워크 식별자(sensorNetworkID)
  - 메시지를 전송하는 센서네트워크 식별자를 표현
- ◆ 프로토콜 버전(version)
  - 호스트와 어댑터 간의 메시지 전송규약에 대한 버전을 표현
  - 상위 1 바이트는 Major 버전을, 하위 1 바이트는 Minor 버전을 표현하는 값을 정수로 표현
    - Major 버전: 1 ~ 255 사이의 정수 값
    - Minor 버전: 0 ~ 255 사이의 정수 값

#### □ 메시지

- ◆ 주고 받을 메시지를 표현

## 6.2 HTTP 응답

### 6.2.1 알 수 없는 클라이언트

수신된 XML 문서는 유효하지만, 인증되지 않은 클라이언트로부터 오는 메시지는 HTTP 응답코드 400(Bad request)을 전달한다.

### 6.2.2 유효하지 않은 메시지

#### 6.2.2.1 XML 문서가 포함되지 않은 경우

HTTP 응답 코드 400(Bad Request)를 전달한다.

#### 6.2.2.2 포함된 XML 문서가 유효하지 않는 경우

HTTP 응답 코드 400(Bad Request)와 함께 메시지 확인 메시지(NakChk)를 전달한다.

### 6.2.3 수신할 수 없는 유형의 메시지

수신된 XML 문서는 유효하지만, 수신할 수 없는 형태의 메시지에는 SNCIC 측의 일회성 명령 메시지(InstantCmd)가 있다. 이는 항상 SNCIC에서 어댑터로 전달되는 메시지이기 때문이다. 어댑터 측에서는 센싱값 보고 메시지(SensingValueRpt)를 예로 들 수 있다.

이러한 경우 인증된 상대에게서 오는 메시지는 확인 메시지(NakChk)를 통해 이를 알리고, 그렇지 않은 경우에는 응답코드 400(Bad request)를 전달한다.

### 6.2.4 연결제어/확인 메시지

#### 6.2.4.1 연결설정 정보 요청 (ConnReqCtrl)

수신된 메시지가 연결설정 정보 요청 메시지(ConnReqCtrl)인 경우 이에 대한 응답은 연결설정 정보 응답 메시지(ConnResCtrl)로 해야 한다. HTTP 응답 코드는 200(OK)으로 한다.

#### 6.2.4.2 채널인증 (AuthReqCtrl)

수신된 메시지가 채널 인증 요청 메시지(AuthReqCtrl)인 경우 이에 대한 응답은 채널인증 응답 메시지(AuthResCtrl)로 해야 한다. HTTP 응답 코드는 200(OK)으로 한다.

#### 6.2.4.3 채널확인 (ChannelCheckCtrl)

수신된 메시지가 채널확인 요청 메시지(ChannelCheckCtrl)인 경우 이에 대한 응답은 HTTP 응답코드 200(OK)으로 한다. 채널확인 응답 메시지(ChannelConfirmCtrl)는 이후에 전달되어야 한다. 그렇지 않으면 상대방은 채널 종료임을 판단하고 메시지 처리를 중단할 수 있기 때문이다. 채널확인 요청 메시지(ChannelCheckCtrl)에 대한 응답을 채널확인 응답 메시지(ChannelConfirmCtrl)로 지정하지 않는 것은 이 메시지는 다른 메시지와 함께 상대방에 전달될 수 있기 때문이다.

### 6.2.5 요청/응답 메시지

수신된 메시지가 요청 메시지(NetworkInfoReq, BufferDataReq, CmdActionReq, UpdateCmdReq, ControlNetworkReq, ControlNodeReq)인 경우 이에 대한 응답은 상응되는 응답 메시지(NetworkInfoRes, BufferDataRes, CmdActionRes, UpdateCmdRes, ControlNetworkRes, ControlNodeRes)로 해야 하며, HTTP 응답코드는 200(OK)으로 한다.

### 6.2.6 명령 메시지

수신된 메시지가 명령 메시지(InstantCmd, ContinuousCmd, InstantEventCmd, InstantAggCmd, ContinuousAggCmd, RunActuatorCmd, MonitoringStartCmd, MonitoringStopCmd)인 경우 이에 대한 HTTP 응답은 200(OK)으로 한다.

### 6.2.7 보고 메시지

수신된 메시지가 보고 메시지(SensingValueRpt, RunActuatorRpt, FinishRpt, MonitoringRpt, ErrorRpt, UpdateRpt)인 경우 이에 대한 HTTP 응답은 200(OK)으로 한다.

### 6.3 메시지 스키마

```
<?xml version="1.0" encoding="euc_kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msg="http://kr/re/etri/cosmos/snat/sncic/schema/message"
targetNamespace="http://kr/re/etri/cosmos/snat/sncic/schema/message"
elementFormDefault="qualified">

<!-- 최상위 엘리먼트 -->
<xsd:element name="commonMessage" type="msg:commonMessage"/>

<!-- 공통 메시지 종류 -->
<xsd:complexType name="commonMessage">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="plainMessage" type="msg:plainMessage"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="encryptMessage" type="msg:encryptMessage"
        minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="version" type="xsd:int" use="required"/>
  <xsd:attribute name="type" type="msg:messageType" use="required"/>
  <xsd:attribute name="sensorNetworkID"
    type="xsd:long" use="required"/>
</xsd:complexType>

<!-- 암호화된 공통 메시지 -->
<xsd:complexType name="encryptMessage">
  <xsd:sequence>
    <!-- 이 값을 복원하면, plainMessage 가 되어야 함 -->
    <xsd:element name="message" type="xsd:base64Binary"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- 암호화 되지 않은 공통 메시지 -->
<xsd:complexType name="plainMessage">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <!-- 연결제어 메시지 -->
      <xsd:element name="reqConnCtrl"
        type="msg:reqConnCtrl"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="connReqCtrl"
        type="msg:connReqCtrl"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="connResCtrl"
        type="msg:connResCtrl"

```

```

        minOccurs="1" maxOccurs="1"/>
<xsd:element name="authReqCtrl"
        type="msg:authReqCtrl"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="authResCtrl"
        type="msg:authResCtrl"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="disConnReqCtrl"
        type="msg:disConnReqCtrl"
        minOccurs="1" maxOccurs="1"/>

<!-- 확인 메시지 -->
<xsd:element name="channelCheckCtrl"
        type="msg:channelCheckCtrl"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="channelConfirmCtrl"
        type="msg:channelConfirmCtrl"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="nakChk"
        type="msg:nakChk"
        minOccurs="1" maxOccurs="1"/>

<!-- 요청 -->
<xsd:element name="networkInfoReq"
        type="msg:networkInfoReq"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="bufferDataReq"
        type="msg:bufferDataReq"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="cmdActionReq"
        type="msg:cmdActionReq"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="updateCmdReq"
        type="msg:updateCmdReq"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="controlNetworkReq"
        type="msg:controlNetworkReq"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="controlNodeReq"
        type="msg:controlNodeReq"
        minOccurs="1" maxOccurs="1"/>

<!-- 응답 -->
<xsd:element name="networkInfoRes"
        type="msg:networkInfoRes"
        minOccurs="1" maxOccurs="1"/>

<xsd:element name="cmdActionRes"
        type="msg:cmdActionRes"
        minOccurs="1" maxOccurs="1"/>
<xsd:element name="bufferDataRes"
        type="msg:bufferDataRes"

```

```
minOccurs="1" maxOccurs="1"/>
```

```
<xsd:element name="updateCmdRes"
  type="msg:updateCmdRes"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="controlNetworkRes"
  type="msg:controlNetworkRes"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="controlNodeRes"
  type="msg:controlNodeRes"
  minOccurs="1" maxOccurs="1"/>
```

```
<!-- 명령 -->
```

```
<xsd:element name="instantCmd"
  type="msg:instantCmd"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="continuousCmd"
  type="msg:continuousCmd"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="instantEventCmd"
  type="msg:instantEventCmd"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="instantAggCmd"
  type="msg:instantAggCmd"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="continuousAggCmd"
  type="msg:continuousAggCmd"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="runActuatorCmd"
  type="msg:runActuatorCmd"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="monitoringStartCmd"
  type="msg:monitoringStartCmd"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="monitoringStopCmd"
  type="msg:monitoringStopCmd"
  minOccurs="1" maxOccurs="1"/>
```

```
<!-- 보고 -->
```

```
<xsd:element name="sensingValueRpt"
  type="msg:sensingValueRpt"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="runActuatorRpt"
  type="msg:runActuatorRpt"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="finishRpt"
  type="msg:finishRpt"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="monitoringRpt"
  type="msg:monitoringRpt"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="errorRpt"
  type="msg:errorRpt"
  minOccurs="1" maxOccurs="1"/>
```

```

        <xsd:element name="updateRpt"
                    type="msg:updateRpt"
                    minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
</xsd:sequence>
</xsd:complexType>

<!-- ReqConnCtrl -->
<xsd:complexType name="reqConnCtrl"/>

<!-- ConnReqCtrl -->
<xsd:complexType name="connReqCtrl"/>

<!-- ConnResCtrl -->
<xsd:complexType name="connResCtrl">
    <xsd:sequence>
        <xsd:element name="retCode" type="msg:retCodeType"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="encryptMode" type="msg:encryptModeType"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="connString" type="xsd:string"
                    nillable="true" minOccurs="1"
maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- AuthReqCtrl -->
<xsd:complexType name="authReqCtrl">
    <xsd:sequence>
        <xsd:element name="channelType" type="msg:channelType"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="credential" type="xsd:base64Binary"
                    nillable="true" minOccurs="1"
maxOccurs="1"/>
        <xsd:element name="address" type="xsd:string"
                    nillable="true" minOccurs="1"
maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- AuthResCtrl -->
<xsd:complexType name="authResCtrl">
    <xsd:sequence>
        <xsd:element name="retCode" type="msg:retCodeType"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="result" type="xsd:base64Binary"
                    nillable="true" minOccurs="1"
maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- DisConnReqCtrl -->
<xsd:complexType name="disConnReqCtrl"/>

```

```

<!-- ChannelCheckCtrl -->
<xsd:complexType name="channelCheckCtrl"/>

<!-- ChannelConfirmCtrl -->
<xsd:complexType name="channelConfirmCtrl"/>

<!-- NakChk -->
<xsd:complexType name="nakChk">
  <xsd:sequence>
    <xsd:element name="version" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="messageType" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- NetworkInfoReq -->
<xsd:complexType name="networkInfoReq"/>

<!-- NetworkInfoRes -->
<xsd:complexType name="networkInfoRes">
  <xsd:sequence>
    <xsd:element name="retCode" type="msg:retCodeType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="mode" type="msg:networkModeType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="nodeInfo" type="msg:nodeMetaType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- BufferDataReq -->
<xsd:complexType name="bufferDataReq"/>

<!-- BufferDataRes -->
<xsd:complexType name="bufferDataRes">
  <xsd:sequence>
    <xsd:element name="retCode" type="msg:retCodeType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- CmdActionReq -->
<xsd:complexType name="cmdActionReq">
  <xsd:sequence>
    <xsd:element name="commandID" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="action" type="msg:commandActionType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- CmdActionRes -->
<xsd:complexType name="cmdActionRes">

```

```

<xsd:sequence>
  <xsd:element name="retCode" type="msg:retCodeType"
    minOccurs="1" maxOccurs="1"/>
  <xsd:element name="commandID" type="xsd:int"
    minOccurs="1" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>

<!-- UpdateCmdReq -->
<xsd:complexType name="updateCmdReq">
  <xsd:sequence>
    <xsd:element name="commandID" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="condition"
        type="msg:condition"
        minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="continuousTime"
        type="msg:continuousTime"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="aggregationInfo"
        type="msg:aggregationInfo"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="type" type="msg:updateCmdType" use="required"/>
</xsd:complexType>

<!-- UpdateCmdRes -->
<xsd:complexType name="updateCmdRes">
  <xsd:sequence>
    <xsd:element name="retCode" type="msg:retCodeType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="commandID" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- ControlNetworkReq -->
<xsd:complexType name="controlNetworkReq">
  <xsd:sequence>
    <xsd:element name="control" type="msg:networkControlType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="channel" type="xsd:int"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="mode" type="msg:networkModeType"
        minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<!-- ControlNetworkRes -->
<xsd:complexType name="controlNetworkRes">

```

```

<xsd:sequence>
  <xsd:element name="retCode" type="msg:retCodeType"
    minOccurs="1" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>

<!-- ControlNodeReq -->
<xsd:complexType name="controlNodeReq">
  <xsd:sequence>
    <xsd:element name="nodeID" type="xsd:long"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="control" type="msg:nodeControlType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="onOff" type="xsd:boolean"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="parentNodeID" type="xsd:long"
        minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<!-- ControlNodeRes -->
<xsd:complexType name="controlNodeRes">
  <xsd:sequence>
    <xsd:element name="retCode" type="msg:retCodeType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="nodeID" type="xsd:long"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- InstantCmd -->
<xsd:complexType name="instantCmd">
  <xsd:sequence>
    <xsd:element name="commandID" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="nodeID" type="xsd:long"
      minOccurs="1" maxOccurs="unbounded"/>
    <xsd:element name="sensingTypeID" type="msg:sensingTypeID"
      minOccurs="1" maxOccurs="unbounded"/>
    <xsd:element name="condition" type="msg:condition"
      nillable="true" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- ContinuousCmd -->
<xsd:complexType name="continuousCmd">
  <xsd:complexContent>
    <xsd:extension base="msg:instantCmd">
      <xsd:sequence>
        <xsd:element name="continuousTime"
          type="msg:continuousTime"
          minOccurs="1" maxOccurs="1"/>

```

```

        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- InstantEventCmd -->
<xsd:complexType name="instantEventCmd">
    <xsd:sequence>
        <xsd:element name="commandID" type="xsd:int"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="nodeID" type="xsd:long"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="sensingTypeID" type="msg:sensingTypeID"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="eventCondition" type="msg:condition"
            minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<!-- InstantAggCmd -->
<xsd:complexType name="instantAggCmd">
    <xsd:sequence>
        <xsd:element name="commandID" type="xsd:int"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="nodeID" type="xsd:long"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="aggregationInfo" type="msg:aggregationInfo"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="condition" type="msg:condition"
            illable="true" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<!-- ContinuousAggCmd -->
<xsd:complexType name="continuousAggCmd">
    <xsd:complexContent>
        <xsd:extension base="msg:instantAggCmd">
            <xsd:sequence>
                <xsd:element name="continuousTime"
                    type="msg:continuousTime"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- RunActuatorCmd -->
<xsd:complexType name="runActuatorCmd">
    <xsd:sequence>
        <xsd:element name="commandID" type="xsd:int"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="nodeID" type="xsd:long"
            minOccurs="1" maxOccurs="1"/>

```

```

        <xsd:element name="actuation" type="msg:actuationType"
                    minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<!-- MonitoringStartCmd -->
<xsd:complexType name="monitoringStartCmd">
    <xsd:sequence>
        <xsd:element name="nodeID" type="xsd:long"
                    minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="parameter" type="msg:monitoringParameter"
                    minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="timeInterval" type="xsd:int"
                    minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- MonitoringStopCmd -->
<xsd:complexType name="monitoringStopCmd">
    <xsd:sequence>
        <xsd:element name="nodeID" type="xsd:long"
                    minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<!-- SensingValueRpt -->
<xsd:complexType name="sensingValueRpt">
    <xsd:sequence>
        <xsd:element name="commandID" type="xsd:int"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="nodeID" type="xsd:long"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="sendingTime" type="xsd:int"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="sensingValue" type="msg:sensingValue"
                    minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<!-- RunActuatorRpt -->
<xsd:complexType name="runActuatorRpt">
    <xsd:sequence>
        <xsd:element name="commandID" type="xsd:int"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="nodeID" type="xsd:long"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="sendigTime" type="xsd:int"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="actuation" type="msg:actuationType"
                    minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<!-- FinishRpt -->

```

```

<xsd:complexType name="finishRpt">
  <xsd:sequence>
    <xsd:element name="finishType" type="msg:finishType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="commandID" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- MonitoringRpt -->
<xsd:complexType name="monitoringRpt">
  <xsd:sequence>
    <xsd:element name="nodeID" type="xsd:long"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="sendingTime" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="monitoringValue" type="msg:monitoringValue"
      minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- ErrorRpt -->
<xsd:complexType name="errorRpt">
  <xsd:sequence>
    <xsd:element name="type" type="msg:errorType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="code" type="msg:errorCodeType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="timeStamp" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="errorInfo" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- UpdateRpt -->
<xsd:complexType name="updateRpt">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="addNode" type="msg:addNodeType"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="addTransducer"
        type="msg:addTransducerType"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="deleteNode"
        type="msg:deleteNodeType"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="deleteTransducer"
        type="msg:deleteTransducerType"
        minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="objectType" type="msg:updateObjectType"
    use="required"/>

```

```

    <xsd:attribute name="opType" type="msg:updateOpType" use="required"/>
</xsd:complexType>

```

```

<!-- 메시지 유형 -->

```

```

<xsd:simpleType name="messageType">
  <xsd:restriction base="xsd:int">
    <!-- ReqConnCtrl -->
    <xsd:enumeration value="1"/>
    <!-- ConnReqCtrl -->
    <xsd:enumeration value="2"/>
    <!-- ConnResCtrl -->
    <xsd:enumeration value="3"/>
    <!-- DisConnReqCtrl -->
    <xsd:enumeration value="15"/>
    <!-- AuthReqCtrl -->
    <xsd:enumeration value="256"/>
    <!-- AuthResCtrl -->
    <xsd:enumeration value="257"/>
    <!-- ChannelCheckCtrl -->
    <xsd:enumeration value="53248"/>
    <!-- ChannelConfirmCtrl -->
    <xsd:enumeration value="53249"/>
    <!-- NakChk -->
    <xsd:enumeration value="57343"/>

    <!-- NetworkInfoReq -->
    <xsd:enumeration value="4096"/>
    <!-- NetworkInfoRes -->
    <xsd:enumeration value="16384"/>
    <!-- BufferDataReq -->
    <xsd:enumeration value="4097"/>
    <!-- BufferDataRes -->
    <xsd:enumeration value="16385"/>
    <!-- CmdActionReq -->
    <xsd:enumeration value="8192"/>
    <!-- CmdActionRes -->
    <xsd:enumeration value="20480"/>
    <!-- UpdateCmdReq -->
    <xsd:enumeration value="8193"/>
    <!-- UpdateCmdRes -->
    <xsd:enumeration value="20481"/>
    <!-- ControlNetworkReq -->
    <xsd:enumeration value="12288"/>
    <!-- ControlNetworkRes -->
    <xsd:enumeration value="24576"/>
    <!-- ControlNodeReq -->
    <xsd:enumeration value="12289"/>
    <!-- ControlNodeRes -->
    <xsd:enumeration value="24577"/>

    <!-- InstantCmd -->
    <xsd:enumeration value="28672"/>
    <!-- ContinuousCmd -->

```

```

<xsd:enumeration value="28673"/>
<!-- InstantEventCmd -->
<xsd:enumeration value="28674"/>
<!-- ContinuousEventCmd -->
<!--
<xsd:enumeration value="28675"/>
-->
<!-- InstantAggCmd -->
<xsd:enumeration value="28676"/>
<!-- ContinuousAggCmd -->
<xsd:enumeration value="28677"/>
<!-- RunActuatorCmd -->
<xsd:enumeration value="31232"/>
<!-- MonitoringStartCmd -->
<xsd:enumeration value="32768"/>
<!-- MonitoringStopCmd -->
<xsd:enumeration value="32769"/>

<!-- SensingValueRpt -->
<xsd:enumeration value="40960"/>
<!-- RunActuatorRpt -->
<xsd:enumeration value="40961"/>
<!-- MonitoringRpt -->
<xsd:enumeration value="45056"/>
<!-- ErrorRpt -->
<xsd:enumeration value="49152"/>
<!-- UpdateRpt -->
<xsd:enumeration value="49153"/>
<!-- FinishRpt -->
<xsd:enumeration value="45055"/>
</xsd:restriction>
</xsd:simpleType>

<!-- 요청에 대한 응답 코드 -->
<xsd:simpleType name="retCodeType">
  <xsd:restriction base="xsd:int">
    <!-- 성공 -->
    <xsd:enumeration value="0"/>
    <!-- 동일한 어댑터가 이미 연결되어 있음. ConnResCtrl 에서 유효 -->
    <xsd:enumeration value="17"/>
    <!-- 등록되지 않은 센서네트워크 식별자. ConnResCtrl 에서 유효 -->
    <xsd:enumeration value="18"/>
    <!-- SNCIC 의 연결 자원 부족. ConnResCtrl 에서 유효 -->
    <xsd:enumeration value="19"/>

    <!-- 연결설정 요청 과정이 이루어지지 않았음. AuthResCtrl 에서 유효
-->

    <xsd:enumeration value="33"/>
    <!-- 유효하지 않은 채널 식별자. AuthResCtrl 에서 유효 -->
    <xsd:enumeration value="34"/>
    <!-- 유효하지 않은 credential. AuthResCtrl 에서 유효 -->
    <xsd:enumeration value="35"/>

    <!-- 실행되고 있지 않은 명령. -->

```

```

<!-- CmdActionRes, UpdateCmdRes 에서 유효 -->
<xsd:enumeration value="81"/>
<!-- 중단되지 않은 명령. CmdActionRes 에서 유효 -->
<xsd:enumeration value="82"/>
<!-- 일부에 대한 제어만 성공. -->
<!-- CmdActionRes, ControlNetworkRes 에서 유효 -->
<xsd:enumeration value="83"/>

<!-- 처리대상이 없는 명령 -->
<!-- 유효한 명령이지만, 해당 명령을 처리할 대상이 존재하지 않는
경우 -->
<xsd:enumeration value="253"/>
<!-- 지원하지 않는 명령. 유효한 명령이지만, -->
<!-- 명령 내에 포함된 정보의 의미를 모두 알 수 없음 -->
<xsd:enumeration value="254"/>
<!-- 정의되지 않은 에러. 명령을 분석하여 처리를 시도하였지만, -->
<!-- 정의되지 않은 원인으로 인하여 처리에 실패하였음 -->
<xsd:enumeration value="255"/>
</xsd:restriction>
</xsd:simpleType>

<!-- 보안 모드 -->
<xsd:simpleType name="encryptModeType">
  <xsd:restriction base="xsd:int">
    <!-- 기밀성 비적용 -->
    <xsd:enumeration value="0"/>
    <!-- 메시지 보안 -->
    <xsd:enumeration value="1"/>
    <!-- 통신보안 -->
    <xsd:enumeration value="2"/>
    <!-- 메시지보안 + 통신보안 -->
    <xsd:enumeration value="3"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- 통신채널 유형 -->
<xsd:simpleType name="channelType">
  <xsd:restriction base="xsd:int">
    <!-- 제어채널 -->
    <xsd:enumeration value="2"/>
    <!-- 명령채널 -->
    <xsd:enumeration value="4"/>
    <!-- 제어채널이자 명령채널 -->
    <xsd:enumeration value="6"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- 네트워크 동작 모드 -->
<xsd:simpleType name="networkModeType">
  <xsd:restriction base="xsd:int">
    <!-- PUSH -->
    <xsd:enumeration value="0"/>
    <!-- PULL -->
    <xsd:enumeration value="1"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

    </xsd:restriction>
</xsd:simpleType>

<!-- 노드의 유형 -->
<xsd:simpleType name="nodeType">
    <xsd:restriction base="xsd:int">
        <!-- 센서노드 -->
        <xsd:enumeration value="0"/>
        <!-- RFID 리더 -->
        <xsd:enumeration value="1"/>
        <!-- IP-USN 노드 -->
        <xsd:enumeration value="2"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 배터리 정보 표현 유형 -->
<xsd:simpleType name="batteryType">
    <xsd:restriction base="xsd:int">
        <!-- 전원정보를 지원하지 않음 -->
        <xsd:enumeration value="0"/>
        <!-- 영구전원이 공급됨 -->
        <xsd:enumeration value="1"/>
        <!-- 현재의 전압값을 실수로 표현 -->
        <xsd:enumeration value="2"/>
        <!-- 남아있는 전원을 백분율(실수)로 표현 -->
        <xsd:enumeration value="3"/>
        <!-- 향후 동작 가능한 시간(단위:초, 정수)을 표현 -->
        <xsd:enumeration value="4"/>
        <!-- 남아있는 전류의 양(단위:mA, 정수)을 표현 -->
        <xsd:enumeration value="5"/>
        <!-- 전원 부족 여부만을 표현 -->
        <xsd:enumeration value="6"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 트랜스 듀서 유형 -->
<xsd:simpleType name="transducerType">
    <xsd:restriction base="xsd:int">
        <!-- 센서 -->
        <xsd:enumeration value="1"/>
        <!-- 구동기 -->
        <xsd:enumeration value="2"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 구동기 동작 유형 -->
<xsd:simpleType name="actionType">
    <xsd:restriction base="xsd:int">
        <!-- OnOff 연산이 가능한 구동기 -->
        <xsd:enumeration value="0"/>
        <!-- 아날로그 값에 기반하여 동작하는 구동기 -->
        <xsd:enumeration value="1"/>
    </xsd:restriction>
</xsd:simpleType>

```

```

<!-- 명령 제어 유형 -->
<xsd:simpleType name="commandActionType">
  <xsd:restriction base="xsd:int">
    <!-- 중지 -->
    <xsd:enumeration value="0"/>
    <!-- 일시중지 -->
    <xsd:enumeration value="1"/>
    <!-- 재시작 -->
    <xsd:enumeration value="2"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

<!-- 집계함수 유형 -->
<xsd:simpleType name="functionType">
  <xsd:restriction base="xsd:int">
    <!-- 최소값 -->
    <xsd:enumeration value="0"/>
    <!-- 최대값 -->
    <xsd:enumeration value="1"/>
    <!-- 합계 -->
    <xsd:enumeration value="2"/>
    <!-- 평균 -->
    <xsd:enumeration value="3"/>
    <!-- 개수 -->
    <xsd:enumeration value="4"/>
    <!-- 의미없음 -->
    <xsd:enumeration value="255"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

<!-- 논리연산자 -->
<xsd:simpleType name="opLogicalType">
  <xsd:restriction base="xsd:int">
    <!-- AND -->
    <xsd:enumeration value="1"/>
    <!-- OR -->
    <xsd:enumeration value="2"/>
    <!-- NOOP -->
    <xsd:enumeration value="0"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

<!-- 관계연산자 -->
<xsd:simpleType name="opRelationalType">
  <xsd:restriction base="xsd:int">
    <!-- not equal -->
    <xsd:enumeration value="0"/>
    <!-- equal -->
    <xsd:enumeration value="1"/>
    <!-- lessThan -->
    <xsd:enumeration value="2"/>
    <!-- lessThanEqual -->
    <xsd:enumeration value="3"/>
  </xsd:restriction>

```

```

        <!-- greater -->
        <xsd:enumeration value="4"/>
        <!-- greaterThanEqual -->
        <xsd:enumeration value="5"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 네트워크 연산 유형 -->
<xsd:simpleType name="networkControlType">
    <xsd:restriction base="xsd:int">
        <!-- 리셋 -->
        <xsd:enumeration value="0"/>
        <!-- 채널변경 -->
        <xsd:enumeration value="1"/>
        <!-- 모드변경 -->
        <xsd:enumeration value="2"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 노드 연산 유형 -->
<xsd:simpleType name="nodeControlType">
    <xsd:restriction base="xsd:int">
        <!-- 리셋 -->
        <xsd:enumeration value="0"/>
        <!-- 켜기/끄기 -->
        <xsd:enumeration value="1"/>
        <!-- 부모노드 식별자 -->
        <xsd:enumeration value="2"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 모니터링 파라미터 -->
<xsd:simpleType name="monitoringParameter">
    <xsd:restriction base="xsd:int">
        <!-- 노드상태 -->
        <xsd:enumeration value="1"/>
        <!-- 위치 -->
        <xsd:enumeration value="2"/>
        <!-- 부모노드 식별자 -->
        <xsd:enumeration value="4"/>
        <!-- 노드간 연결강도 -->
        <xsd:enumeration value="8"/>
        <!-- 잔존배터리량 -->
        <xsd:enumeration value="16"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 완료보고 유형 -->
<xsd:simpleType name="finishType">
    <xsd:restriction base="xsd:int">
        <!-- 주기의 완료 -->
        <xsd:enumeration value="0"/>
        <!-- 명령의 완료 -->
        <xsd:enumeration value="1"/>

```

```

        </xsd:restriction>
    </xsd:simpleType>

    <!-- 에러유형 -->
    <xsd:simpleType name="errorType">
        <xsd:restriction base="xsd:int">
            <!-- 센싱/구동기 명령 -->
            <xsd:enumeration value="0"/>
            <!-- 센서네트워크 -->
            <xsd:enumeration value="1"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!-- 에러코드 -->
    <xsd:simpleType name="errorCodeType">
        <xsd:restriction base="xsd:int">
            <!-- 명령의 개수를 초과하여 더이상 수행할 수 없음 -->
            <xsd:enumeration value="0"/>
            <!-- 명령 수행이 중단되었거나 센서네트워크 동작이 중단되었음 -->
            <xsd:enumeration value="1"/>
            <!-- 명령은 분석되었거나, 내부 요소들을 지원하지 않음 -->
            <xsd:enumeration value="254"/>
            <!-- 명령 처리를 시도하였으나 처리에 실패하였음 -->
            <!-- 혹은 센서네트워크에서 알 수 없는 문제가 발생하였음 -->
            <xsd:enumeration value="255"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!-- 센싱데이터 유형 -->
    <xsd:simpleType name="sensingTypeID">
        <xsd:restriction base="xsd:int">
            <xsd:enumeration value="5889"/>
            <xsd:enumeration value="5890"/>
            <xsd:enumeration value="5891"/>
            <xsd:enumeration value="5892"/>
            <xsd:enumeration value="5893"/>
            <xsd:enumeration value="5894"/>
            <xsd:enumeration value="5895"/>
            <xsd:enumeration value="5896"/>
            <xsd:enumeration value="5897"/>
            <xsd:enumeration value="5898"/>
            <xsd:enumeration value="5899"/>
            <xsd:enumeration value="5900"/>
            <xsd:enumeration value="5901"/>
            <xsd:enumeration value="5902"/>
            <xsd:enumeration value="5903"/>
            <xsd:enumeration value="5904"/>
            <xsd:enumeration value="5905"/>
            <xsd:enumeration value="5906"/>
            <xsd:enumeration value="5907"/>
            <xsd:enumeration value="5908"/>
            <xsd:enumeration value="5909"/>
            <xsd:enumeration value="5910"/>
        </xsd:restriction>
    </xsd:simpleType>

```

```

        <xsd:enumeration value="5911"/>
        <xsd:enumeration value="5912"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 명령 갱신 유형 -->
<xsd:simpleType name="updateCmdType">
    <xsd:restriction base="xsd:int">
        <!-- 센싱 조건 -->
        <xsd:enumeration value="1"/>
        <!-- 시간 -->
        <xsd:enumeration value="2"/>
        <!-- 함수 -->
        <xsd:enumeration value="4"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 갱신 대상 정보 유형 -->
<xsd:simpleType name="updateObjectType">
    <xsd:restriction base="xsd:int">
        <!-- 노드 -->
        <xsd:enumeration value="0"/>
        <!-- 트랜스듀서 -->
        <xsd:enumeration value="1"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 추가, 삭제 -->
<xsd:simpleType name="updateOpType">
    <xsd:restriction base="xsd:int">
        <!-- 추가 -->
        <xsd:enumeration value="0"/>
        <!-- 삭제 -->
        <xsd:enumeration value="1"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- 노드정보 -->
<xsd:complexType name="addNodeType">
    <xsd:sequence>
        <xsd:element name="nodeType" type="msg:nodeType"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="nodeID" type="xsd:long"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="batteryType" type="msg:batteryType"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="hwSpecID" type="xsd:long"
            minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- 노드에 대한 정적 메타 정보 유형 -->
<xsd:complexType name="nodeMetaType">

```

```

<xsd:complexContent>
  <xsd:extension base="msg:addNodeType">
    <xsd:sequence>
      <xsd:element name="transducerMeta"
        type="msg:transducerMetaType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- 트랜스듀서에 대한 정적 메타 정보 유형 -->
<xsd:complexType name="transducerMetaType">
  <xsd:sequence>
    <xsd:element name="transducerType" type="msg:transducerType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="transducerID" type="xsd:long"
      minOccurs="1" maxOccurs="1"/>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="sensingTypeID"
type="msg:sensingTypeID"
      minOccurs="1" maxOccurs="1"/>
      <xsd:element name="action" type="msg:actionType"
      minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
    <xsd:element name="hwSpecID" type="xsd:long"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- 트랜스듀서에 대한 정보 유형 -->
<xsd:complexType name="addTransducerType">
  <xsd:complexContent>
    <xsd:extension base="msg:transducerMetaType">
      <xsd:sequence>
        <xsd:element name="nodeID" type="xsd:long"
          minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- 명령에 포함되는 조건 -->
<xsd:complexType name="condition">
  <xsd:sequence>
    <xsd:element name="sensingTypeID" type="msg:sensingTypeID"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="opLogical" type="msg:opLogicalType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="opRelational" type="msg:opRelationalType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="compareValue" type="msg:valueType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>

```

```

</xsd:complexType>

<!-- 명령에 포함되는 시간 -->
<xsd:complexType name="continuousTime">
  <xsd:sequence>
    <xsd:element name="timeInterval" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="lifeTime" type="xsd:int"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- 집계정보 -->
<xsd:complexType name="aggregationInfo">
  <xsd:sequence>
    <xsd:element name="functionType" type="msg:functionType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="sensingTypeID" type="msg:sensingTypeID"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- 센싱값 -->
<xsd:complexType name="valueType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="byte" type="xsd:base64Binary"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="short" type="xsd:short"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="int" type="xsd:int"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="float" type="xsd:float"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="string" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="position" type="msg:positionType"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="long" type="xsd:long"
        minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<!-- 위치유형 -->
<xsd:complexType name="positionType">
  <xsd:sequence>
    <xsd:element name="x" type="xsd:float"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="y" type="xsd:float"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="z" type="xsd:float"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>

```

```

        </xsd:sequence>
    </xsd:complexType>

    <!-- 센싱값 -->
    <xsd:complexType name="sensingValue">
        <xsd:sequence>
            <xsd:element name="functionType" type="msg:functionType"
                minOccurs="1" maxOccurs="1"/>
            <xsd:element name="sensingTypeID" type="msg:sensingTypeID"
                minOccurs="1" maxOccurs="1"/>
            <xsd:element name="value" type="msg:valueType"
                minOccurs="1" maxOccurs="1"/>
            <xsd:element name="timeStamp" type="xsd:int"
                minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <!-- 모니터링값 -->
    <xsd:complexType name="monitoringValue">
        <xsd:sequence>
            <xsd:element name="parameter" type="msg:monitoringParameter"
                minOccurs="1" maxOccurs="1"/>
            <xsd:element name="value" type="msg:valueType"
                minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <!-- 구동기 연산 형태-->
    <xsd:complexType name="actuationType">
        <xsd:sequence>
            <xsd:element name="actuatorID" type="xsd:long"
                minOccurs="1" maxOccurs="1"/>
            <xsd:element name="actionValueType" type="msg:actionValueType"
                minOccurs="1" maxOccurs="1"/>
            <xsd:element name="actionValue" type="xsd:float"
                minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <!-- 구동기 연산 동작 값 (actionValue 의 값의 유형) -->
    <xsd:simpleType name="actionValueType">
        <xsd:restriction base="xsd:int">
            <!-- On/Off -->
            <xsd:enumeration value="0"/>
            <!-- Analog -->
            <xsd:enumeration value="1"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!-- 노드삭제에 필요한 유형 -->
    <xsd:complexType name="deleteNodeType">
        <xsd:sequence>
            <xsd:element name="nodeID" type="xsd:long"
                minOccurs="1" maxOccurs="1"/>

```

```
</xsd:sequence>
</xsd:complexType>

<!-- 트랜스듀서 삭제에 필요한 유형 -->
<xsd:complexType name="deleteTransducerType">
  <xsd:sequence>
    <xsd:element name="transducerID" type="xsd:long"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="nodeID" type="xsd:long"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```



## 7 표현

메시지 내부에 표현되는 다양한 데이터는 다음과 같이 표현된다.

### 7.1 값의 표현

값은 빅 엔디안(Big Endian)으로 표현한다. 예를 들어 SensingValueRpt 메시지 코드 값은 다음과 같이 표현된다. 실수를 표현하는 경우 이는 IEEE754를 따른다.

예) SensingValueRpt: 0xA000

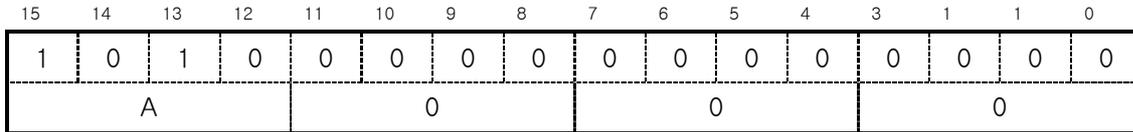


표 5- 41 값의 표현 방법

## 7.2 날짜와 시간

### 7.2.1 TimeStamp

날짜와 시간은 총 4바이트로 표현되며, 1970년 1월 1일 자정 이후로 흐른 시간을 초단위로 표현한다. 예를 들어 자바에서 현재시간을 표현하고자 한다면 System.currentTimeMillis() / 1000으로 표현할 수 있다.

### 7.2.2 LifeTime

메시지를 수신한 시점부터 기술된 시간 동안 해당 명령이 수행되어야 함을 표현한다. 시간단위는 초이다.

## 7.3 센싱값

센싱값을 표현하기 위해서는 메시지 내에 해당 값의 유형에 대한 식별자를 표현하여야 한다.

## 부록 : 센서데이터 유형 정의(예제)

센싱값 유형 정의표는 센서네트워크공통인터페이스를 이용하여 통신하고자 할 때 이용되는 정의표로서, 도메인별로 정의되어 운용될 수 있다. 본 부록은 하나의 예제로서 본 표준안에 포함되었다.

센서네트워크에서 센싱된 값들은 어댑터에서 공통메시지로 표현되어 호스트로 전달된다. 센싱값은 측정된 데이터 값과 그 데이터를 표현하기 위한 여러가지 부가정보로 기술될 수 있다. 이러한 부가정보들을 센싱값 유형이라고 한다.

식별자	이름	표현단위	유형
0x1701 (5889) <sub>10</sub>	온도	°C	고정길이 4바이트, 실수
0x1702 (5890) <sub>10</sub>	염분	%	고정길이 4바이트, 실수
0x1703 (5891) <sub>10</sub>	용존산소량	%	고정길이 4바이트, 실수
0x1704 (5892) <sub>10</sub>	전원잔량	V	고정길이 4바이트, 실수
0x1705 (5893) <sub>10</sub>	전원잔량	%	고정길이 4바이트, 실수
0x1706 (5894) <sub>10</sub>	맥박	bpm	고정길이 4바이트, 정수
0x1707 (5895) <sub>10</sub>	운동량	milli-g	고정길이 2바이트, 정수
0x1708 (5896) <sub>10</sub>	문자열	없음	가변길이 첫번째 1바이트 : 문자열의 길이 n 바이트 : 문자열
0x1709 (5897) <sub>10</sub>	압력	gf/cm <sup>2</sup>	고정길이 2바이트, 정수
0x170A (5898) <sub>10</sub>	높이	cm	고정길이 2바이트, 정수
0x170B (5899) <sub>10</sub>	CO	ppm	고정길이 4바이트, 실수
0x170C (5900) <sub>10</sub>	조도(TSR)	lux	고정길이 2바이트, 정수
0x170D (5901) <sub>10</sub>	MIC	dB	고정길이 2바이트, 정수
0x170E (5902) <sub>10</sub>	습도	%	고정길이 4바이트, 정수

정보통신단체표준(국문표준)

0x170F (5903) <sub>10</sub>	배기압	mmH <sub>2</sub> O	고정길이 2바이트, 정수
0x1710 (5904) <sub>10</sub>	위치	없음	고정길이 12바이트, 실수 X : 4바이트 실수 Y : 4바이트 실수 Z : 4바이트 실수
0x1711 (5905) <sub>10</sub>	RFID태그	없음	가변길이 첫번째 1바이트 : 코드의 길이 n 바이트 : 태그코드
0x1712 (5906) <sub>10</sub>	바이너리 데이터	없음	가변길이 첫번째 1바이트 : 바이트 길이 n 바이트 : 데이터
0x1713 (5907) <sub>10</sub>	노드식별자	없음	고정길이 8바이트, 정수
0x1714 (5908) <sub>10</sub>	심전도	mV	고정길이 2바이트, 정수
0x1715 (5909) <sub>10</sub>	가스농도	ppb	고정길이 4바이트, 실수
0x1716 (5910) <sub>10</sub>	미세먼지 농도	μg/m <sup>3</sup>	고정길이 4바이트, 실수
0x1717 (5911) <sub>10</sub>	풍향	도	고정길이 2바이트, 정수
0x1718 (5912) <sub>10</sub>	풍속	m/s	고정길이 4바이트, 실수
0x1719 (5913) <sub>10</sub>	CO <sub>2</sub>	ppm	고정길이 4바이트, 실수
0x171A (5914) <sub>10</sub>	O <sub>3</sub>	ppm	고정길이 4바이트, 실수
0x171B (5915) <sub>10</sub>	NO <sub>2</sub>	ppm	고정길이 4바이트, 실수
0x171C (5916) <sub>10</sub>	SO <sub>2</sub>	ppm	고정길이 4바이트, 실수
0x171D (5917) <sub>10</sub>	체온	°C	고정길이 4바이트, 실수
0x171E	조도(PAR)	lux	고정길이

(5918)10			2바이트, 정수
----------	--	--	----------

표 5- 42 센싱값 유형



표준작성 공헌자

표준 번호 : TTAS.KO-06.0169/R1

이 표준의 개정 및 발간을 위해 아래와 같이 여러분들이 공헌하였습니다.

구분	성명	위원회 및 직위	연락처	소속사
과제 제안	김말희	네트워크연동실무반 위원	042-860-1590 mariekim@etri.re.kr	ETRI
표준 초안 제출	김말희	네트워크연동실무반 위원	042-860-1590 mariekim@etri.re.kr	ETRI
	이경우	이사	02-501-3322 thiscase@galimit.com	가림정 보기술
표준 초안 검토 및 작성	표철식	RFID/USN 프로젝트그룹 의장	042-860-4929 cspyo@etri.re.kr	ETRI
	외 프로젝트그룹 위원			
표준안 심의	채종석	전파통신기술위원회 의장	042-860-1600 jschae@etri.re.kr	ETRI
	외 기술위원회 위원			
사무국 담당	김대중	-	031-724-0090 kdj@tta.or.kr	TTA
	김수학	RFID/USN 프로젝트그룹 간사	070-7780-0096 soohagi@tta.or.kr	TTA

---

정보통신단체표준(국문표준)

센서 네트워크 공통 인터페이스  
(The Standard Interface for Heterogeneous Sensor Networks)

발행인 : 한국정보통신기술협회 회장

발행처 : 한국정보통신기술협회

463-824, 경기도 성남시 분당구 서현동 267-2

Tel : 031-724-0114, Fax : 031-724-0019

발행일 : 2009.6.18

---