UM3 Protocol Recommendation (2012.05)

KT Corporation

UXA-1017

blank

UM3 Protocol Recommendation (2012.05)

UM3 Protocol Recommendation (2012.05)

서울특별시 서초구 우면동 17 ㈜케이티 중앙연구소

본 권고안의 수정 및 개정에 관한 업무는 ㈜케이티 중앙연구소가 담당하고 있으며 그 연락처는 02-526-5350 혹은 wunbae.jeon@kt.com 입니다.

본 권고안의 내용을 기반으로 이루어지는 공동사업, 사업제휴, 제안 등의 업무는 또한 ㈜케이티 중앙연구소의 주관으로 각 유관 사업부서를 확인해야 합니다.

본 권고안의 정식 명칭은 'UM3 프로토콜 권고안 (2012.05)'이며 영문명칭은 'UM3 Protocol Recommendation (2012.05)' 입니다. 본 권고안의 발행번호는 정식 명칭 중 '2012.05' 입니다. 발행번호는 UM3 프로토콜 버전번호와 동일하게 사용됩니다. 본 권고안에 대한 수정본 및 개정본은 연중 수시로 발간됩니다.

본 권고안의 작성, 편집, 확인 및 발행인 정보는 **작성/편집** 팀장 전운배, **확인** 상무보 정학진 **발행** 중앙연구소장 상무 정한욱 ㈜케이티 종합기술원 중앙연구소 입니다.

The translated specification of the document in English is described pararelly from page 23.

♠♠

blank

차 례

그림 목차

표 목차

1. UM3 프로토콜 권고안의 제정 목적 및 적용 범위

1.1. 목적

UM3 프로토콜 권고안은 M2M 서비스 환경에서 원격 센서 및 원격 제어장치로부터 데이터를 수집하거나 원격제어를 위한 응용계층의 데이터 송수신 표준을 마련하는데 그 목적을 두고 있습니다. UM3 프로토콜은 '이해하기 쉽고 적용이 간편한 표준'이라는 목표와 '국제표준의 형식을 갖춘 표준', 그리고 '확장성과 범용성을 갖춘 표준'이라는 주요 목표를 설정하고 개발된 프로토콜 입니다.

1.2. 적용 범위

UM3 프로토콜은 단순한 센서 디바이스와 컴퓨팅 장치 사이의 데이터 송수신만을 위한 표준이 아니라, 센서를 활용하는 서비스에 대한 운용관리 프로세스 및 본연의 서비스 프로세스의 지원까지를 그 적용 범위로 합니다. 즉, UM3 프로토콜은 서비스 프로세스의 실행을 위해 필요한 데이터의 송수신, 해당 서비스 시스템을 최상의 상태로 유지할 수 있는 서비스 운용관리 프로세스의 실행까지 지원하는 표준 프로토콜입니다.

1.3. UM3 프로토콜과 관련된 지적재산권의 적용 범위 및 사용

본 권고안에 정의된 UM3 프로토콜에 대한 지적재산권은 ㈜케이티가 소유하고 있습니다. 본 권고안에 정의된 UM3 프로토콜을 상용목적으로 사용하기 위해서는 ㈜케이티와의 협약, 계약, 협의 등에 의한 사용권의 허락을 필요로 합니다.

상기 ㈜케이티와의 협약, 계약, 협의 등은 연락처를 참고하시기 바랍니다.

1.4. UM3 프로토콜의 수정 및 개정

본 권고안에 대한 수정 및 개정은 ㈜케이티 중앙연구소가 담당하고 있으며 향후 UM3 표준화 위원회 (가칭)의 구성 및 개정 작업 등을 진행할 계획입니다. 또한 UM3 프로토콜은 ㈜케이티의 사내표준으로 제정될 예정이며 ETSI, OneM2M 등의 국제표준화 기구 및 TTA 를 통한 국내 표준화를 위한 표준으로 반영하기 위한 노력이 계속되고 있습니다.

상기 UM3 표준화 위원회 (가칭)와 관련된 사항은 연락처를 참고하시기 바랍니다.

1.5. 본 권고안을 참조하기 위한 사전 요건 및 대상인력의 업무

본 권고안은 '이해하기 쉽고, 구현이 용이하고 신속하며, 유연성과 확장성을 갖고 있으며, 국제표준의 형식과 틀을 부여'하는 것을 목표로 제정되었습니다. 그러나, 본 권고안을 참조하기 위해서는 다음과 같은 몇 가지 사전지식이 필요합니다.

- 객체지향개발 방법론의 이해 (Understanding of Object-oriented methodology)
- C++, C#, Java 와 같은 객체지향 프로그래밍언어와 C 프로그래밍 언어를 이용한 일부 개발 경험 (Experiences in C++, C#, Java and C programming language)
- TCP/IP 통신 프로토콜의 구현을 위한 소켓 프로그래밍 경험 (Experiences in TCP/IP socket programming)
- XML (Extensible Markup Language) 혹은 HTML (Hyper Text Markup Language) 문서의 작성 경험

상기 객체지향개발 방법론은 본 권고안이 정의하는 정보모델과 통신서비스모델에 대한 이해를 위해 필요합니다. 객체지향 프로그래밍언어와 C 프로그래밍언어는 본 권고안이 정의하는 데이터 통신 방식을 이해하고 이를 구현하기 위해 필요합니다. 소켓 프로그래밍 경험은 본 권고안이 정의하는 통신서비스모델이 TCP/IP 응용계층 (application layer) 을 대상으로 하고 있기 때문입니다.

또한 상기 XML 혹은 HTML 문서의 작성 경험은 본 권고안이 정의하는 XML 기반의 데이터 송수신을 위한 데이터의 구성을 파악하는데 도움이 되기 때문입니다.

따라서, 본 권고안이 대상으로 하는 독자는 다음과 같은 업무를 담당하고 계신 분들입니다.

- 컴퓨터 하드웨어, 센서, 제어장치 등의 하드웨어 개발자
- 게이트웨이 (Gateway), RTU (Remote terminal unit) 등 특수목적의 컴퓨터의 구동을 담당하는 펌웨어 (firmware) 등의 임베디드 소프트웨어 (embedded software) 개발자
- 서버 등의 컴퓨터에서 비즈니스 로직 혹은 서비스 로직의 구현을 담당하는 응용 애플리케이션 개발자
- 상기 시스템의 개발을 총괄하는 PM (Project manager), 설계자 혹은 아키텍트 (architect)

마케팅 혹은 사업기획, 사업경영 등을 담당하고 계신 분들은 본 권고안의 내용을 직접 활용하기 보다는 소프트웨어나 혹은 임베디드 장치의 개발을 담당하는 개발인력을 통해 내용을 참조해야 합니다.

또한 본 권고안이 정의하는 내용들 즉, UM3 프로토콜을 상품화하고 이를 기반으로 마케팅 활동을 진행하고자 할 경우에는 본 권고안을 설계하고 그 내용을 기술한 ㈜케이티 중앙연구소의 적절한 업무지원을 요청하도록 권고합니다.

2. 약어의 정의

UM3	Universal Management, Maintenance, Monitoring
M2M	Machine-to-machine
USN	Ubiquitous Sensor Network
MOS	Monitoring, Management and Maintenance Operation Service
ITIL	IT Infrastructure Library
FCAPS	Fault, Configuration, Account, Performance and Security management
RDN	Relative distinguished name
ASN.1	Abstract syntax notation one
BER	Basic encoding rule
SER	Simple encoding rule
ITU	International telecommunication unit
APDU	Application protocol data unit
SDU	Service data unit
SNMP	Simple network management protocol
OLVD	Function overloaded
XML	Extensible markup language
OSI	Open system interconnection
CMIP	Common management information protocol
OS	Operating system
CPU	Central processing unit

3. UM3 프로토콜 권고안의 구성

UM3 프로토콜 권고안은 정보모델의 정의, 통신서비스모델의 정의 및 인코딩 룰의 정의로 구성되어 있습니다. UM3 프로토콜 권고안이 정의하는 정보모델은 서비스관리정보모델의 정의와 응용서비스 정보모델의 정의로 구성되어 있습니다.

1) 정보모델의 정의 (Information model)
   가) 서비스관리 정보모델의 정의 (Service management information model)
   나) 응용서비스 정보모델의 정의 (Application service information model)
2) 통신서비스모델의 정의 (Communication service model)
3) 인코딩 룰의 정의 (Encoding rule)

서비스관리 정보모델은 ITIL이 정의하고 있는 각 운용관리 프로세스를 위한 M2M 서비스 관점의 주요 정보모델을 정의하고 있습니다. 응용서비스 정보모델은 M2M 서비스를 활용하는 여러 가지 비즈니스 및 서비스를 위해 필요한 정보모델을 정의합니다.

통신서비스모델은 상기 서비스관리 정보모델 및 응용서비스 정보모델을 이용한 응용계층의 데이터 송수신 모델을 정의합니다.

인코딩 룰의 정의는 상기 정보모델이 갖고 있는 정보를 통신 네트워크를 통해 송신하거나 수신할 경우 해당 APDU의 비트의 나열을 위한 규칙을 정의합니다. 또한 XML 기반의 데이터 교환을 위한 XML schema 도 함께 정의합니다.

4. 응용계층 통신과 원격관리 시스템의 개념적 구조

4.1. 응용계층간의 통신

일반적으로 개방형 컴퓨터 네트워크라 함은 두 대 이상의 컴퓨터가 미리 약속한 형태와 절차에 따라 데이터를 주고 받는 프로토콜을 갖고 있음을 뜻합니다. 이 때 사용하는 프로토콜은 전송계층이라 표현할 수 있는 OSI 7 layer 의 응용계층 (application layer) 를 제외한 하위 계층간의 데이터 통신과 응용계층의 데이터 통신을 모두 포함하게 됩니다.

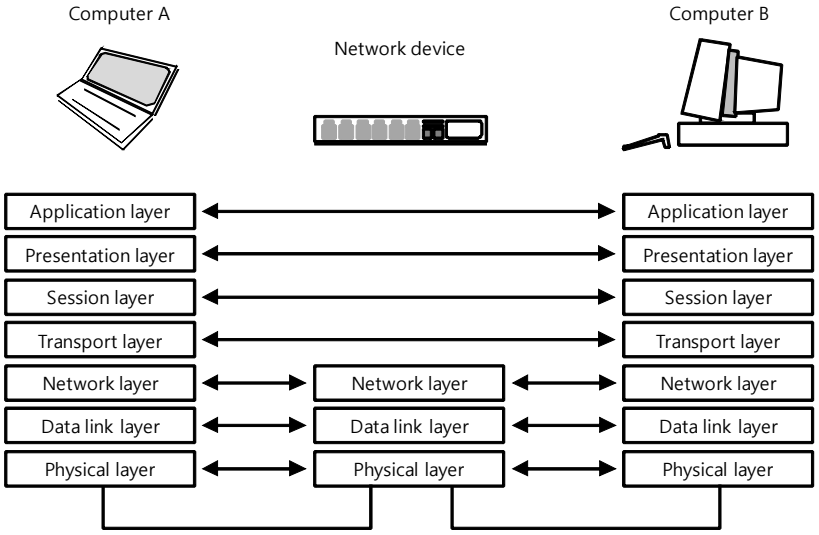


그림 1-OSI 7 layer

그림 1 은 OSI 7 계층의 구조를 나타내고 있습니다. OSI 7 계층은 개념적인 구조로 볼 수 있으며 컴퓨터간의 통신을 위한 프로토콜은 Ethernet 기반의 TCP/IP 프로토콜을 사용하는 경우가 대부분입니다. 본 권고안이 정의하는 UM3 프로토콜은 TCP/IP 프로토콜의 응용계층 간의 데이터 통신을 위한 데이터의 형식과 그 절차를 정의하고 있습니다. OSI 7 계층의 구조를 Ethernet 기반의 TCP/IP 통신 프로토콜 구조와 비교하면 그림 2 와 같습니다.

즉, 그림 2 에서 보는 바와 같이 UM3 프로토콜은 우리가 흔히 사용하는 Telnet, Ftp 등의 TCP/IP 응용계층 프로토콜과 동일한 수준의 프로토콜입니다.

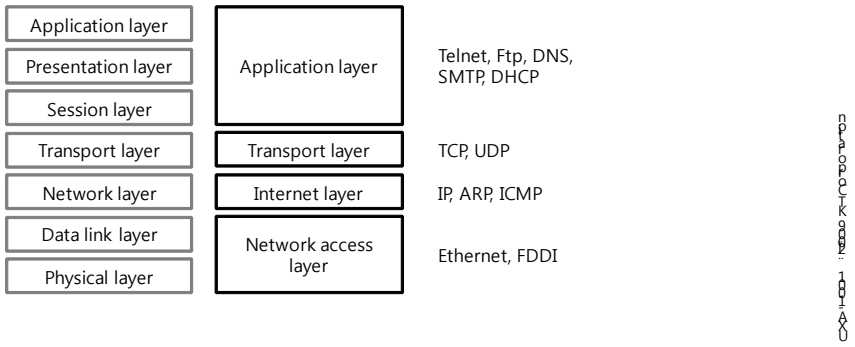


그림 2-OSI 7 layer 와 TCP/IP layer 의 비교

본 권고안은 이와 같은 응용계층간의 데이터 통신을 위한 0 과 1 의 조합으로 이루어진 비트의 나열에 관한 규칙, 데이터의 형식, 데이터를 주고 받는 절차 등을 정의하며 이를 UM3 프로토콜이라는 이름으로 정의합니다.

4.2. 원격관리 시스템의 매니저와 에이전트

UM3 프로토콜을 사용할 수 있는 응용서비스 및 서비스 관리 분야에 있어서, 원격의 센서로부터 데이터를 수집하거나 혹은 원격의 제어장치를 제어하기 위해 데이터를 송신하는 경우, 각 서비스 시스템은 아래와 같은 두 가지의 개념적 소프트웨어 혹은 하드웨어 모듈을 내장할 수 있습니다.

- 매니저 (Manager)
- 에이전트 (Agent)

매니저는 에이전트에게 데이터의 송신을 명령할 수 있으며, 매니저는 에이전트에게 데이터의 수신을 명령할 수 있습니다.

에이전트는 매니저에게 데이터를 송신할 수 있으며, 매니저의 데이터를 수신하여 자신이 담당하고 있는 서비스 구성 자원의 데이터를 수정할 수 있습니다. 즉, 에이전트는 자신이 담당하는 구성자원의 상

태와 관련된 데이터를 매니저에게 송신하며, 매니저로부터 수신되는 데이터에 따라 자신이 관리하는 구성자원의 상태를 변경합니다. 단, 매니저와 에이전트는 개념적인 구조와 역할이며 경우에 따라 매니저와 에이전트를 모두 하나의 장치에 내장하여 그 역할을 함께 동작하도록 하는 경우도 가능합니다.

4.3. 정보모델과 매니저 및 에이전트의 관계

그림 3에서 보는 바와 같이 매니저는 에이전트를 통해 센서 및 원격제어장치들의 데이터를 수집합니다. 이 관계는 매니저가 생성하고 관리하는 정보모델에 있어서, 이를 구성하는 오브젝트의 값을 실제로 동작하고 있는 센서나 원격제어장치의 상태 값으로 동기화하는 것과 동일한 개념입니다.

원격제어장치의 제어를 위해 에이전트로 데이터를 송신하는 경우에도, 이는 매니저가 생성하고 관리하는 오브젝트가 갖고 있는 값과 동일한 값으로 에이전트가 관리하는 원격제어장치의 값을 동기화시키는 과정으로 볼 수 있습니다.

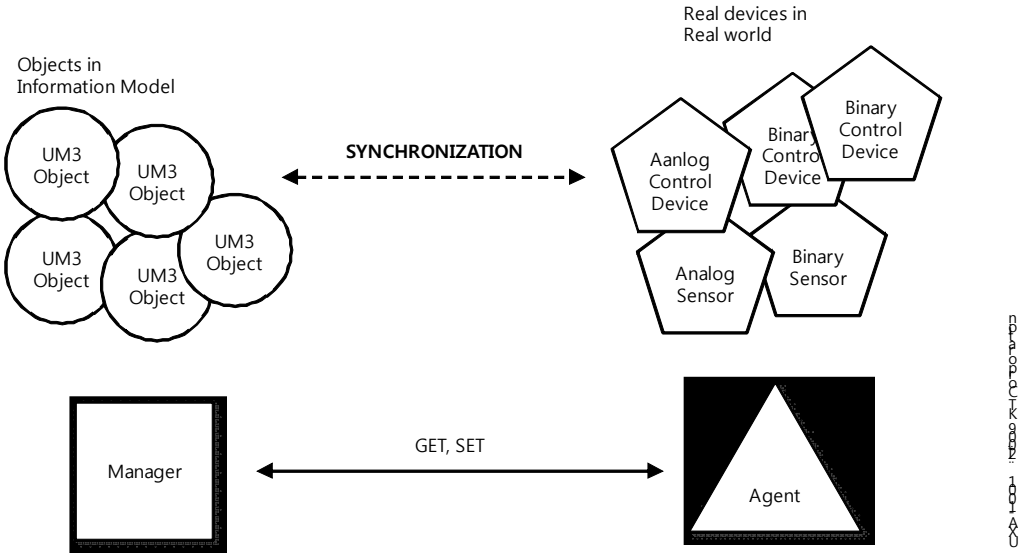


그림 3-UM3 프로토콜을 사용하는 서비스 시스템의 개념도

이상과 같은 정보모델과 장치 사이의 정보를 동기화시키기 위해서는 매니저뿐만 아니라 에이전트도 UM3 프로토콜 권고안이 정의하는 정보모델을 생성하고 관리할 수 있어야 합니다.

5. UM3 프로토콜의 구조와 활용방법의 이해

5.1. UM3 정보모델과 객체지향 방법론

UM3 프로토콜은 객체지향 방법론 (Object oriented methodology) 을 활용하여 정보모델을 정의합니다. 즉, 클래스와 클래스의 특성을 결정하는 애트리뷰트로 UM3 정보모델을 정의합니다. 정보모델을 구성하는 클래스와 애트리뷰트는 컴퓨터, 센서 등의 물리적 자원을 추상화하여 정의하거나, 혹은 측정주기 등과 같은 논리적 자원을 추상화하여 정의할 수도 있습니다.

클래스가 인스턴스 (instance) 화 되어 컴퓨터의 메모리에 상주할 경우 이를 오브젝트 (object) 로 정의합니다. 오브젝트 클래스는 클래스와 동일한 의미이며, 애트리뷰트 클래스 혹은 애트리뷰트 오브젝트는 특정 클래스를 구성하고 있는 애트리뷰트가 속한 클래스 혹은 애트리뷰트 클래스가 인스턴스화 된 오브젝트로 정의됩니다.

5.2. 응용서비스의 정의

본 권고안은, 매니저가 원격에 설치된 센서의 측정값을 에이전트를 통해 수집하는 경우, 혹은 매니저가 에이전트를 통해 원격에 설치된 제어장치를 제어하는 경우, 이를 응용서비스의 영역으로 구분합니다. 즉, 서비스가 제공하는 주요 기능이 원격 센서 및 장치에 대한 데이터 수집과 장치제어인 경우, 센서 데이터의 수집과 원격장치의 제어를 위한 데이터의 송수신은 응용서비스가 제공하는 본질적인 기능으로 분류합니다. 그림 4는 특정 지점의 온도를 측정하기 위한 매니저와 에이전트, 응용서비스 정보모델과의 관계를 나타냅니다. 그림 4에서 보는 것과 같이 정보모델을 구성하는 오브젝트 Temperature는 위참조의 그림 3과는 달리 응용서비스를 위한 논리적인 오브젝트입니다. 즉, 해당 오브젝트는 원격지의 특정 대상에 대한 온도를 감시하기 위한 것이며, 이러한 목적으로 정보모델이 사용되는 경우, 본 권고안은 이를 응용서비스 영역으로 분류합니다.

5.3. 서비스관리의 정의

본 권고안은, 매니저가 에이전트를 통해 원격에 설치된 센서에 대해, 센서의 정상작동 여부, 점검, 교체 등을 위한 데이터를 수집하는 경우, 이를 서비스관리 영역으로 분류합니다. 본 권고안은, 매니저가 에이전트를 통해 원격에 설치된 제어장치의 정상작동 여부, 점검 교체 등을 위한 데이터의 수집, 원격 수리 등을 위한 데이터를 수집하거나 전송하는 경우, 이를 서비스관리 영역으로 분류합니다.

이러한 응용서비스가 웹 서버를 기반으로 제공되는 포털 서비스일 경우, 서비스의 본질적인 기능은

클라이언트 웹브라우저를 통해 서버에 접속하여 정보를 다운로드 받는 것으로 정의할 수 있습니다. 즉 웹브라우저는 매니저의 역할을 담당하며, 웹 서버는 에이전트의 역할을 담당하는 것으로 볼 수 있습니다.

위의 웹 서버와 웹브라우저를 이용한 정보의 수집기능을 제공하는 서비스 구성자원에 있어서, 웹서버 소프트웨어, 하드웨어 서버, 네트워크 등이 정상적으로 작동 중인가를 확인하고, 비정상작동 중인 구성자원에 대한 수리 등의 절차가 진행되는 경우, 본 권고안은 이를 서비스관리 영역으로 정의합니다.

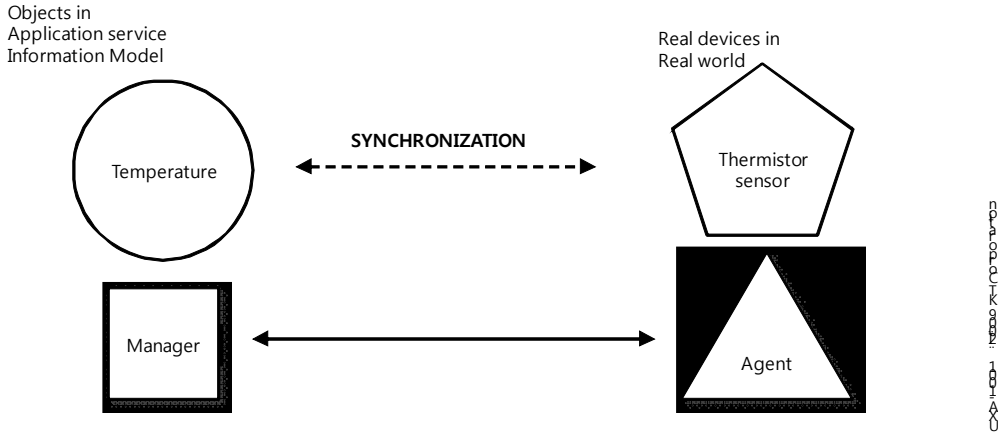


그림 4-응용서비스 영역의 개념적 구조

5.4. 서비스관리 정보모델과 응용서비스 정보모델의 관계

그림 5 는 본 권고안이 정의하는 서비스관리 정보모델과 응용서비스 정보모델의 관계들 중 하나를 보여주고 있습니다. 즉, 응용서비스 정보모델을 추가로 정의해야 할 경우, 새롭게 정의된 오브젝트 클래스는 본 권고안이 정의하는 서비스관리 정보모델의 클래스들로부터 속성 (attribute)을 상속 (inherit)받아 사용할 수 있습니다. 혹은 응용서비스 정보모델의 오브젝트 클래스가 서비스관리 정보모델의 오브젝트 클래스를 종속관계 (aggregation relationship) 로 설정하여 활용할 수도 있습니다.

5.5. UM3 통신서비스 모델의 활용

UM3 통신서비스 모델은 전체 서비스를 7개로 분류하고 각 서비스 별로 별도의 오퍼레이션을 정의하였습니다. 단, UM3 프로토콜을 사용하는 서비스 시스템의 확장성과 유연성을 높이기 위해 그 오퍼레

이션의 종류를 최소로 줄여 정의함으로써 일반적인 응용 소프트웨어를 위한 프로토콜들이 갖고 있는 문제를 해결하였습니다.

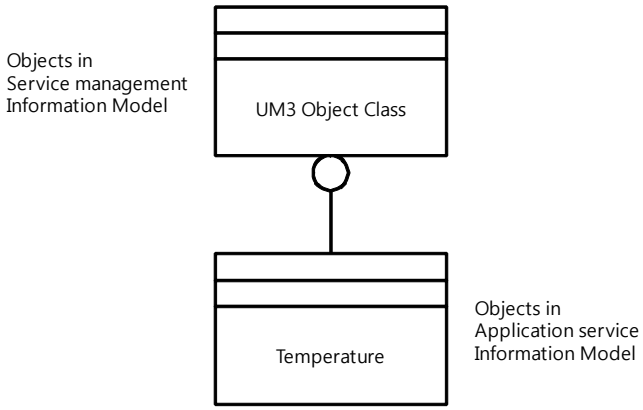


그림 5-응용서비스 모델과 서비스관리 모델의 관계

UM3 프로토콜의 정보모델은 매니저와 에이전트가 데이터를 교환하기 위해 유지하고 관리해야 하는 데이터 구조입니다. 매니저가 에이전트의 데이터를 가져와 자신이 관리하는 정보모델의 데이터를 에이전트의 최신상태와 동일하게 유지하기 위해서는, 에이전트와 미리 약속된 형태로 데이터를 인코딩(encoding)하여 주고 받아야 합니다.

UM3 프로토콜이 정의하고 있는 오퍼레이션들에 있어서, 매니저와 에이전트에게 보내는 명령들은 대부분의 경우 매니저가 에이전트에게 데이터를 요청하는 GET 형태의 통신과, 매니저가 에이전트에게 특정 값을 보내어 에이전트로 하여금 자신과 동일한 상태를 유지토록 하는 SET 형태의 명령들입니다. 이렇게 GET 혹은 SET 형태의 명령을 전송할 때 매니저는 에이전트가 관리하는 정보모델에서 특정 오브젝트와 해당 오브젝트의 애트리뷰트를 지정하고 해당 값을 가져오거나 보내게 됩니다.

서비스의 확장성을 확보하기 위해 UM3 프로토콜은 오퍼레이션의 종류를 최소한으로 규정하고 해당 오퍼레이션의 파라메터 값을 매니저와 에이전트가 관리하는 정보모델을 구성하는 오브젝트로 정의함으로써 프로토콜의 확장성과 유연성을 확보할 수 있게 설계되어 있습니다.

5.6. UM3 프로토콜의 데이터 타입의 종류 및 구분

UM3 프로토콜이 사용하는 데이터 타입의 종류는 아래와 같습니다.

- 기본 타입 (Primitive type)
- 복합형식 타입 (Complex format type)
- 클래스 타입 (Class type)

본 권고안이 사용하는 기본 타입은 일반적인 컴퓨터에서 사용하는 기본 타입들과 거의 유사합니다. 즉, 정수, 실수, 문자열 등과 같은 형태의 타입들로 이루어져 있습니다. 복합형식 타입은 UM3 SEQUENCE OF 등 여러 종류의 타입을 갖는 1 개 이상의 오브젝트들을 나열하는 방식입니다. 마지막으로 클래스 타입은 애트리뷰트를 가지며 인스턴스화 되어 메모리에 상주하게 되는 형태의 타입입니다.

상기 모든 타입들은 서로를 구분하기 위한 고유의 클래스 아이디 값을 갖고 있습니다.

5.7. UM3ClassIdentifier와 UM3ObjectName 클래스 타입의 활용

앞서 기술한 바와 같이 UM3 통신서비스 모델의 오퍼레이션들은 UM3 정보모델을 파라메터로 활용합니다. 이는 UM3 프로토콜 권고안이 정의하는 정보모델의 오브젝트 클래스들에 대한 고유의 식별자가 필요함을 뜻합니다. UM3ClassIdentifier 타입은 4바이트 크기의 부호화하지 않은 정수 (unsigned integer)로 그 기본타입 (primitive type)이 지정되어 있습니다. 예를 들어 본 권고안이 정의한 Gateway 오브젝트 클래스는 12의 정수값으로 정의되어 있습니다.

동일한 오브젝트 클래스로부터 생성된 인스턴스 혹은 오브젝트를 구분하기 위한 식별자는 UM3ObjectName 타입을 활용합니다. 예를 들어 Gateway 오브젝트 클래스로부터 생성된 오브젝트들 중 하나의 이름을 “first gateway”라 명명할 경우, 그리고 이 값을 저장할 변수 혹은 애트리뷰트의 이름이 name일 경우, name 변수의 타입은 UM3ObjectName 타입이며, 이 변수에 저장된 값은 “first gateway”가 됩니다.

본 권고안이 정의하는 오퍼레이션은 이상과 같은 UM3ClassIdentifier 타입의 변수와 UM3ObjectName 타입의 변수를 이용하여 정보모델을 구성하는 오브젝트들 중에 필요한 오브젝트를 식별하여 데이터의 동기화를 이루게 됩니다.

5.8. ASN.1 정규표현식의 활용

본 권고안은 모든 정보모델과 통신서비스 모델의 표현에 ITU-T (국제전기통신 표준화 기구)가 정의한

ASN.1 정규표현식 (formal language)을 병행하여 표기하고 있습니다. ASN.1 정규표현식은 통신 프로토콜을 정의하는 권고안의 데이터 및 오퍼레이션을 표현하기 위해 널리 활용되는 표현식입니다. 특정 데이터 구조와 기능을 컴퓨터 혹은 이와 유사한 기능을 제공하는 장치에서 운용되는 소프트웨어로 구현하거나, 혹은 하드웨어장치를 이용하여 구현하기 위해서, 자연어 (natural language) 혹은 특정 프로그래밍 언어 (programming language)로만 기술할 경우 혼란과 오류가 발생할 수 있으므로 특정 프로그래밍 언어에 종속되지 않는 형태의 표현식인 ASN.1 정규표현식으로 이를 표현합니다.

그러나, 이러한 원래의 취지와는 달리 ASN.1 정규표현식은 그 복잡성과 생소함으로 인해 특정 기술분야 특히, 프로토콜 관련 분야에서 일종의 기술장벽으로 인식되고 있는 것도 사실입니다. 따라서, 본 권고안은 정의된 모든 정보모델과 통신서비스모델의 오브젝트 클래스, 오퍼레이션 등을 우리가 일상적으로 사용하는 자연어와 그림으로 상세하게 기술하고, 더불어 ITU-T가 정의한 ASN.1 정규표현식을 병기하는 형식을 취함으로써 이해도를 높이고 오류를 방지할 수 있게 노력하였습니다.

상기 ASN.1 정규표현식에 대한 국제표준 권고안은 ITU-T 홈페이지에서 무료로 다운받을 수 있으며, 그 URL은 http://www.itu.int/rec/T-REC-X/e 입니다. 해당 URL에서 참고할 권고안의 번호는 X.208, X.209, X.680, X.681, X.682, X.683 입니다. 또한 ASN.1 정규표현식에 대한 이해를 돕기 위한 자료로는 http://www.oss.com/asn1/booksintro.html 의 서적리스트를 참고하시기 바랍니다.

인코딩 룰에 있어서, 본 권고안은 ITU-T가 정의한 BER 등 여러 가지 인코딩 룰의 그 기본 틀만을 활용하며 UM3 SER 인코딩 룰을 새롭게 정의합니다.

5.9. XML 데이터의 지원

UM3 프로토콜은 정보모델의 정의를 통해 오브젝트 혹은 오브젝트 애트리뷰트의 값을 0과 1의 비트의 조합으로 생성하는 인코딩 룰을 정의하고 있습니다. 이는 매우 간편하면서도 낮은 비용으로 TCP/IP 응용계층의 프로토콜을 구현하는 일반적인 방법입니다. 그러나, UM3 프로토콜도 XML 기반의 데이터교환방식을 지원하도록 설계되어 있습니다. 즉, UM3 프로토콜의 인코딩 룰은 UM3 SER 과 XML 방식의 두 가지로 정의됩니다.

XML 데이터 교환방식을 사용할 경우, 이는 해당 서비스의 원가상승 요인으로 작용할 수 있습니다. 즉, UM3 SER을 사용하는 경우보다 더 많은 메모리를 게이트웨이 장치에 장착하여야 하며, XML을 구성하는 태그 등의 구분 (parsing)을 위해 더 많은 데이터 처리 시간이 소요될 수도 있습니다. 다만, 최근의 기술동향이 저사양의 컴퓨터로부터 고사양의 컴퓨터로 점점 그 추세가 옮겨가고 있는 상황임을 감안할 때, XML 방식의 데이터 교환은 그 수요가 더욱 크게 증가할 것으로 예측됩니다. 따라서, 본 권고안이 정의하는 XML 기반의 데이터 교환 방식을 반드시 지원하도록 권고합니다.

단, 본 권고안이 정의하는 XML 기반의 데이터 교환은 SOAP (Simple object access protocol) 프로토콜을 기반으로 데이터를 송수신하는 Web service 와는 다른 개념입니다. 즉, 게이트웨이와 이들로부터 데이터를 송수신하는 서버측 응용 소프트웨어 사이의 데이터 표현을 위해 텍스트기반의 XML 형식을 사용할 뿐, 이는 WSDL (Web service definition language) 등으로 정의되는 웹 서비스와는 다른 개념임에 유의해야 합니다.

5.10. 비정형 데이터의 송수신 (Unstructured data type)

UM3 정보모델은 본 권고안이 제안하는 정보모델 클래스와 더불어 특정 데이터 타입 혹은 특정 데이터 구조로 고정되어 있지 않은 비정형데이터의 송수신을 지원합니다. 이는 UM3UnstructuredData 오브젝트를 이용해 정의되며 UM3 프로토콜의 장점들 중의 하나인 self descriptive 특성을 이용하여, 데이터를 송신하는 것과 동시에 해당 데이터의 schema 정보를 함께 송신할 수 있게 합니다.

6. UM3 프로토콜과 ITIL

6.1. ITIL 지원 범위

서비스관리 정보모델 (Service management information model) 은 앞서 기술한 바와 같이 원활한 서비스를 제공하기 위해 필요한 물리적인 자원과 논리적인 자원에 대해 객체지향 방법론을 적용하여 이들을 추상화한 결과입니다. 이렇게 서비스관리 정보모델을 구성하는 이유는 UM3 프로토콜을 사용하는 서비스 시스템의 서비스관리 업무를 지원하기 위함이며, 서비스관리 업무는 서비스관리 절차 (Service management process) 와 동일한 용어입니다.

세계적으로 통용되고 있으며 많은 기업과 조직들이 활용하는 서비스관리 절차는 ITIL 입니다. ITIL은 세계의 수많은 서비스관리 업무를 담당하는 조직들이 채택하고 있는 여러 가지 서비스 관리 업무를 모두 조사하여 이를 체계적으로 분류하고 가장 합리적이고 효율적인 절차의 형태로 다시 정의한 일종의 국제 표준입니다. ITIL v3 이 정의한 서비스관리 업무의 영역은 다음과 같습니다.

UM3 프로토콜은 표 1의 여러 서비스관리 프로세스들 중 '✔' 기호로 표시된 프로세스들을 중심으로 서비스관리 정보모델을 정의합니다. 이는 UM3 프로토콜이 ITIL 서비스관리 프로세스를 지원하기 위해 반드시 갖고 있어야 할 송수신 데이터의 종류를 의미하기도 합니다. 특히 이들 중 아래에 열거한 서비스관리 프로세스는 반드시 지원 가능해야 하며 이는 아래의 서비스관리 영역은 ITIL 이 정의한 다른 서비스관리 프로세스의 기초 혹은 시작점이 되기 때문입니다.

- Service asset and configuration management (구성관리)
- Event management (이벤트관리)
- Service level management (서비스수준관리)

6.2. 구성관리의 정의

구성관리는 서비스를 구성하는 서비스자원 (Asset)과 해당 자원의 구성 (Configuration)에 관한 서비스관리 프로세스입니다. 구성관리 프로세스는 각 응용서비스에 따라 조금씩 달라질 수 있으나 그 기본적인 기능은 다음과 같이 정의할 수 있습니다.

- 서비스를 구성하는 물리적인 자산 (asset) 과 논리적인 구성자원 (configuration item) 들에 대해 이들을 구분하고, 제어하며, 그 정보를 기록하고, 보고하며, 검증하며, 주기적인 감사활동을 진행
- 서비스를 구성하는 자산과 구성정보에 관해 무결성을 유지할 수 있게 그 정보에 대한 관리

및 보호 기능을 제공

이상과 같은 구성관리 프로세스의 목적을 달성하기 위해서는 서비스를 구성하고 있는 자원들에 대한 자산관리 측면의 자료를 저장, 출력할 수 있어야 합니다. 또한, 구성관리 프로세스는 불필요하고 부적절하며 공인되지 않은 구성자원 혹은 구성자산에 대한 변경을 방지할 수 있어야 하며, 변경관리 프로세스 전반에 걸친 제어가 가능해야 합니다.

이러한 구성관리 기능을 제공하기 위해 서비스관리 시스템은 서비스를 구성하고 있는 자산을 분류하여 클래스로 정의하고, 이들을 인스턴스화 하여 해당 자산과 구성자원에 대한 정보를 기록하며, 필요할 경우 제반 사항을 일목요연하게 정리하여 보고할 수 있어야 하며, 실제 구성자산과 구성자원들이 저장되어 관리중인 정보와 동일한 정보를 갖고 있는 지를 검증하는 기능들을 제공해야 합니다.

표 1. ITIL version 3의 서비스관리 프로세스 분류

Volume	Process	UM3 support
Service strategy	Service portfolio management	
	Demand management	
	IT financial management	
Service design	Service level management	✓
	Availability management	✓
	Capacity management	✓
	IT service continuity management	
	Information security management	
	Supplier management	
	Service catalog management	
Service transition	Service asset and configuration management	✓
	Validation and testing	
	Release and deployment management	
	Change management	✓
	Knowledge management	
Service operation	Event management	✓
	Incident management	✓
	Problem management	
	Request fulfillment management	✓
	Access management	✓
Continual service improvement	Service level management	✓
	Service measurement and reporting	✓
	Continual service improvement	

따라서 이러한 구성관리 프로세스를 지원하는 프로토콜은 서비스를 구성하고 있는 모든 자산에 대해 자동으로 자산의 분류, 자산의 종류, 자산을 구성하고 있는 부품, 도입 및 사용 이력 추적, 다른 구성

자산과의 관련성 등 모든 종류의 정보를 송수신 할 수 있는 형태로 정의되어야 합니다. UM3 프로토콜 권고안은 상기 구성관리 영역의 데이터를 송수신하기 위해 서비스관리 정보모델을 정의하고 있습니다.

6.3. 이벤트관리의 정의

ITIL v3 이 정의하는 이벤트 (event) 는 ITIL v2 에서 정의했던 인시던트 (incident) 보다 한 단계 아래 혹은 제일 먼저 접하게 되는 사건 (incident) 으로 정의할 수 있습니다.

이벤트관리 프로세스는 다음과 같이 정의됩니다.

- 서비스 제공 도중 발생하는 모든 정상, 비정상 이벤트를 모니터링하고 특히 비정상 이벤트를 감지하고 이를 보고하는 기능과 절차

인시던트관리 (Incident management) 프로세스와 문제관리 (Problem management) 프로세스는 위의 이벤트 관리 프로세스와는 달리 가장 빠른 시간 안에 서비스를 다시 원상 복구하는 것을 목표로 하고 있습니다. 즉, 이벤트관리 프로세스는 서비스의 상태를 모니터링하고 정상 비정상을 판별하는 프로세스이며, 인시던트 및 문제관리 프로세스는 이벤트관리 프로세스로부터 올라오는 비정상 관련 인시던트들에 대한 처리 프로세스로 볼 수 있습니다.

따라서 이벤트관리 프로세스를 지원하기 위해서는 상기 구성관리 영역에서 정의한 구성자산 및 구성자원들에 대한 모니터링과 이를 위한 데이터 송수신이 가능해야 합니다. 또한 각 구성자산과 구성자원들에 대한 상태를 분류하고, 발생하는 이벤트에 대해 정상 비정상 확인 등을 위한 데이터 송수신도 지원할 수 있어야 합니다.

6.4. 서비스수준관리의 정의

서비스수준관리는 서비스를 제공하는 동안 발생하는 인시던트와 문제 (problem) 들에 대해 서비스 제공 수준에 영향을 미치는 범위를 측정하고, SLA (Service level agreement) 에 정의한 서비스의 수준을 만족할 수 있도록 하기 위한 관리 프로세스 입니다.

서비스수준관리는 SLA, OLA (Operational level agreement) 등 서비스의 수준과 관련된 계약 혹은 약속들이 서비스의 수준에 일치하도록 해야 합니다.

다만 프로토콜의 관점에서 본다면 상기 이벤트관리 프로세스와 마찬가지로 서비스수준관리 또한 지속적인 서비스 수준에 대한 모니터링과, 모니터링 결과에 따라 도출되는 서비스의 수준관련 리포트, 그리고 고객의 정기적인 리뷰 등에 관한 일련의 기능과 절차를 정의한 관리 프로세스 입니다.

UM3 프로토콜이 지원하는 서비스수준관리 관련 데이터의 형식은 응용서비스 정보모델에서 함께 정의될 수 있습니다. 이는 서비스수준관리에는 서비스의 종류와 그 특성이 반영되어야 하기 때문입니다.

6.5. UM3 프로토콜의 ITIL 지원 방식

앞서 언급한 바와 같이 UM3 서비스관리 정보모델은 물리적 자원 및 논리적인 자원들에 대한 모델링 결과로 생성된 클래스들을 정의하고 있습니다. 이는 클래스의 정의에 있어서 서비스 구성관리를 위한 요소, 이벤트 관리 및 인시던트관리를 위한 요소, 서비스 수준 관리를 위한 요소 등이 모두 반영되고 있음을 뜻합니다. 특히 이들 중 구성관리 및 이벤트 관리 분야는 모든 클래스의 정의에 필수적으로 포함되어 있습니다.

즉, 서비스 자산 (asset) 과 서비스 구성 자원 (configuration item) 의 측면에서 정의된 애트리뷰트들은 유지보수 담당자의 연락처, 장치의 시리얼넘버, configuration tree 에서의 위치 등과 같은 애트리뷰트들입니다. 이들 애트리뷰트들은 시스템의 구성이 어떤 형상으로 이루어졌는가, 서비스 시스템이 어떤 자원들의 구성으로 이루어져 있으며 이들의 물리적 지리적 위치 등은 어떤 관계에 있는가 등의 서비스 구성에 관한 여러가지 정보들을 갖고 있게 됩니다.

이벤트 관리를 위한 애트리뷰트 혹은 UM3 오퍼레이션의 예는 에이전트가 관리하는 information tree 의 오브젝트 애트리뷰트의 값을 지속적으로 모니터링하여 해당 애트리뷰트의 값의 변화를 감지하여 특정 이벤트를 검출하는 방식이 있습니다.

본 권고안이 정의하는 UM3 프로토콜은 상기 정보모델의 구성을 통해 물리적 자원과 논리적 자원에 대한 모델링 과정을 진행하고, 모델링 과정의 결과물인 클래스 오브젝트의 애트리뷰트 값들에 대한 매니저와 에이전트 사이의 동기화과정을 UM3 오퍼레이션으로 진행하는 형식으로 정의합니다. 즉, 서비스 구성관리, 이벤트 관리, 서비스 수준 관리 등의 ITIL 프로세스 관련 영역은 결국 정보모델을 구성하는 클래스의 모델링을 통해 클래스 애트리뷰트의 형태로 반영되게 되는 것입니다.

7. UM3 프로토콜 지원 레벨과 N, NL 필드의 인코딩 방식

본 권고안의 구성은 앞서 소개한 바와 같이 정보모델의 정의와 통신서비스모델의 정의 및 UM3 SER 인코딩 룰의 정의 등으로 구성되어 있습니다. 이들 중 UM3 SER 인코딩 룰과 통신서비스모델 및 정보모델의 UM3 기본 타입만을 지원할 경우 이를 'UM3 LEVEL 1' 을 지원하는 것으로 정의합니다. 즉, 클래스 타입 등을 지원하지 않고 기본 타입과 해당 기본 타입의 UM3 SER 인코딩 룰 만을 적용한 후, 해당 APUD 를 UM3 오퍼레이션을 이용하여 데이터를 송수신하는 경우입니다. 이 때 UM3 오퍼레이션의 파라메터로 주어지는 오브젝트의 이름, 애트리뷰트의 이름 등은 큰 의미가 없을 수도 있습니다.

반면에 본 권고안이 정의하는 모든 타입들 즉, 기본 타입, 복합형식 타입, 클래스 타입 등의 모든 정보모델과 통신서비스모델 등을 구현하고, 이를 UM3 SER 인코딩 룰을 적용하여 인코딩한 후, 다시 이를 UM3 오퍼레이션을 통해 주고받을 수 있을 경우 이는 'UM3 LEVEL 2' 를 지원하는 것으로 정의합니다.

표 2. UM3 LEVEL 1과 UM3 LEVEL 2 지원의 구분

지원항목	UM3 LEVEL 1	UM3 LEVEL 2
UM3 Primitive type	O	O
UM3 Complex format type	O	O
UM3 Class type	×	O
UM3 SER	O	O
UM3 Operations	△	O

표 2. UM3 LEVEL 1과 UM3 LEVEL 2 지원의 구분에서 보는 바와 같이 UM3 LEVE 1 을 지원하는 경우 UM3 Class type 을 지원하지 않습니다. 이는 본 권고안이 정의한 정보모델을 구현하지 않아도 된다는 의미이며, 궁극적으로는 오브젝트와 클래스의 구분 없이 데이터를 송수신한다는 의미이기도 합니다. 이는 산업현장에서 많이 사용되고 있는 Modbus 프로토콜과 유사한 개념이며, Modbus 프로토콜이 사용하는 16진수 4 비트 단위의 데이터 포맷을 대신하여 UM3 SER 인코딩 룰과 UM3 기본 타입 (Primitive type) 만을 사용하는 경우로 볼 수 있습니다. 특히 이러한 간단한 형태의 데이터 포맷을 사용해야 할 경우, 본 권고안이 정의하고 있는 UM3 SER 인코딩 룰은 오브젝트 이름을 기록하는 N 필드를 생략하기 위해 NL 필드의 값을 0_{10} 으로 두어 N 필드를 사용하지 않음을 표시하기도 합니다. 또한 본 권고안이 정의하는 정보모델의 클래스 타입들을 지원하지 않을 경우, 이는 매니저와 에이전트의 관계가 아닌 peer to peer 데이터 통신을 위한 경우로 볼 수 있으며, 오브젝트의 이름과 클래스 타입, 애트리뷰트의 이름과 그 값 등의 개념을 사용하지 않는 경우이기도 합니다. 특히 UM3 LEVEL 1 을 지원하는

경우 정보모델의 개념을 적용하지 않으며 따라서 UM3 오퍼레이션의 클래스아이디와 오브젝트의 이름 등에 관한 파라메터를 사용하지 않게됩니다. 다만 위와 같은 경우에는 클래스아이디는 UM3 기본 타입의 클래스아이디와 오브젝트 이름의 경우 일반적인 변수의 이름을 사용하여 UM3 프로토콜을 구현할 수도 있습니다. 표 2 의 표시는 UM3 오퍼레이션에 대한 UM3 SER 인코딩 룰을 그대로 적용하되, 파라메터로 주어지는 클래스아이디와 오브젝트의 이름 등을 본 권고안이 정의한 UM3 Primitive type 만을 사용하는 경우로 볼 수 있습니다. 또한 상기와 같은 UM3 LEVEL 1 만을 지원하는 경우 거의 대부분은 GetObjectAttributeValue 혹은 SetObjectAttributeValue 오퍼레이션 만을 이용하는 경우가 대부분입니다.

8. UM3 Primitive Type

UM3 프로토콜이 사용하는 기본타입 (UM3 primitive type)은 표 3-UM3 기본 타입 [와 같습니다. 표 3-UM3 기본 타입 [에서 타입으로 표기된 열은 UM3 기본 타입들을 나타냅니다. 키워드로 표현된 열은 UM3 프로토콜이 사용하는 해당 타입의 키워드를 나타내며, 설명란의 표기는 일반적인 C 프로그래밍 언어에서 사용하는 데이터 타입 키워드로 해당 타입을 설명하며, 클래스 아이디 값은 해당 기본 타입이 UM3 SER을 활용하여 인코딩 (encoding) 될 경우, 해당 APDU의 타입을 구분하기 위해 APDU의 특정 필드에 저장되는 값을 나타냅니다.

UM3 primitive types defined in the UM3 protocol are shown in Table 3 in the Type column. The Keyword column shows the keywords of the corresponding type defined in the UM3 protocol, and the Description column describes the corresponding type by using data type keywords used in common C programming language. The class ID column shows the values stored in specific fields in APDU for identifying the corresponding APDU type when the corresponding primitive type is encoded by using UM3 SER.

표 3-UM3 기본 타입 [UM3 Primitive Type]

타입	의미 (키워드)	설명	클래스 아이디
Type	**Meaning (Keyword)**	**Description**	**Class ID**
UM3Integer16	SHORT INTEGER	short int	30
UM3Integer32	INTEGER	Int	31
UM3Integer64	LONG INTEGER	long long int	32
UM3UnsignedInteger16	UNSIGNED SHORT INTEGER	unsigned short int	33
UM3UnsignedInteger32	UNSIGNED INTEGER	unsigned int	34
UM3UnsignedInteger64	UNSIGNED LONG INTEGER	unsigned long long int	35
UM3CharacterString	CHARACTER STRING	char[]	36
UM3BitString	BIT STRING	char[]	37
UM3OctetString	OCTET STRING	char[]	38
UM3Boolean	BOOLEAN	bool	39
UM3Enumerated	ENUMERATED	enum	40
UM3Real	REAL	double	41
UM3Null	NULL	null	42
UM3DateTime	DATETIME	int	43
UM3ClassIdentifier	UNSIGNED INTEGER	unsigned int	45
UM3ObjectName	CHARACTER STRING	char[]	46

이상과 같은 UM3 기본 타입에 대한 ASN.1 정규표현식은 아래와 같습니다.

ASN.1 regular expression for the above UM3 primitive types is as follows.

```
UM3 Primitive Type ::=
        UM3Integer16 |
```

UM3Integer32 |
UM3Integer64 |
UM3UnsignedInteger16 |
UM3UnsignedInteger32 |
UM3UnsignedInteger64 |
UM3CharacterString |
UM3BitString |
UM3OctetString |
UM3Boolean |
UM3Enumerated |
UM3Real |
UM3Null |
UM3DateTime |
UM3ClassIdentifier |
UM3ObjectName

상기 ASN.1 정규식의 표현에서 '|' 기호는 OR 의 뜻을 나타냅니다. 즉, UM3 Primitive Type은 UM3Integer16 타입 혹은 UM3Boolean 등의 타입으로 구성될 수 있다는 의미입니다. 또한 '::=' 기호는 좌측의 문자열이 나타내는 타입은 '::=' 기호 우측의 표현으로 정의된다는 뜻입니다.

이상과 같은 UM3 프로토콜의 UM3 Primitive Type을 갖는 값들을 다음과 같이 정의합니다.

The "|" symbol in the above ASN.1 regular expression is an OR operation, which means that UM3 primitive types can be configured with either UM3inter16 type or UM3Boolean type. The "::=" symbol means that the type represented by the character string on the left side is defined as the expression on the right side of the "::=" symbol.

The values of the above UM3 primitive types in the UM3 protocol are defined as follows.

UM3 Primitive Value ::=
um3Integer16Value |
um3Integer32Value |
um3Integer64Value |
um3UnsignedInteger16Value |
um3UnsignedInteger32Value |
um3UnsignedInteger64Value |
um3CharacterStringValue |
um3BitStringValue |
um3OctetStringValue |
um3BooleanValue |
um3EnumeratedValue |
um3RealValue |
um3NullValue |
um3DateTimeValue |
um3ClassIdentifierValue |
um3ObjectNameValue

상기 각 타입별 값들을 정의하는 이유는 해당 타입들에 대한 일반적인 값들을 명명하고 해당 값들이 구성되는 방법을 ASN.1 정규표현식으로 정의하기 위해서 입니다.

다음으로 상기 모든 UM3 Primitive Type과 키워드 들에 대한 설명을 기술합니다. 단, 편의상 본 권고안은 INTEGER, REAL 등과 같은 UM3 키워드에 대한 설명을 표 3-UM3 기본 타입 [으로 대체하고 바로 UM3 Primitive Type 에 대한 정의를 기술합니다.

The purpose of defining the value of each type in the table above is to assign names for general values of the corresponding types and to define the method to configure the corresponding values with ASN.1 regular expressions.

The next column provides the descriptions for all of the above UM3 primitive types and keywords. The description of the UM3 keyword such as INTEGER and REAL are replaced with a primitive type in Table 3 in this recommendation for convenience, and the definition of the UM3 primitive type is provided.

8.1. UM3Integer16 Type

UM3Integer16 타입은 2 바이트로 구성된 short integer 타입을 나타냅니다. 일반적인 32비트 컴퓨터에서 나타내는 short int 키워드와 동일한 타입입니다. 표현할 수 있는 수의 범위는 0을 포함하는 -32,768 ~ 32,767 입니다.

UM3Ineger16 타입의 ASN.1 정규표현식의 정의는 아래와 같습니다.

UM3Integer16 is short integer type with two bytes of data. It is the same as the short int keywords used in general 32 bit computers. The range of numbers that can be expressed with this type is -32,768 ~ 32767, including 0.

The definition of UM3Integer16 using ASN.1 regular expressions is as follows.

UM3Integer16 ::= SHORT INTEGER

8.2. UM3Integer32 Type

UM3Integer32 타입은 4 바이트로 구성된 integer 타입을 나타냅니다. 일반적인 32비트 컴퓨터에서 나타내는 int 키워드와 동일한 타입입니다. 표현할 수 있는 수의 범위는 0을 포함하는 -2,147,483,648 ~ 2,147,483,647 입니다. UM3Intger32 타입의 ASN.1 정규표현식의 정의는 아래와 같습니다.

UM3Integer32 is an integer type with four bytes of data. It is the same as the int keyword used in general 32 bit computers. The range of numbers that can be expressed with this type is -2,147,483,648 ~ 2,147,483,647, including 0. The definition of UM3Integer32 using ASN.1 regular expressions is as follows.

UM3Integer32 ::= INTEGER

8.3. UM3Integer64 Type

UM3Integer64 타입은 부호화 정수를 표현하기 위해 8 바이트를 사용합니다. 일반적인 32 비트 컴퓨터에서 나타내는 long long int 타입과 동일한 타입입니다. 32비트 컴퓨터에서는 일반적으로 long int 타입과 int 타입의 바이트 수가 동일하므로 UM3 프로토콜은 해당 타입을 LONG LONG INTEGER로 표현하지 않고 LONG INTEGER 로 표현하는 점에 주의하여야 합니다. 표현할 수 있는 수의 범위는 0을 포함하는 -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 입니다. UM3Integer64 타입의 ASN.1 정규표현식의 정의는 아래와 같습니다.

UM3Integer64 type uses 8 bytes to represent signed integer numbers, and it is the same as the long long int type in general 32 bit computers. Caution is required because this type is defined as LONG INTEGER instead of LONG LONG INTEGER in the UM3 protocol (long int type and int type use same number of bits). The range of numbers that can be expressed with this type is -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807, including 0. The definition of UM3Integer64 using ASN.1 regular expressions is as follows.

UM3Integer64 ::= LONG INTEGER

8.4. UM3UnsignedInteger16 Type

UM3UnsignedInteger16 타입은 부호화하지 않은 정수를 표현하기 위해 2바이트를 사용합니다. 일반적인 32비트 컴퓨터에서 사용하는 unsigned short int 타입과 동일한 타입입니다. 표현할 수 있는 값의 범위는 0 ~ 65,535 입니다. UM3Integer16 타입의 ASN.1 정규표현식의 정의는 아래와 같습니다.

UM3UnsingedInteger16 type uses two bytes to represent an unsigned integer, and it is the same as the unsigned short int type in general 32 bit computers. The range of numbers that can be expressed with this type is 0 ~ 65,535. The definition of UM3Integer16 using ASN.1 regular expressions is as follows.

UM3UnsignedInteger16 ::= UNSIGNED SHORT INTEGER

8.5. UM3UnsignedInteger32 Type

UM3UnsignedInteger32 타입은 부호화하지 않은 정수를 표현하기 위해 4 바이트를 사용합니다. 일반적인 32비트 컴퓨터에서 사용하는 unsigned int 타입과 동일한 타입입니다. 표현할 수 있는 값의 범위는 0 ~ 4,294,967,295 입니다. UM3UnsignedInteger32 타입의 ASN.1 정규표현식에 의한 정의는 아래와 같습니

다.

UM3UnsignedInteger32 type uses four bytes to represent an unsigned integer number, and it is the same as the unsigned int type in general 32 bit computers. The range of numbers that can be expressed with this type is 0 ~ 4,294,967,295. The definition of UM3UnsignedInteger32 using ASN.1 regular expressions is as follows.

UM3UnsignedInteger32 ::= UNSIGNED INTEGER

8.6. UM3UnsignedInteger64 Type

UM3UnsignedInteger64 타입은 부호화하지 않은 정수를 표현하기 위해 8 바이트를 사용합니다. 일반적인 32비트 컴퓨터에서 사용하는 long long int 타입과 동일한 타입입니다. 또한 UM3Integer64와 마찬가지로 키워드를 LONG LONG INTEGER로 표현하지 않는 점에 주의하여야 합니다. 사용할 수 있는 수의 범위는 0 ~ 18,446,744,073,709,551,615 입니다. UM3UnsignedInteger64의 ASN.1 정규표현식에 의한 정의는 아래와 같습니다.

UM3UnsignedInteger64 type uses eight bytes to represent an unsigned integer number, and it is the same as the unsigned long long int type in general 32 bit computers. Caution is required with UM3Integer64 since this type is not defined as UNSIGNED LONG LONG INTEGER. The range of numbers that can be expressed with this type is 0 ~ 18,446,744,073,709,551,615. The definition of UM3Integer64 using ASN.1 regular expressions is as follows.

‘

UM3UnsignedInteger64 ::= UNSIGNED LONG INTEGER

8.7. UM3CharacterString Type

일반적인 문자열을 나타냅니다. 스트링 값의 인코딩은 UM3 SER이 정의하는 인코딩 방법과는 다른 의미로 사용되며, 유니코드, 한글의 조합형 등을 포함합니다. 이러한 별도의 스트링 값의 인코딩은 본 권고안이 정의하는 UM3CharacterString 타입의 문자열을 한 개 혹은 그 이상의 바이트들을 결합하여 처리할 수 있습니다. 또한 UM3CharacterString은 유니코드, UTF8 등 문자열의 코드값을 지정하는 것과는 관계없이 모든 스트링을 처리할 수 있는 것으로 정의합니다. 또한 UM3CharactgerString 타입은 그 문자열의 끝에 0x00 값을 삽입하여 해당 문자열의 끝을 표시하는 방식을 취합니다.

C, C++, Java, C# 등의 프로그래밍 언어는 UM3CharacterString 타입을 char[], char* 타입 혹은 String 등의 타입으로 표현합니다.

UM3CharacterString 타입의 ASN.1 정규표현식에 의한 정의는 아래와 같습니다.

This type is used for general strings. The encoding of the string value has a different meaning from the encoding method defined with UM3 SER, and it includes Unicode and Korean Combination Code. The encoding of these separate string values can be processed by using one or combining one or multiple UM3CharacterString type strings as defined in this recommendation. UM3CharacterString is defined to process all types of strings independently from the assignment of code values of strings such as Unicode or UTF8. The 0x00 value is inserted at the end of the UM3CharacterString type string to indicate the end of the corresponding string.

The UM3CharacterString type is implemented as char[], char*, or String type in languages like C, C++, Java, and C#.

The definition of UM3CharacterString using ASN.1 regular expressions is as follows.

UM3CharacterString ::= CHARACTER STRING

8.8. UM3BitString Type

이 타입은 상기 UM3CharacterString과는 달리 문자열의 인코딩 방법과는 관계없이 1과 0으로 구성된 비트들로 이루어진 데이터를 표현합니다. 또한 별도의 길이정보가 주어지기 전에는 그 길이를 자체적으로 표현하는 방법은 없습니다. 즉, 데이터의 의미 있는 길이 혹은 단위가 1 비트의 단위일 때 적용하는 데이터 타입입니다. 주로 음성데이터, 이미지데이터, 동영상 및 기타 대용량의 데이터를 송수신할 경우 사용하는 데이터 타입입니다.

C# 등의 프로그래밍 언어는 해당 데이터 타입을 byte[] 등으로 표현합니다. UM3BitString의 ASN.1 정규표현식에 의한 정의는 아래와 같습니다.

Unlike the abovementioned UM3CharacterString type, this type represents data in bits with 1s and 0s independently from the encoding method of the character string. There is no method of internally expressing the length unless separate length information is provided. In other words, this is a data type that is used when the unit of effective length or unit of data is one bit. It is used mainly for transmission or reception of large amounts of data such as voice data, image data, and video.

This data type is implemented as byte[] in the C# programming language. The definition of UM3BitString using ASN.1 regular expressions is as follows.

UM3BitString ::= BIT STRING

8.9. UM3OctetString Type

UM3OctetString 타입은 문자열을 구성하는 데이터의 의미 있는 단위가 4비트 즉, octet인 경우에 사용하는 타입입니다. 특정 숫자 혹은 문자를 16진수 (hex number)로 표현할 때 사용합니다. 특히 Modbus,

Lontalk 등 산업현장의 디바이스간의 통신을 위해 많이 사용하고 있는 프로토콜의 데이터를 표현하기 위해 사용하는 타입입니다. C, C++, C#, Java 등의 프로그래밍 언어는 해당 타입을 UM3CharacterString 혹은 UM3BitString 과 같이 char[], char* 등으로 표현합니다.

UM3OctetString 타입의 ASN.1 정규 표현식에 의한 정의는 다음과 같습니다.

UM3OctetString is a type that is used when the unit of meaningful data configuring the character array is 4 bits i.e. octet. This is used to represent specific numbers or characters as hex numbers. It is especially used for representing the data in protocols commonly used in communications between devices in the industrial field such as Modbus or Lontalk. This type is implemented as char[] or char* like UM3CharacterString or UM3BitString in C, C++, C#, and Java programming languages.

The definition of UM3 OctetString using ASN.1 regular expressions is as follows.

UM3OctetString ::= OCTET STRING

8.10. UM3Boolean Type

UM3Boolean 타입은 2 개의 구분 가능한 값을 갖는 타입입니다. 본 권고안이 정의하는 UM3Boolean 타입은 일반적인 프로그래밍 언어의 bool, Boolean 타입과 같이 True와 False의 두 가지 값을 갖고 있는 것으로 정의합니다. UM3Boolean 타입의 ASN.1 정규 표현식에 의한 정의는 다음과 같습니다.

The UM3Boolean is a data type that has two distinct states. The UM3Boolean type defined in this recommendation is defined as a variable that can have either a True or False value like bool or Boolean type in other general programming languages. The definition of UM3Boolean using ASN.1 regular expressions is as follows.

UM3Boolean ::= BOOLEAN

um3BooleanValue UM3Boolean ::= True | False

상기 ASN.1 정규표현식에 있어서 um3BooleanValue는 UM3Boolean 타입의 변수를 뜻하며, 해당 um3BooleanValue 변수는 True와 False의 두 가지 값으로 표현할 수 있음을 나타냅니다. 이는 정보모델의 관점에서 보는 타입의 정의이며, 해당 타입에 대한 UM3 SER 관점에서의 값, 혹은 비트의 조합은 UM3Boolean 타입의 인코딩 방법에서 상세하게 설명합니다.

The um3BooleanValue in the above ASN1.1 regular expression means a UM3Boolean type variable, and the corresponding um3BooleanValue type variable can have either a True or False value. This is the definition of the type from the perspective of the information model, and the value or combination of bits from the perspective of UM3 SER for the corresponding type will be explained in more detail in the section about the encoding method for

UM3Boolean type.

8.11. UM3Enumerated Type

UM3Enumerated 타입은 타입을 구성하는 각각의 변수들에 특정 아이디 값이 미리 상수 값으로 정의된 타입을 뜻합니다. UM3Enumerated 타입은 C, C++, C#, Java 등의 프로그래밍 언어에서 enum 타입으로 표현됩니다. UM3Enumerated 타입의 ASN.1 정규표현식에 의한 정의는 다음과 같습니다.

UM3Enumerated is a type defined in such a way that constants are assigned to specific ID values of each element of variables. UM3Enumerated type is implemented as enum type in C, C++, C#, and Java programming languages. The definition of UM3Enumerated using ASN.1 regular expressions is as follows.

UM3Enumerated ::= ENUMERATED "{"Enumerations"}"

Enumerations ::=
RootEnumeration |
RootEnumeration "," "…" |
RootEnumeration "," "…" "," AdditionalEnumeration

RootEnumeration ::= Enumeration

AdditionalEnumeration ::= Enumeration

Enumeration ::=
EnumerationItem |
EnumerationItem "," Enumeration

EnumerationItem ::=
identifier |
NamedNumber

NamedNumber ::=
identifier "(" SignedNumber")"

SignedNumber ::=
number | "-" number

상기 ASN.1 정규표현식이 나타내는 의미는, UM3Enumerated 타입의 값은 identifier로 표현된 타입을 구성하는 각각의 변수들만 나열하거나 혹은 변수명에 이어 괄호 안에 번호를 부여하는 방법, 혹은 이미 정해진 번호를 괄호 안에 두어 변수의 값을 지정하는 형태로 정의한다는 뜻입니다. 상기 ASN.1 정규표현식에서 identifier는 변수의 이름을 뜻합니다. number는 숫자를 의미하며 한자리 수를 제외하고 0으로 시작할 수 없는 것으로 정의합니다. 즉, SignedNumber는 부호화 된 정수값을 의미하며, 본 권고안이 정의하는 UM3Integer32의 타입을 갖는 것으로 정의합니다. 이는 C, C++, C#, Java 등과 같은 일반적인

프로그래밍 언어에서 enum 타입의 변수를 정의하는 것과 동일한 문법임을 나타냅니다.

In the above ASN1.1 regular expression, the value of variables of UM3Enumerated type is defined by listing each element by using an identifier, variable name followed by a number in parentheses for assignment, or putting predetermined numbers inside the parentheses. The identifier in the above ASN.1 regular expression is the name of variable, and the number denotes a number that does not begin with 0 except for single digit numbers. SignedNumber is a signed integer number, and its type should be UM3Integer32 as defined in this recommendation. This means that the syntax is the same as the definition of enum type in other general programming languages like C, C++, C#, and Java.

8.12. UM3Real Type

UM3Real 타입은 실수 (real value) 타입의 숫자를 나타내기 위해 사용합니다. 해당 타입을 갖는 변수의 길이는 8 바이트이며, 일반적으로 32비트 컴퓨터의 double 타입의 값과 동일한 의미를 갖습니다. UM3Real 타입의 ASN.1 정규표현식에 의한 정의는 다음과 같습니다.

UM3Real is a type used to represent real numbers. The length of the variable with this type is eight bytes, and it is the same as double type in general 32 bit computers. The definition of UM3Real using ASN.1 regular expressions is as follows.

UM3Real ::= REAL

8.13. UM3Null Type

UM3Null 타입은 그 값으로 항상 0x00의 값을 갖는 상수이며 크기는 1 바이트로 구성됩니다. UM3Null 타입의 ASN.1 정규표현식에 의한 정의는 다음과 같습니다.

UM3Null type is a constant with the value of 0x00 always, and the size is one byte. The definition of UM3Null using ASN.1 regular expressions is as follows.

UM3Null::= NULL

um3NullValue ::= NULL

um3NullValue는 UM3Null 타입으로 정의된 변수이며 해당 변수의 값으로 NULL 키워드가 뜻하는 값을 정의한다는 의미입니다.

ITU-T 권고안은 NULL 키워드로 표현되는 타입에 대해 인코딩 과정을 거치지 않는 것으로 정의되어 있습니다. 이를 UM3 프로토콜 권고안에 정의할 경우 특정 오퍼레이션에 대한 응답에 있어서, 그 결과

를 NULL 키워드로 표현되는 UM3Null 타입으로 응답할 경우, UM3Enumuerated 타입의 값으로 응답하는 것보다 더 간편하고 적은 수의 패킷으로 응답을 할 수 있음을 뜻합니다.

The um3NullValue is a variable defined with UM3Null type, and the above statement means that the value of the corresponding variable is the one defined by the NULL keyword.

The ITU-T recommendation defines the type represented as the NULL keyword as the one without encoding process. When it is defined in UM3 protocol recommendation, it means that the response to a specific operation can be simpler with fewer packets if UM3Null type defined as a NULL keyword instead of UM3Enumerated type is used for responding with the result.

8.14. UM3DateTime Type

POSIX 국제표준을 지원하는 컴퓨터에서는 현재의 시각을 계산할 때 32 비트 혹은 64 비트의 정수값을 이용하여, 1970년 1월1일 00:00시로부터 경과한 시간을 초 단위로 나타냅니다. UNIX 계열의 컴퓨터는 <time.h> 시스템 헤더 파일에 해당 타입이 정의되어 있으며 주로 long 타입으로 정의되어 있습니다. 그러나 운영체제에 따라 int 타입과 long 타입이 동일한 크기를 갖고 있는 경우가 많으며, 특히 Linux 계열의 운영체제를 사용하는 경우 대부분은 int 타입으로 정의되어 있는 경우가 많습니다. MicroSoft의 Windows 운영체제의 경우 4 바이트 크기의 정수로 해당 타입을 나타냅니다.

Computers that support the POSIX international standard use 32 bit or 64 bit integer values to represent the current time, and it is represented as the total elapsed time in seconds from 00:00, Jan 1st, 1970. The corresponding type is defined in the <time.h> system header file in the UNIX family of computers, and it is usually defined as long type. However, int type and long type may have the same size depending on the operating system, and quite often it is defined as int type in most of computers using Linux operating systems. This type is implemented as four byte integers in the Microsoft Windows operating system.

UM3DateTime 타입은 현재의 시각을 나타내기 위해 사용되기도 하지만 주로 시간에 관한 정보의 교환이 필요할 경우 시간을 초단위로 표현하기 위해 사용됩니다. 즉, UM3 프로토콜은 모든 시간을 초 단위로 표현하여 송수신하도록 정의하고 있으며, 해당 UM3DateTime 타입은 일반적인 32비트 컴퓨터에서와 같이 INTEGER 키워드를 사용하는 UM3Integer32로 정의합니다.

The UM3DateTime type is used when exchanging time information in the unit of seconds as well as for representing the current time. In other words, the UM3 protocol defines all time in the unit of seconds for information exchange, and the UM3DateTime type is defined with UM3Integer32 that uses the INTEGER keyword like general 32 bit computers.

본 권고안이 정의하는 UM3DateTime 타입과는 달리, ITU-T 권고안과 일부 ASN.1 정규표현식을 사용하는 표준안에서는 날짜와 시간에 관련된 타입을 문자열의 타입으로 정의하고 있습니다. UM3 프로토콜 권고안은 날짜 및 시각에 대한 정보가 필요할 경우 즉, 사람이 읽고 인지하기 위한 시간정보를 표현

하기 위해서는 UM3DateTime 타입의 변수에 대해, 일반적인 프로그램 개발용 라이브러리가 제공하는 형식의 변경기능을 이용하여 문자열로 변경하고, 이를 화면에 표출하거나 저장하는 것으로 정의합니다.

다음은 ASN.1 정규표현식으로 나타낸 UM3DateTime 타입에 대한 정의입니다.

Unlike UM3DateTime type defined in this recommendation, the ITU-T recommendation and some standards using ASN.1 regular expressions define the type related to date and time as string type. The UM3 protocol recommendation defines this type in such a way that UM3DateTime type variables should be converted to string by using the libraries provided in general program development for screen display or storage to present the time information in human readable form when date and time information is required.

The definition of UM3DateTime using ASN.1 regular expressions is as follows.

UM3DateTime ::= INTEGER

8.15. UM3ClassIdentifier Type

UM3ClassIdentifier 타입은 UM3 프로토콜 권고안이 정의하는 기본 타입, 복합 형식 타입 및 정보모델의 클래스 타입들을 구분하고 그 값을 저장하기 위해 사용합니다. 다음은 ASN.1 정규표현식으로 나타낸 UM3ClassIdentifier 타입에 대한 정의입니다.

UM3ClassIdentifier type is used to identify the basic type, complex type, and class type of the information model defined in the UM3 protocol recommendation. The definition of UM3ClassIdentifier using ASN.1 regular expressions is as follows.

UM3ClassIdentifier ::= UNSIGNED INTEGER

8.16. UM3ObjectName Type

UM3ObjectName 타입은 클래스가 인스턴스화 되었을 경우, 해당 오브젝트를 동일한 클래스로부터 생성된 다른 오브젝트들로부터 구분하기 위해 사용합니다.

다음은 UM3ObjectName 타입의 ASN.1 정규표현식에 의한 정의입니다.

UM3ObjectName type is used to identify various objects created from the same class when a class is instantiated.

The definition of UM3ObjectName using ASN.1 regular expressions is as follows

UM3ObjectName ::= CHARACTER STRING

9. UM3 Complex Format Type

UM3 복합형식 타입 (complex format type) 은 앞서 정의한 UM3 기본 타입들과 이후 정의하는 모든 정보모델들의 클래스 타입들에 있어서, 1 개 이상의 타입들이 리스트 혹은 집합의 형태로 모여서 이루어지는 타입입니다.

UM3 Complex Format The type is combination of more than one UM3 primitive types defined earlier and class types of all information models to be defined later in list or set form.

표 4-UM3 복합형식 타입의 구성 [Configuration of UM3 Complex Format Type]

타입	의미 (키워드)	설명	클래스 아이디
Type	**Meaning (Keyword)**	**Description**	**Class ID**
UM3 SEQUENCE	SEQUENCE	Sequence is important	47
UM3 SEQUENCE OF	SEQUENCE OF	Sequence is important	48
UM3 SET	SET	Sequence is not important	49
UM3 SET OF	SET OF	Sequence is not important	50

표 4-UM3 복합형식 타입의 구성 [에 나타낸 UM3 SEQUENCE 타입과 UM3 SEQUENCE OF 타입, UM3 SET 타입과 UM3 SET OF 타입에 대한 비교는 그림 6와 같습니다. UM3 SEQUENCE 타입과 UM3 SEQUENCE OF 타입은 순서에 의미가 있으며 반드시 그 순서가 지켜져야 하지만 UM3 SET 타입과 UM3 SET OF 타입은 순서에 의미가 부여되지 않는 타입들입니다.

Fig. 6 is the comparison of UM3 SEQUENCE type, UM3 SEQUENCE OF type, UM3 SET type, and UM3 SET OF type shown in Table 4. Sequence has a meaning in UM3 SEQUENCE type and UM3 SEQUENCE OF type. The correct sequence must be flowed, but it doesn’t have meaning for UM3 SET type and UM3 SET OF type.

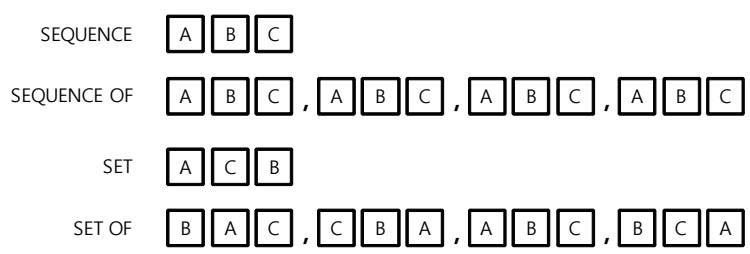


그림 6-UM3 복합형식 타입의 비교 [Comparison of UM3 Complex Format Types]

UM3 SEQUENCE 타입은 1 개 이상의 기본 타입, 복합형식 타입, 클래스 타입 등의 오브젝트들을 순서대로 나열하기 위해 쓰입니다. UM3 SEQUENCE OF 타입은 UM3 SEQUENCE 혹은 UM3 SET 등의 타입들을 가진 복수의 오브젝트들을 순서대로 나열할 때 사용됩니다. UM3 SET 타입은 몇 개의 타입들을 순서에 관계없이 집합의 형태로 표현할 때 사용됩니다. UM3 SET OF 타입은 UM3 SEQUENCE, UM3 SET 등의 타입으로 정의된 1개 이상의 오브젝트들을 순서에 관계없이 나열할 때 사용됩니다.

The UM3 SEQUENCE type is used to list more than one object of primitive type, complex format type, and class type in order. The UM3 SEQUENCE OF type is used to list multiple objects of UM3 SEQUENCE or UM3 SET type in order. UM3 SET type is used to represent multiple types in a set form without ordering. The UM3 SET OF type is used to list multiple objects defined with UM3 SEQUENCE or UM3 SET types without ordering.

UM3 복합형식 타입은 1 개 이상의 오브젝트 혹은 서로 다른 오브젝트에 속한 애트리뷰트들에 대한 그룹을 표현하기 위해 사용됩니다. 이는 UM3 오퍼레이션들 중 그룹에 대한 값들에 대해 UM3 SET 혹은 GET 관련 기능을 제공하는 오퍼레이션들의 파라메터로 사용되거나, 혹은 해당 오퍼레이션의 응답을 담당하는 OperationResponse 오퍼레이션에 의해 사용됩니다.

UM3 Complex Format The type is used to represent to the groups for attributes of multiple objects or different objects. It is used either as operation parameters that provide UM3 SET or GET related functions about the values of groups among UM3 operations, or by the OperationResponse operation responsible for the response of the corresponding operation.

다음은 상기 UM3 SET, UM3 SET OF, UM3 SEQUENCE, UM3 SEQUENCE OF 타입의 ASN.1 정규표현식을 이용한 정의입니다.

The definition of UM3 SET, UM3 SET OF, UM3 SEQUENCE, and UM3 SEQUENCE OF using ASN.1 regular expressions is as follows.

UM3 SET ::= UM3 SET “{“ “}” |
UM3 SET “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::= “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::=
UM3ComplexType |
UM3ComplexTypeList “,” UM3ComplexType

UM3ComplexType ::=
UM3NamedType |
UM3NamedType OPTIONAL |
UM3NamedType DEFAULT VALUE |
COMPONENT OF UM3Type

UM3NamedType ::=
identifier UM3Type |
UM3Type

UM3Type ::=
UM3 Primitive type |
UM3 Complex type |
UM3 InformationModel Class type

다음은 UM3 SEQUENCE 타입을 ASN.1 정규표현식으로 나타낸 경우입니다.

The definition of UM3 SEQUENCE using ASN.1 regular expressions is as follows.

UM3 SEQUENCE ::= UM3 SEQUENCE “{“ “}” |
UM3 SEQUENCE “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::= “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::=
UM3ComplexType |
UM3ComplexTypeList “,” UM3ComplexType

UM3ComplexType ::=
UM3NamedType |
UM3NamedType OPTIONAL |
UM3NamedType DEFAULT VALUE |
COMPONENT OF UM3Type

UM3NamedType ::=
identifier UM3Type |
UM3Type

UM3Type ::=
UM3 Primitive type |
UM3 Complex type |
UM3 InformationModel Class type

다음은 UM3 SET OF 타입을 ASN.1 정규표현식으로 나타낸 경우입니다.

The definition of UM3 SET OF using ASN.1 regular expressions is as follows.

UM3 SET OF ::= UM3 SET OF UM3Type

다음은 UM3 SEQUENCE OF 타입을 ASN.1 정규표현식으로 나타낸 경우입니다.

The definition of UM3 SEQUENCE OF using ASN.1 regular expressions is as follows.

UM3 SEQUENCE OF ::= UM3 SEQUENCE OF UM3Type

이상 UM3 SET, UM3 SET OF, UM3 SEQUENCE, UM3 SEQUENCE OF 타입을 정의하였습니다. 이상의 UM3 복합형식 타입은 그 형식과 컴포넌트들의 순서가 어떻게 정의되는가에 관한 사항을 명확하게 구분하여 적용할 수 있어야 합니다.

The UM3 SET, UM3 SET OF, UM3 SEQUENCE, and UM3 SEQUENCE OF types are defined above. The definition of types and sequences of UM3 complex format type components should be clearly identified in application.

10. UM3 Service Management Information Model

서비스관리 정보모델은 앞서 기술한 바와 같이 구성관리, 이벤트관리, 서비스수준관리 등을 중점적으로 지원하도록 정의되어 있습니다. 이러한 3가지의 서비스관리 프로세스는 다른 모든 서비스관리 프로세스들의 출발점이므로, 해당 프로세스가 정상적으로 지원될 경우 본 권고안이 정의하는 UM3 프로토콜은 나머지 서비스관리 프로세스들도 정상적으로 지원하게 됩니다.

The service management information model is mainly defined to support configuration management, event management, and service level management as described previously. Since these three types of service level processes are starting points for all other service management processes, other service management processes in UM3 protocol will operate correctly when they are supported.

본 권고안이 정의하는 서비스관리 정보모델은 물리적 자원과 논리적 자원에 대한 모든 구성자산 및 구성자원에 대한 클래스들을 포함합니다. 이는 물리적 자원 및 논리적 자원에 대한 모델링 결과를 뜻합니다. 표 5 는 서비스 관리 정보모델이 정의하는 클래스 타입 및 타입에 대한 클래스 아이디의 구분을 나타내고 있습니다.

The service management information model defined in this recommendation includes all configuration assets and configuration resources related to physical resources and logical resources. These are the modeling results about physical resources and logical resources. Table 5 shows the class types defined by service management information model and classification of class ID for the class type.

앞서 정의한 기본타입들과는 달리 서비스 관리 정보모델의 클래스아이디는 101번부터 지정됩니다. 또한 응용서비스 정보모델의 클래스아이디는 1001 번부터 지정되게 됩니다. 이와는 달리 UM3 프로토콜을 구성하는 각 오퍼레이션들은 0 번부터 22번 까지의 클래스 아이디로 지정되게 됩니다.

Unlike the primitive types defined previously, class ID of the service management information model is assigned from number 101. The class ID of application service information model is assigned from number 1001. On the other hand, each operation configuring UM3 protocol is assigned with class IDs between number 0 and 22.

10.1. UM3 Protocol and Action

본 권고안이 정의하는 클래스들은 각 클래스별로 액션 (Action) 을 갖고 있거나 혹은 상위 클래스로부터 상속받은 액션을 갖게 됩니다. 액션은 객체지향 방법론의 각 클래스별 독립적으로 실행가능한 member function, method 등과 동일한 개념입니다.

The classes defined in this recommendation have actions for each individual class or actions inherited from upper level classes. Action is the same concept as member function or method that can be executed independently by each class in object oriented methodology.

특히 이러한 액션들중 그 활용 빈도가 가장 높은 액션은 Notify 액션입니다. Notify 액션은 미리 지정해둔 조건이 만족될 경우 자동으로 에이전트가 매니저로 이벤트 발생을 알리는 액션입니다. 이는 알람통지 (alarm notification) 등으로도 표현할 수 있습니다. 이러한 Notify 액션을 위해서는 주기적으로 특정 오브젝트의 애트리뷰트 값을 점검하여 주어진 조건과 일치하는지를 확인해야 합니다. 이를 구현측면에서 기술한다면 컴퓨터의 특정 프로세스가 새롭게 생성되어 주기적으로 해당 애트리뷰트의 값을 읽어들인 후, 해당 애트리뷰트의 값에 조건문을 적용하여 조건문이 만족하는지를 확인한 후, 만족한다면 미리 정해둔 규칙 즉, 해당 조건문이 만족하였음에 관한 통지를 받아야하는 매니저로 이러한 사실을 전송하는 과정으로 요약할 수 있습니다. 이러한 새로운 프로세스는 컴퓨터의 프로세스 (process) 혹은 쓰레드 (thread) 로 구현할 수 있습니다.

The most frequently used action among those actions is the Notify action. This is an action that notifies event to manager automatically when certain preset conditions are satisfied. It can be represented as alarm notification. For this kind of Notify action, the attribute value of the specific object should be periodically checked and examined to confirm whether given conditions are satisfied. From the implementation perspective, when a certain computer process is newly created, it reads the corresponding attributes periodically, applies the values of attributes to the conditional statement, and checks whether the condition is satisfied, and that the pre-defined rule and notification status of conditional statement is executed by the corresponding manager. This new process can be implemented as a process or thread in the computer.

표 5-서비스관리 정보모델 클래스 및 타입의 클래스 아이디 [class ID of Service Management Information Model Class and Type]

클래스 혹은 타입	클래스 아이디	비고
Class or Type	**Class ID**	**Note**
AccumulatorSensor	101	
AnalogControl	102	
AnalogSensor	103	
BinaryControl	104	
BinarySensor	105	
CompanyInfo	106	
ControlBase	107	
DeviceMaintenanceScheduleInfo	108	
DeviceOperationalCondition	109	
DeviceOperationStatus	110	
Gateway	111	
InstalledBattery	112	
InstalledCPU	113	
InstalledEnvironment	114	
InstalledHardDrive	115	
InstalledMemory	116	

InstalledSoftware	117
MultiStateControl	118
MultiStateSensor	119
PersonInfo	120
PresentAccumulatorSensorValue	121
PresentAnalogControlValue	122
PresentAnalogSensorValue	123
PresentBatteryValue	124
PresentBinaryControlValue	125
PresentBinarySensorValue	126
PresentGatewayValue	127
PresentMultiStateControlValue	128
PresentMultiStateSensorValue	129
SensorBase	130
SensorMaintenanceScheduleInfo	131
UM3ActionInvoker	132
UM3Base	133
UM3ObjectIndicator	134
UM3ObjectList	135
UM3ObjectListToBeCreated	136
UM3ObjectValueCondition	137
UM3OperationErrorCode	138
UM3TcpIpAddress	139
UM3ProtocolSupportDescription	140
UM3ActionBroker	141
UM3EventReport	142
UM3UnstructuredObject	143

클래스가 갖고 있는 Notify 액션은 UM3-EVENT-REPORT 서비스를 구성하는 오퍼레이션들인 RequestChangeOfAttributeValueReport, RequestChangeOfAttributeGroupValueReport 및 RequestConditionDetectedReport 오퍼레이션에 의해 기동됩니다. 또한 Notify 액션의 기동 후 그 결과는 OperationResponse 오퍼레이션이 아닌 ReportChangeOfAttributeValue, ReportChangeOfAttributeGroupValue 및 ReportConditionDetected 오퍼레이션에 의해 이루어져야 합니다.

Notify action that belongs to a specific class is implemented by RequestChangeOfAttributeValueReport, RequestChangeOfAttributeGroupValueReport, and RequestConditionDetectedReport operations that configure the UM3-EVENT-REPORT service. The result of the Notify action should be obtained by ReportChangeOfAttributeValue, ReportChangeOfAttributeGroupValue, and ReportConditionDetected operations instead of the OperationResponse operation.

10.2. UM3Base Class

앞서 정의한 바와 같이 본 권고안이 정의하는 UM3 프로토콜의 정보모델은 객체지향 개념을 적용하여

정의합니다. UM3Base 클래스는 본 권고안이 정의하는 모든 정보모델 클래스의 최상위 클래스 입니다. 본 권고안이 정의하는 모든 클래스 들은 UM3Base 클래스로부터 그 애트리뷰트들을 상속받게 (inherited) 됩니다.

As shown previously, the information model of UM3 protocol defined in this recommendation is defined using object oriented concept. UM3Base class is the highest level class of all information model classes defined in this recommendation. All calluses defined in this recommendation have attributes inherited from this UM3Base class.

표 6-UM3Base 클래스의 애트리뷰트 구성 [Configuration of attributes of UM3Base class]

애트리뷰트	타입	M/O
Attributes	**Type**	**M/O**
um3ClassIdentifier	UM3ClassIdentifier	M
um3ObjectName	UM3ObjectName	M
um3ObjectNameAlias	UM3ObjectNameAlias	O
um3ObjectDescription	UM3CharacterString	O
um3AttributeList	UM3ObjectList with ObjectIndicator as its element	M

UM3Base 클래스를 상속받는 모든 클래스들은 um3ClassIdentifer 와 um3ObjectName 애트리뷰트를 갖게 됩니다.

다음은 UM3Base 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

All classes inherited from the UM3Base class have um3ClassIdentifier and um3ObjectName attributes.

The following statement is the representation of the UM3Base class in UM3 CLASS DEFINITION SYNTAX.

```
UM3Base UM3 OBJECT CLASS
        DERIVED FROM
            NONE;
        CHARACTERIZED BY
            최상위 베이스클래스, um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분;
            Highest level base class, identified as um3ClassIdentifier and um3ObjectName attributes;

        ATTRIBUTE NAME AS
            um3ClassIdentifier                  um3ClassIdentifier,
            um3ObjectName                       um3ObjectName,
            um3ObjectNameAlias                  um3ObjectNameAlias,
```

um3ObjectDescription um3ObjectDescription,
um3AttributeList um3AttributeList;
ACTION DEFINED AS
Notify
Target 으로 지정된 애트리뷰트에 대해 그 값이 변화할 경우, 변화사실을 Notify 액션을 호출한 환경 즉, UM3ActionBroker 오브젝트로 되돌려 줌, UM3ActionBroker 오브젝트는 변화사실을 인지한 후 해당 애트리뷰트의 값이 조건을 만족하는지를 판단하여 만족할 경우 매니저로 전송하는 등의 임무를 수행;
When the values of the attributes are specified as targets, the changes are returned to the UM3ActionBroker object that requested the Notify action. UM3ActionBroker recognizes the changes, examines whether the changes of attributes satisfy the conditions, and sends the information to the corresponding manager when required;
BEHAVIOR
UM3 프로토콜의 오퍼레이션을 인지하고, 해당 오퍼레이션에 대응하는 ACTION 을 수행 ;
Recognize the operations of UM3 protocol, and execute ACTION associated to the corresponding operation;
;;

이하 본 권고안이 정의하는 모든 클래스 타입들은 UM3Base 클래스로부터 그 애트리뷰트와 ACTION 및 BEHVIOR 등을 상속받는 것으로 정의합니다. 단, UM3 통신서비스 모델의 오퍼레이션은 그 모델링 과정에 있어서 UM3Base 클래스를 직접적으로 상속받지 않는 것으로 정의합니다.

다음은 UM3Base 클래스의 ASN.1 정규표현식에 의한 정의입니다.

All class types defined in this recommendation as shown below are defined to inherit attributes, ACTION, and BEHAVIOR from UM3Base class. The operations of UM3 communication service model do not directly inherit from the UM3Base class in its modeling process in the definition.

The definition of UM3Base class using ASN.1 regular expressions is as follows.

```
UM3Base ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList
}
```

10.2.1. um3ClassIdentifier Attribute

UM3ClassIdentifier 타입의 애트리뷰트이며 해당 클래스가 속한 클래스 아이디를 나타내게 됩니다. 해당 클래스 아이디는 본 권고안이 정의하는 클래스 아이디 값으로 정해져야 하며 임의로 변경될 수 없

습니다.

This is a UM3ClassIdentifier type attribute, which represents the ID of the corresponding class. The class ID should be assigned by the class ID value defined in this recommendation, and it should not be changed without valid reason.

10.2.2. um3ObjectName Attribute

UM3ObjectName 타입의 애트리뷰트이며 해당 클래스가 인스턴스화 되었을 때 클래스 오브젝트의 이름을 그 값으로 갖게됩니다. um3ObjectName 애트리뷰트는 상기 um3ClassIdentifier 로 구분되는 클래스의 오브젝트들에 대해서 유일한 값으로 주어져야 합니다.

This is a UM3ObjectName type attribute, which has the name of class object as its value when the corresponding class is instantiated. The um3ObjectName attribute should have a unique value among the class objects identified by the above um3ClassIdentifier.

10.2.3. um3AttributeList Attribute

um3AttributeList 애트리뷰트는 오브젝트가 생성되어 메모리에 상주하는 동안 해당 오브젝트가 갖고 있는, 혹은 해당 오브젝트를 구성하는 애트리뷰트의 클래스아이디와 오브젝트이름을 갖고 있어야 합니다. 해당 애트리뷰트는 UM3ObjectList 타입의 애트리뷰트이며 UM3ObjectList 타입을 구성하는 엘리먼트는 UM3ObjectIndicator 타입의 엘리먼트들로 구성됩니다.

The um3AttributeList attribute should have the class ID and object name of the class or attributes configuring the corresponding class when the corresponding object is created and resident in memory. The corresponding attribute should be UM3ObjectList type, and the elements of UM3ObjectList type should be UM3ObjectIndicator type.

본 권고안을 읽어감에 있어서 현재 위치에서 한 번도 언급된적이 없는 클래스 타입 혹은 내용 등이 언급된다면, 본 권고안 맨 앞의 차례 혹은 색인 부분을 참고하여 해당 내용 혹은 클래스 타입에 대한 정확한 내용을 바로 확인해야 합니다. 참고로 UM3ObjectList 타입은 10.36 절에 정의되어 있으며, UM3ObjectIndicator 타입은 10.37 절에 정의되어 있습니다.

If there is a type or description of a class that has not been found until this point while you read through this recommendation, then it is recommended to check the accurate contents on the type or description of the corresponding class through the Table of Contents at the beginning of this document or Index at the end. For reference, UM3ObjectList type is defined in Section 10.36 and UM3ObjectIndicator type is defined in Section 10.37.

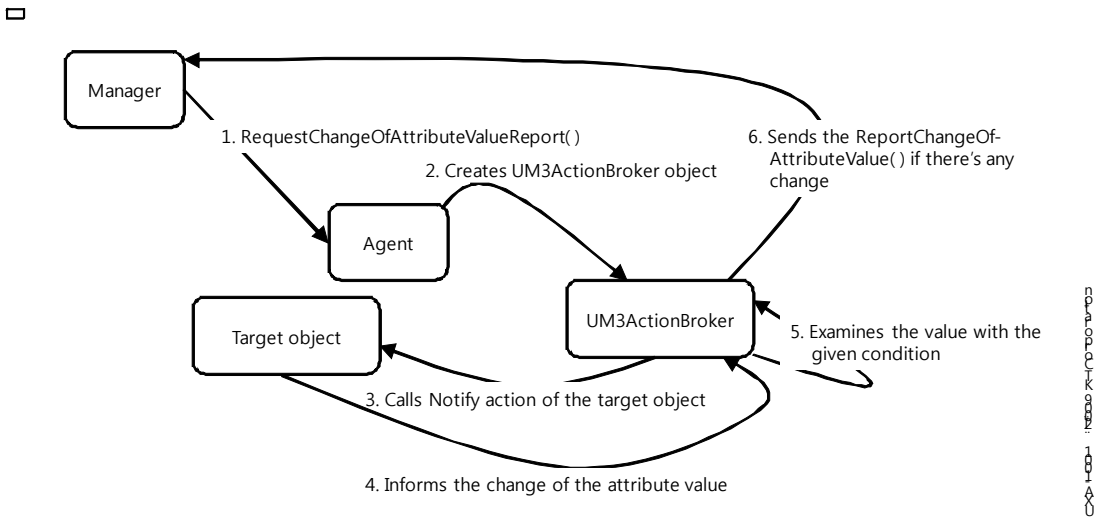


그림 7-Notify 액션의 collaboration diagram [Collaboration diagram of Notify action]

본 권고안은 특정 클래스가 인스턴스화 되어 오브젝트의 형식으로 존재할 때 다이내믹한 애트리뷰트의 생성 혹은 삭제를 허용하지 않습니다. 즉, 애트리뷰트의 생성 혹은 삭제를 지원하는 오퍼레이션을 정의하지 않고 있습니다. 단, 특정 클래스 오브젝트의 애트리뷰트의 타입과 이름을 um3AttributeList 애트리뷰트에 저장하여 관리하고, 매니저의 요청에 있을 경우 해당 정보를 매니저로 전송하거나 혹은 다른 오퍼레이션 및 액션을 위한 참고자료로 활용합니다. um3AttributeList 애트리뷰트는 필수요소로 정의되어 있습니다.

This recommendation does not allow the creation or deletion of dynamic attributes when a class is instantiated and exists as an object. In another words, operations to support creating and deletion of attribute are not defined. The type and name of specific class objects, however, are managed by storing them in the um3AttributeList, and they can be sent to the relevant manager upon request or used for reference

10.2.4. Notify ACTION

Notify 액션은 특정 애트리뷰트의 값에 변화가 있을 경우 자신을 호출하거나 기동 시킨 환경으로 해당 변화 사실을 보고하는 역할을 수행합니다. 대부분의 경우 이러한 액션의 호출 혹은 기동은 UM3ActionBroker 클래스 타입의 오브젝트에 의해 이루어집니다. UM3ActionBroker 클래스 타입의 오브젝트는 이벤트 검출 대상 오브젝트의 애트리뷰트 값을 갱신하기 위한 주기에 관한 정보도 함께 갖고

있으며 이를 감시대상 오브젝트의 애트리뷰트에 미리 적용할 수 있어야 합니다.

Notify action calls itself or reports the changes to the operating environment when there is a change in a specific attribute. Most of the time, these actions or operations take place by UM3ActionBroker class type. Objects of UM3ActionBroker class type also contain the information about the period required to update the attribute values of the target detection objects, and they should be applicable to the attributes of the target monitoring objects.

애트리뷰트 값의 변화를 감지한 UM3ActionBroker 클래스 타입 오브젝트는 자신의 애트리뷰트 값으로 갖고 있는 조건문 등과 변화한 감시 대상 애트리뷰트의 값을 비교하여 그 결과가 조건을 만족하는 경우 매니저에게 ReportXXXXX 관련 오퍼레이션을 송신합니다. 오퍼레이션의 송신은 오퍼레이션의 호출과 동일한 의미이며 이는 UM3 SER 이 정의하는 APDU 의 형태로 매니저에게 전송됩니다.

When an object of UM3ActionBroker class type detects changes in attribute values, it compares the attribute values of the target monitoring object based on the conditions it has as its own attribute values. It then sends the ReportXXXXX related operation of the corresponding manager if the condition is satisfied. The transmission of operation has the same meaning as calling an operation, which sends the information in APDU type defined by UM3 SER.

10.3. Gateway Class

Gateway 는 중앙처리장치, 주기억장치, 보조기억장치, 입출력 포트 등을 갖춘 컴퓨터입니다. Gateway는 사용목적이 일반적인 범용의 컴퓨터와는 달리 AI, AO, DI, DO 등의 포트와 Ethernet, RS 등의 통신 포트를 갖추고, 자신과 연결된 센서, 제어장치 등으로부터 데이터를 수집하여 상위 컴퓨터로 전송하거나, 혹은 상위 컴퓨터로부터 명령을 받아 제어장치로 데이터를 송신하는 등 하위연결자원에 대한 관리대리자 즉, 에이전트의 역할을 담당합니다. 즉, gateway 에 설치 운용되는 소프트웨어가 에이전트의 역할을 수행하는 것으로 정의할 수 있습니다.

A Gateway is a computer with CPU, main memory, auxiliary memory, and IO ports. Unlike general purpose computers, the purpose of a Gateway is to serve as a management agent of lower level connectivity resources, such as collecting data from the sensors connected to the system or control devices, transmitting the data to upper level computers or receiving commands from upper level computers, and transmitting data to control devices using the AI, AO, DI, DO port and communication ports like Ethernet and RS. In other words, it can be defined as the software that is installed and operated in the gateway device to perform the agent function.

Gateway 클래스는 UM3Base 클래스로부터 애트리뷰트와 ACTION 을 상속받는 것으로 정의됩니다. 따라서, UM3Base 클래스로부터 애트리뷰트를 상속받은 Gateway 클래스를 나타내는 표 7 에는 UM3Base 클래스의 애트리뷰트인 um3ClassIdentifier, um3ObjectName, um3ObjectNameAlias 애트리뷰트는 표시하지 않으며 묵시적으로 해당 애트리뷰트들이 포함된것으로 정의합니다. 이하 본 권고안이 정의하는 모든 클래스 타입에 있어서 해당 클래스가 UM3Base 클래스와 상속관계 (inheritance relationship) 에 있다면,

parent class 의 애트리뷰트와 액션은 자동으로 상속되어 갖고 있는 것으로 간주해야 합니다.

The Gateway class inherits attributes and actions from the UM3Base class in its definition. Therefore, Table 7 showing the attributes of the Gateway class that inherits the attributes from the UM3Base class does not contain the UM3Base class attributes such as um3ClassIdentifier, um3ObjectName, and um3ObjectNameAlias, and they are implicitly included by definition. If any class in this recommendation has the inheritance relationship with the UM3Base class, then it should be considered that the attributes and actions of the parent class are inherited automatically.

앞서 정의한 바와 같이 표 7 에는 UM3Base 클래스가 갖고 있는 um3ClassIdentifier, um3ObjectName, um3ObjectNameAlias 등의 애트리뷰트의 표시는 생략합니다. 또한 Gateway 클래스는 UM3Base 클래스가 갖고 있는 모든 애트리뷰트와 액션을 상속 받습니다. 즉, 상기 표 7 에 표기된 애트리뷰트와 더불어 UM3Base 클래스의 애트리뷰트와 액션 관련 특성들을 모두 그대로 상속받아 갖고 있어야 합니다.

As described previously, the attributes of the UM3Base class including um3ClassIdentifier, um3ObjectName, and um3ObjectNameAlias are not shown in Table 7. The Gateway class also inherits all attributes and actions of the UM3Base class, which means that it should have all the attributes and actions inherited from the parent class in addition to the ones shown in Table 7.

표 7-Gateway 클래스의 애트리뷰트 구성 [Attributes of Gateway class]

Attributes	Type	M/O
manufacturerInfo	CompanyInfo	O
vendorInfo	CompanyInfo	O
maintenancePersonel	PersonInfo	O
operationalCondition	DeviceOperationalCondition	O
hasBattery	UM3Boolean	M
batteryInfo	InstalledBattery	O
hasBackupBattery	UM3Boolean	M
backupBatteryInfo	InstalledBattery	O
installedEnvironment	InstalledEnvironment	O
hasCoolingFan	UM3Boolean	O
cpuInfo	InstalledCPU	M
memoryInfo	InstalledMemory	M
um3ProtocolVersion	UM3CharacterString	M
powerConsumption	UM3Real	M
presentBatteryValueInfo	PresentBatteryValue	M
presentBackupBatteryValueInfo	PresentBatteryValue	O
numberOfAnalogInputPort	UM3UnsignedInteger16	M
numberOfAnalogOutputPort	UM3UnsignedInteger16	M
numberOfBinaryInputPort	UM3UnsignedInteger16	M
numberOfBinaryOutputPort	UM3UnsignedInteger16	M
numberOfRS232Port	UM3UnsignedInteger16	M
numberOfRS485Port	UM3UnsignedInteger16	M
numberOfWirelessPort	UM3UnsignedInteger16	M

numberOfEthernetPort	UM3UnsignedInteger16	M
numberOfIRPort	UM3UnsignedInteger16	M
numberOfOpticalPort	UM3UnsignedInteger16	M
doesSupportExternalMemoryCard	UM3Boolean	M
externalMemoryCardType	UM3CharacterString	O
analogInputPortDescription	UM3CharacterString	O
analogOutputPortDescription	UM3CharacterString	O
binaryInputPortDescription	UM3CharacterString	O
binaryOutputPortDescription	UM3CharacterString	O
rs232PortDescription	UM3CharacterString	O
rs485PortDescription	UM3CharacterString	O
wirelessPortDescription	UM3CharacterString	O
ethernetPortDescription	UM3CharacterString	O
irPortDescription	UM3CharacterString	O
opticalPortDescription	UM3CharacterString	O
softwareInfo	InstalledSoftware	M
hasHardDrive	UM3Boolean	M
hardDriveInfo	InstalledHardDrive	O
isDedicatedForOneNode	UM3Boolean	O
levelInAConfigurationTree	UM3UnsignedInteger16	M
numberOfLowerGateway	UM3UngisnedInteger32	M
numberOfLowerNode	UM3UnsignedInteger32	M
upperNodeAddress	UM3TcpIpAddress	M
address	UM3TcpIpAddress	M
doesSupportTelnet	UM3Boolean	M
doesSupportFtp	UM3Boolean	M
analogSensorList	UM3 SEQUENCE OF AnalogSensor	M
binarySensorList	UM3 SEQUENCE OF BinarySensor	M
accumulatedAnalogSensorList	UM3 SEQUENCE OF AccumulatedAnalogSensorList	M
analogControlList	UM3 SEQUENCE OF AnalogControl	M
binaryControlList	UM3 SEQUENCE OF BinaryControl	M
multiStateSensorList	UM3 SEQUENCE OF MultiStateSensor	M
multiStateControlList	UM3 SEQUENCE OF MultiStateControl	M
operationalTimeInfo	DeviceMaintenanceScheduleInfo	M
presentValue	PresentGatewayValue	M
presentGatewayStatus	DeviceOperationStatus	O
maxAPDU	UM3UnsignedInteger32	M
sereialNumber	UM3CharacterString	M

다음은 Gateway 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

Gateway class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
Gateway UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base;
        CHARACTERIZED BY
```

um3ClassIdentifer 와 um3ObjectName 애트리뷰트의 값으로 Gateway 클래스내의 다른 오브젝트들과 구분되며, 하위 레벨의 게이트웨이, 센서노드, 컨트롤노드의 구성 등으로 특징 지워짐;

um3ClassIdentifier and um3ObjectName attribute values are used to identify the object from other objects in the Gateway class, and it is characterized by the configuration of lower level gateway, sensor node, and control node;

ATTRIBUTE NAME AS

manufacturerInfo	manufacturerInfo,
vendorInfo	vendorInfo,
maintenancePersonel	maintenancePersonel,
operationalCondition	operationalCondition,
hasBattery	hasBattery,
batteryInfo	batteryInfo,
hasBackupBattery	hasBackupBattery,
backupBatteryInfo	backupBatteryInfo,
installedEnvironment	installedEnvironment,
hasCoolingFan	hasCoolingFan,
cpuInfo	cpuInfo,
memoryInfo	memoryInfo,
um3ProtocolVersion	um3ProtocolVersion,
powerConsumption	powerConsumption,
presentBatteryValueInfo	presentBatteryValueInfo,
presentBackupBatteryValueInfo	presentBackupBatteryValueInfo,
numberOfAnalogInputPort	numberOfAnalogInputPort,
numberOfAnalogOutputPort	numberOfAnalogOutputPort,
numberOfBinaryInputPort	numberOfBinaryInputPort,
numberOfBinaryOutputPort	numberOfBinaryOutputPort,
numberOfRS232Port	numberOfRS232Port,
numberOfRS485Port	numberOfRS485Port,
numberOfWirelessPort	numberOfWirelessPort,
numberOfEthernetPort	numberOfEthernetPort,
numberOfIRPort	numberOfIRPort,
numberOfOpticalPort	numberOfOpticalPort,
doesSupportExternalMemoryCard	doesSupportExternalMemoryCard,
externalMemoryCardType	externalMemoryCardType,
analogInputPortDescription	analogInputPortDescription,
analogOutputPortDescription	analogOutputPortDescription,
binaryInputPortDescription	binaryInputPortDescription,
binaryOutputPortDescription	binaryOutputPortDescription,
rs232PortDescription	rs232PortDescription,
rs485PortDescription	rs485PortDescription,
wirelessPortDescription	wirelessPortDescription,
ethernetPortDescription	ehternetPortDescription,
irPortDescription	irPortDescription,
opticalPortDescription	opticalPortDescription,
softwareInfo	softwareInfo,
hasHardDrive	hasHardDrive,
hardDriveInfo	hardDriveInfo,

isDedicatedForOneNode isDedicatedForOneNode,
levelInAConfigurationTree levelInAConfigurationTree,
numberOfLowerGateway numberOfLowerGateway,
numberOfLowerNode numberOfLowerNode,
upperNodeAddress upperNodeAddress,
address address,
doesSupportTelnet doesSupportTelnet,
doesSupportFtp doesSupportFtp,
analogSensorList analogSensorList,
binarySensorList binarySensorList,
accumulatedAnalogSensorList accumulatedAnalogSensorList,
analogControlList analogControlList,
binaryControlList binaryControlList,
multiStateSensorList multiStateSensorList,
multiStateControlList multiStateControlList,
operationalTimeInfo operationalTimeInfo,
presentValue presentValue,
presentGatewayStatus presentGatewayStatus,
maxAPDU maxAPDU,
sereialNumber sereialNumber;

ACTION DEFINED AS
NONE;
BEHAVIOR
UM3 프로토콜의 오퍼레이션을 인지하고, 해당 오퍼레이션에 대응하는 ACTION 을 수행하며, 하위레벨의 게이트웨이와 노드들을 관리하는 역할을 수행함;
It recognizes the operations of UM3 protocol, performs ACTION for the corresponding operation, and manages lower level gateway and nodes;
;;

상기 UM3 CLASS DEFINITION SYNTAX 로 표현한 부분에도 UM3Base 클래스로부터 상속받은 애트리뷰트와 액션은 별도로 명기하지 않습니다. Gateway 클래스는 UM3Base 클래스로부터 상속받은 애트리뷰트와 액션을 모두 갖고 있는 것으로 간주해야 합니다.

다음은 UM3Base 클래스의 ASN.1 정규표현식에 의한 정의입니다.

Attributes and actions inherited from the UM3Base class are not specified separately in the above statements expressed by UM3 CLASS DEFINITION SYNTAX. The Gateway class should be considered as the one with all attributes and actions inherited from the UM3Base class.

The Gateway class can be defined by using ASN.1 regular expressions as follows.

Gateway ::= UM3 CLASS {
&um3ClassIdentifier UM3ClassIdentifier,
&um3ObjectName UM3ObjectName,
&um3ObjectNameAlias UM3CharacterString OPTIONAL,

&um3ObjectDescription	**UM3CharacterString OPTIONAL,**
&um3AttributeList	**UM3ObjectList,**
&manufacturerInfo	**CompanyInfo OPTIONAL,**
&vendorInfo	**CompanyInfo OPTIONAL,**
&maintenancePersonel	**PersonInfo OPTIONAL,**
&operationalCondition	**DeviceOperationalCondition OPTIONAL,**
&hasBattery	**UM3Boolean,**
&batteryInfo	**InstalledBattery OPTIONAL,**
&hasBackupBattery	**UM3Boolean,**
&backupBatteryInfo	**InstalledBattery OPTIONAL,**
&installedEnvironment	**InstalledEnvironment OPTIONAL,**
&hasCoolingFan	**UM3Boolean OPTIONAL,**
&cpuInfo	**InstalledCPU,**
&memoryInfo	**InstalledMemory,**
&um3ProtocolVersion	**UM3CharacterString,**
&powerConsumption	**UM3Real,**
&presentBatteryValueInfo	**PresentBatteryValue,**
&presentBackupBatteryValueInfo	**PresentBatteryValue OPTIONAL,**
&numberOfAnalogInputPort	**UM3UnsignedInteger16,**
&numberOfAnalogOutputPort	**UM3UnsignedInteger16,**
&numberOfBinaryInputPort	**UM3UnsignedInteger16,**
&numberOfBinaryOutputPort	**UM3UnsignedInteger16,**
&numberOfRS232Port	**UM3UnsignedInteger16,**
&numberOfRS485Port	**UM3UnsignedInteger16,**
&numberOfWirelessPort	**UM3UnsignedInteger16,**
&numberOfEthernetPort	**UM3UnsignedInteger16,**
&numberOfIRPort	**UM3UnsignedInteger16,**
&numberOfOpticalPort	**UM3UnsignedInteger16,**
&doesSupportExternalMemoryCard	**UM3Boolean,**
&externalMemoryCardType	**UM3CharacterString OPTIONAL,**
&analogInputPortDescription	**UM3CharacterString OPTIONAL,**
&analogOutputPortDescription	**UM3CharacterString OPTIONAL,**
&binaryInputPortDescription	**UM3CharacterString OPTIONAL,**
&binaryOutputPortDescription	**UM3CharacterString OPTIONAL,**
&rs232PortDescription	**UM3CharacterString OPTIONAL,**
&rs485PortDescription	**UM3CharacterString OPTIONAL,**
&wirelessPortDescription	**UM3CharacterString OPTIONAL,**
&ethernetPortDescription	**UM3CharacterString OPTIONAL,**
&irPortDescription	**UM3CharacterString OPTIONAL,**
&opticalPortDescription	**UM3CharacterString OPTIONAL,**
&softwareInfo	**InstalledSoftware,**
&hasHardDrive	**UM3Boolean,**
&hardDriveInfo	**InstalledHardDrive OPTIONAL,**
&isDedicatedForOneNode	**UM3Boolean OPTIONAL,**
&levelInAConfigurationTree	**UM3UnsignedInteger16,**
&numberOfLowerGateway	**UM3UngisnedInteger32,**
&numberOfLowerNode	**UM3UnsignedInteger32,**
&upperNodeAddress	**UM3TcpIpAddress,**
&address	**UM3TcpIpAddress,**
&doesSupportTelnet	**UM3Boolean,**
&doesSupportFtp	**UM3Boolean,**

```
        &analogSensorList                UM3 SEQUENCE OF AnalogSensor,
        &binarySensorList                UM3 SEQUENCE OF BinarySensor,
        &accumulatedAnalogSensorList     UM3 SEQUENCE OF AccumulatedAnalogSensorList,
        &analogControlList               UM3 SEQUENCE OF AnalogControl,
        &binaryControlList               UM3 SEQUENCE OF BinaryControl,
        &multiStateSensorList            UM3 SEQUENCE OF MultiStateSensor,
        &multiStateControlList           UM3 SEQUENCE OF MultiStateControl,
        &operationalTimeInfo             DeviceMaintenanceScheduleInfo,
        &presentValue                    PresentGatewayValue,
        &presentGatewayStatus            DeviceOperationStatus OPTIONAL,
        &maxAPDU                         UM3UnsignedInteger32,
        &sereialNumber                   UM3CharacterString
}
```

상기 ASN.1 정규표현식에 의한 표현에 있어서 앞의 표와 CLASS DEFINITION SYNTAX 에서의 표현과는 달리 UM3Base 클래스로부터의 상속관계를 별도로 표시하지 않고 직접 UM3Base 클래스의 모든 애트리뷰트와 액션들을 함께 표시합니다.

Unlike the earlier table and statement using CLASS DEFINITION SYNTAX, the inheritance relationship with the UM3Base class is not specified separately in ASN.1 regular expression. All attributes and actions of the UM3Base class are specified together.

10.3.1. manufactureInfo Attribute

게이트웨이 혹은 RTU 장치의 제조사에 관한 정보가 기록됩니다. 해당 애트리뷰트는 CompanyInfo 클래스 타입이며 CompanyInfo 클래스의 상세 정의는 10.4 절의 CompanyInfo 클래스의 정의를 참조해야 합니다.

This contains the information about the manufacturer of gateway or RTU device. The corresponding attribute is of CompanyInfo class type. Refer to Section 10.4 Definition of CompanyInfo Class for detailed information about the CompanyInfo class.

10.3.2. vendorInfo Attribute

게이트웨이 장치를 공급한 공급사에 관한 정보를 갖고 있습니다.

This contains the information about the vendor that supplies the gateway device.

10.3.3. maintenancePersonel Attribute

유지보수를 위한 관리자 혹은 인력의 정보가 기록되어 있습니다.

This contains the information about the manager or personnel responsible for maintenance.

10.3.4. operationalCondition Attribute

게이트웨이 장치의 정상적인 운영을 위한 운영조건에 관한 정보를 갖고 있습니다. 해당 애트리뷰트는 비교대상을 위한 기준값과 같은 역할을 합니다. 현재 운영중인 환경에 관한 정보는 installedEnvironment 등의 애트리뷰트를 참조해야 합니다.

This contains the information about operating conditions for the normal operation of gateway device. This attribute becomes a reference for the comparison target. Refer to other attributes like installedEnvironment for the information about the current environment of operation.

10.3.5. hasBattery Attribute

배터리 장착 여부를 나타내는 애트리뷰트입니다. UM3Boolean 타입의 애트리뷰트이며 배터리가 장착되었을 경우 그 값을 TRUE 로 나타냅니다.

This indicates the status of battery installation. This attribute is of UM3Boolean type, and the value becomes TRUE when a battery is installed.

10.3.6. batteryInfo Attribute

배터리 관련 정보를 기록해 두기 위한 애트리뷰트입니다.

This attribute is for storing the information related to the battery.

10.3.7. hasBackupBattery Attribute

메인 배터리 (main battery) 이외에 백업용 보조 배터리의 장착 유무를 나타내는 애트리뷰트입니다. 백업용 보조 배터리를 장착하고 있을 경우에는 그 값을 TRUE 로 기록합니다.

This attribute indicates whether there is a backup battery in addition to the main battery. The value should be TRUE when an aux battery for backup is installed.

10.3.8. backupBatteryInfo Attribute

백업용 보조 배터리의 관련 정보를 기록해두기 위한 애트리뷰트입니다.

This attribute is for storing the information related to the aux battery for backup.

10.3.9. installedEnvironment Attribute

게이트웨이 장치가 설치된 현장의 설치환경 정보를 갖고 있습니다. 애트리뷰트는 InstalledEnvironment

클래스 타입이며 해당 클래스 타입의 상세 정보는 10.8 절을 참조해야 합니다.

This contains the information about the installation environment where the gateway device is installed. It is InstalledEnvironment class type. Refer to section 10.8 for detailed information about the class type.

10.3.10. hasCoolingFan Attribute

게이트웨이를 구성하는 부품 (component) 들 중 냉각팬의 장착 여부를 나타냅니다. 냉각팬이 장착되어 있을 경우 해당 애트리뷰트의 값은 TRUE 로 기록됩니다.

This contains the installation status of the cooling fan, one of the components of the gateway device. The value of this attribute becomes TRUE if a cooling fan is installed.

10.3.11. cpuInfo Attribute

게이트웨이 장치에 장착된 중앙처리장치 (CPU) 의 정보를 기록합니다.

This contains the information about the CPU device installed in the gateway device.

10.3.12. memoryInfo Attribute

게이트웨이 장치에 장착된 주기억장치 (main memory) 의 정보를 기록합니다.

This contains the information about the main memory installed in the gateway device.

10.3.13. um3ProtocolVersion Attribute

게이트웨이가 인식하고 응답할 수 있는 즉, 지원하는 UM3 프로토콜의 버전번호를 기록합니다. 본 권고안의 발행번호와 일치하며, 본 권고안의 발행번호는 2009.11.DRAFT 입니다.

This contains the version number of the supported UM3 protocol that the gateway device can recognize and respond. It is the same as the issue number of this recommendation, and the issue number of this recommendation is 2009.11.DRAFT.

10.3.14. powerConsumption Attribute

게이트웨이 장치의 정격 소모 전력을 나타내며 그 단위는 Watt 입니다.

This indicates the rated power consumption of the gateway device, and the unit is in watts.

10.3.15. presentBatteryValueInfo Attribute

메인 배터리의 현재 상태 정보를 나타내는 애트리뷰트 입니다. 현재 배터리의 잔량 등을 확인하기 위해서는 해당 애트리뷰트 클래스 오브젝트의 애트리뷰트 값을 참조해야 합니다.

This indicates the current status of the main battery. The attribute value of the corresponding attribute class object should be referred to check the battery's remaining capacity.

10.3.16. presentBackupBatteryValueInfo Attribute

백업용 보조 배터리의 현재 상태 정보를 나타내는 애트리뷰트 입니다. 현재 배터리의 잔량 등을 확인하기 위해서는 해당 애트리뷰트 클래스 오브젝트의 애트리뷰트 값을 참조해야 합니다.

This indicates the current status of the aux battery for backup. The attribute value of the corresponding attribute class object should be referred to check the remaining capacity of the battery.

10.3.17. numberOfAnalogInputPort Attribute

게이트웨이에 장착된 AI (analog input) 포트의 개수를 나타냅니다. 해당 애트리뷰트의 값이 0 이라 할지라도 게이트웨이와 연결된 AnalogSensorNode 의 개수는 0 개 이상일 수 있음에 유의해야 합니다. 즉, 센서노드 혹은 제어노드와 게이트웨이는 유무선통신망을 통해 연결되어 있을 수도 있습니다.

This indicates the number of AI (analog input) ports installed in the gateway device. Caution is required when the value of the corresponding attribute is 0, as the number of AnalogSensorNode connected to the gateway device is not necessarily 0 and could be higher than 0. In other words, the sensor node or control node can be connected to the gateway device through a wired and wireless network.

10.3.18. numberOfAnalogOutputPort Attribute

게이트웨이에 장착된 AO (analog output) 포트의 개수를 나타냅니다. 해당 애트리뷰트의 값이 0 이라 할지라도 게이트웨이와 연결된 AnalogSensorNode 의 개수는 0 개 이상일 수 있음에 유의해야 합니다.

This indicates the number of AO (analog output) ports installed in the gateway device. Caution is required when the value of the corresponding attribute is 0 because the number of AnalogSensorNode connected to the gateway device is not necessarily 0 and could be higher than 0. In other words, a sensor node or control node can be connected to the gateway device through a wired and wireless network.

10.3.19. numberOfBinaryInputPort Attribute

게이트웨이에 장착된 BI (binary output) 포트의 개수를 나타냅니다. 해당 애트리뷰트의 값이 0 이라 할지라도 게이트웨이와 연결된 BinarySensorNode 의 개수는 0 개 이상일 수 있음에 유의해야 합니다.

This indicates the number of BI (binary input) ports installed in the gateway device. Caution is required when the value of the corresponding attribute is 0 because the number of BinarySensorNode connected to the gateway device is not necessarily 0 and could be higher than 0.

10.3.20. numberOfBinaryOutputPort Attribute

게이트웨이에 장착된 BO (binary output) 포트의 개수를 나타냅니다. 해당 애트리뷰트의 값이 0 이라 할지라도 게이트웨이와 연결된 BinarySensor 의 개수는 0 개 이상일 수 있음에 유의해야 합니다.

This indicates the number of BO (binary output) ports installed in the gateway device. Caution is required when the value of the corresponding attribute is 0 because the number of BinarySensor connected to the gateway device is not necessarily 0 and could be higher than 0.

10.3.21. numberOfRS232Port Attribute

게이트웨이에 장착된 RS232 포트의 개수를 나타냅니다.

This indicates the number of RS232 ports installed in the gateway device.

10.3.22. numberOfRS485Port Attribute

게이트웨이에 장착된 RS485 포트의 개수를 나타냅니다.

This indicates the number of RS485 ports installed in the gateway device.

10.3.23. numberOfWirelessPort Attribute[1]

게이트웨이에 장착된 무선통신 포트의 개수를 나타냅니다.

This indicates the number of wireless communication ports installed in the gateway device.

10.3.24. numberOfEthernetPort Attribute

게이트웨이 장치에 장착된 이더넷 포트의 개수를 나타냅니다.

This indicates the number of Ethernet ports installed in the gateway device.

[1] 해당 애트리뷰트는 그 필요성이 확인될 경우 Wi-Fi, HSDPA, WCDMA 등으로 더욱 세분화되어 개정할 수 있습니다.

10.3.25. numberOfIRPort Attribute

게이트웨이 장치에 장착된 IR 포트의 개수를 나타냅니다.

This indicates the number of IR ports installed in the gateway device.

10.3.26. numberOfOpticalPort Attribute

게이트웨이 장치에 장착된 광통신포트의 개수를 나타냅니다.

This indicates the number of optical communication ports installed in the gateway device.

10.3.27. doesSupportExternalMemoryCard Attribute

외부 메모리카드와의 데이터 교환을 지원하는지 여부에 관한 값을 나타냅니다. 그 값이 TRUE 일 경우 해당 게이트웨이는 외부메모리 장착이 가능한 상태임을 나타냅니다.

This indicates whether the data exchange with external memory card can be supported. External memory can be installed in the corresponding gateway device if this value is TRUE.

10.3.28. externalMemoryCardType Attribute

상기 doesSupportExternalMemoryCard 애트리뷰트의 값이 TRUE 일 경우 해당 메모리 카드의 종류를 나타내는 애트리뷰트입니다. 그 값으로는 “SD”, “MiniSD” 등이 있을 수 있습니다.

When the above doesSupportExternalMemoryCard attribute value is TRUE, this attribute indicates the type of the memory card. Value could be “SD”, “MiniSD”, etc.

10.3.29. analogInputPortDescription Attribute

게이트웨이에 장착된 AI 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of AI ports installed in the gateway device in UM3CharacterString type.

10.3.30. analogOutputPortDescription Attribute

게이트웨이에 장착된 AO 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of AO ports installed in the gateway device in UM3CharacterString type.

10.3.31. binaryInputPortDescription Attribute

게이트웨이에 장착된 BI 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of BI ports installed in the gateway device in UM3CharacterString type.

10.3.32. binaryOutputPortDescription Attribute

게이트웨이에 장착된 BO 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of BO ports installed in the gateway device in UM3CharacterString type.

10.3.33. rs232PortDescription Attribute

게이트웨이에 장착된 RS232 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of RS232 ports installed in the gateway device in UM3CharacterString type.

10.3.34. rs485PortDescription Attribute

게이트웨이에 장착된 RS485 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of RS485 ports installed in the gateway device in UM3CharacterString type.

10.3.35. wirelessPortDescription Attribute

게이트웨이에 장착된 무선통신포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of wireless communication ports installed in the gateway device in UM3CharacterString type.

10.3.36. ehternetPortDescription Attribute

게이트웨이에 장착된 이더넷 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of Ethernet ports installed in the gateway device in UM3CharacterString type.

10.3.37. irPortDescription Attribute

게이트웨이에 장착된 적외선 포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of IR ports installed in the gateway device in UM3CharacterString type

10.3.38. opticalPortDescription Attribute

게이트웨이에 장착된 광통신포트의 사양 등에 관한 설명을 UM3CharacterString 타입으로 기록합니다.

This contains the description about the specification of optical communication ports installed in the gateway device in UM3CharacterString type.

10.3.39. softwareInfo Attribute

게이트웨이의 운영체제 (OS), 응용소프트웨어, 펌웨어 (firmware) 등의 정보를 기록합니다.

This contains the information about the operating system, application software, and firmware of the gateway device.

10.3.40. hasHardDrive Attribute

하드디스크의 장착 여부를 나타냅니다. 게이트웨이가 하드디스크를 장착하고 있을 경우 해당 애트리뷰트의 값은 TRUE 로 기록됩니다.

This indicates the status of hard disk installation. The value of this attribute becomes TRUE when a hard disk is installed in the gateway device.

10.3.41. hardDriveInfo Attribute

장착된 하드디스크의 정보를 기록하기 위한 애트리뷰트입니다.

This contains the information about the installed hard disk.

10.3.42. isDedicatedForOneNode Attribute

게이트웨이 장치가 하나의 센서 혹은 컨트롤러 즉 제어장치만을 위한 장치인지를 나타냅니다. 해당 게이트웨이가 하나의 센서 혹은 제어장치를 위한 게이트웨이인 경우, 해당 애트리뷰트의 값은 TRUE 로 기록됩니다.

This indicates whether the gateway device is for only one sensor or controller device. The value of this attribute becomes TRUE when the gateway device is for only one sensor or control device.

대부분의 경우 게이트웨이와 센서 혹은 원격 제어장치는 별도의 장치로 구현되며 특히 게이트웨이와 센서 혹은 게이트웨이와 제어장치는 게이트웨이의 AI, AO, DI, DO 포트와 연결되게 됩니다. 이는 센서 혹은 제어장치를 제어하기 위한 회로가 장착되어 있지 않은 경우로 볼 수 있습니다. 그러나, 하나의 회로기판위에 센서소자 혹은 제어장치와 해당 노드를 제어하기 위한 회로가 함께 구현되어 있을 경우에는 이를 하나의 센서소자 혹은 제어장치를 위한 게이트웨이와 해당 센서 혹은 제어장치가 연결되어 있는 것으로 모델링 합니다.

In most cases, the gateway and sensor or remote control device are implemented as separated devices, and the sensor or control device are connected to the gateway device through the AI, AO, DI, or DO ports of the gateway device. In other words, the circuit to control the sensor or control device are not included. The sensor or control device as well as the circuit to control the corresponding node, however, can be installed on one board. This attribute is to model the gateway for only one sensor device of control device or when the gateway and corresponding sensor or control device are hard wired.

그림 8 에는 isDedicatedForOneNode 애트리뷰트의 의미가 그림으로 표현되어 있습니다. 우측의 그림은 게이트웨이 즉, 외부시스템과의 연결 혹은 연동을 위한 게이트웨이와 센서소자가 하나의 보드위에 함께 장착되어 있는 경우이며 이런 경우 isDedicatedForOneNode 애트리뷰트는 TRUE 가 됩니다.

Fig. 8 is the graphical explanation of the isDedicatedForOneNode attribute. The picture shown on the right side is the gateway system shows that gateway device and sensor devices are integrated on one board, and the value of isDedicatedForOneNode attribute is TRUE in this case.

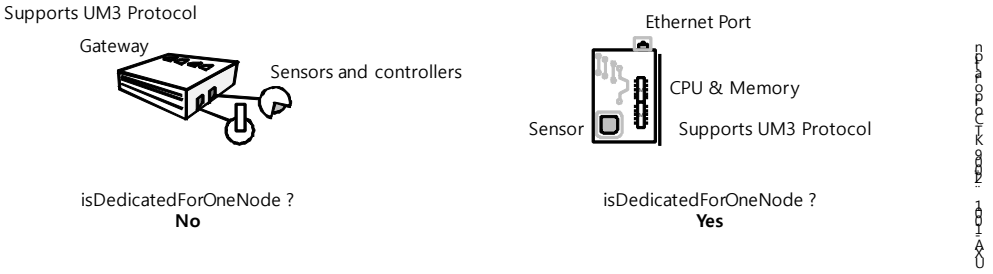


그림 8-isDedicatedForOneNode 애트리뷰트의 의미 [Explanation of isDedicatedForOneNode]

10.3.43. levelInAConfigurationTree Attribute

본 권고안이 정의하는 configuration tree 는 궁극적으로 information tree, containment tree 와 동일한 개념으로 볼 수 있습니다. Configuration tree 는 하나의 서비스를 제공하기 위해 시스템을 구성하는 모든 서비스 자원, 특히 물리적이고도 논리적인 자원들을 최상위 레벨로부터 순서대로 나열한 tree 형태의 구조로 정의합니다. 이는 그림 10-Containment tree 와 UM3 RDN 및 UM3 DN [의 containment tree 와 동일한 개념으로 볼 수 있습니다.

levelInAConfigurationTree 애트리뷰트는 해당 게이트웨이가 configuration tree 를 구성하는 레벨 들 중 몇 번째 레벨에 속해있는가를 나타내는 애트리뷰트입니다.

The configuration tree defined in this recommendation can be considered to be the fundamentally same concept as the information tree or containment tree. The configuration tree defines all service resources configuring the system providing a service in a tree type listing, especially physical and logical resources, in order from the top level ones. It can be considered to be the same concept as the containment tree in Fig. 10 as follows.

The levelInAConfigurationTree attribute indicates to which level the corresponding gateway device belongs in the configuration tree.

10.3.44. numberOfLowerGateway Attribute

해당 게이트웨이의 하위 레벨에 게이트웨이가 연결되어 있을 경우, 그 하위 게이트웨이의 개수가 몇 개인지를 나타내는 애트리뷰트입니다.

This indicates the number of lower level gateway devices when the corresponding gateway device is connected to lower level gateway devices.

10.3.45. numberOfLowerNode Attribute

해당 게이트웨이의 하위 레벨에 존재하는 모든 자원들 즉, 게이트웨이, 센서노드, 제어노드, 네트워크 장비 등 모든 물리적 자원들의 개수를 나타냅니다. 해당 numberOfLowerNode 애트리뷰트의 값에서 numberOfLowerGateway 애트리뷰트의 값을 뺄 경우 현재 게이트웨이의 하위 레벨의 자원들 중 게이트웨이를 제외한 나머지 자원들의 개수를 알 수 있습니다. 또한 대부분의 경우 게이트웨이 하위 레벨의 물리적 자원들은 센서노드 혹은 제어노드로 구성됩니다.

This indicates the number of all physical resources that exist in the lower level of the corresponding gateway device such as gateway, sensor node, control node, and network equipment. Subtracting the value of numberOfLowerGateway attribute from the value of numberOfLowerNod attribute will give the number of all lower level resources except the gateway. In most cases, the physical resources in the lower level of the gateway are sensor nodes or control nodes.

10.3.46. upperNodeAddress Attribute

UM3TcpIpAddress 클래스 타입의 애트리뷰트이며, 상위노드인 게이트웨이 혹은 서버 컴퓨터와의 TCP/IP 통신을 위해 필요한 IP Address, Port 번호 등과 같은 주소 정보를 갖고 있습니다.

This attribute is UM3TcpIpAddress class type, and it contains the information required for TCP/IP communication with a gateway device or server computer in the upper level node such as IP address and port number.

10.3.47. address Attribute

게이트웨이 장치의 IP Address 정보를 갖고 있습니다.

This contains the IP address of the gateway device.

10.3.48. doesSupportTelnet Attribute

해당 게이트웨이 장치가 TCP/IP 기반의 Telnet 프로토콜을 지원하지의 여부를 나타냅니다. Telnet 기능을 지원할 경우 해당 애트리뷰트의 값은 TRUE 로 지정됩니다.

This indicates whether the corresponding gateway devices support TCP/IP based Telnet protocol. The value of this attribute is TRUE if Telnet function is supported.

10.3.49. doesSupportFtp Attribute

해당 게이트웨이 장치가 TCP/IP 기반의 응용 프로토콜인 ftp 통신을 지원하는지의 여부를 나타냅니다. ftp 기능을 지원할 경우 해당 애트리뷰트의 값은 TRUE 로 지정됩니다.

This indicates whether the corresponding gateway devices support TCP/IP based ftp protocol. The value of this attribute is TRUE if ftp function is supported.

10.3.50. analogSensorList Attribute

해당 게이트웨이의 configuration tree 혹은 information tree 에서의 레벨을 *n* 이라한다면, 게이트웨이 오브젝트는 레벨 *(n + 1)* 에서의 AnalogSensor 의 오브젝트들을 aggregation relationship 으로 갖고 있으며, 그 오브젝트들의 게이트웨이 메모리 상에서의 주소 값을 갖고 있는 애트리뷰트입니다.

If the level in the configuration tree or information tree of the corresponding gateway device is *n*, then the gateway object has the aggregation relationship with AnalogSensor objects at the level *(n + 1)*. This attribute contains the address value of those objects in the gateway memory.

analogSensorList 애트리뷰트의 타입은 UM3 SEQUENCE OF 입니다. 즉, C, C++, C#, Java 등의 프로그래밍 언어를 사용할 경우 해당 타입은 linked list 의 형태로 구현됩니다. 그러나, 본 권고안은 해당 애트리뷰트의 구현 방법 혹은 프로그래밍 언어에 제한을 두지 않습니다. 다만, 정확하게 UM3 SEQUENCE OF 타입의 의미를 설명할 경우, 해당 analogSensorList 애트리뷰트의 값이 인코딩 되어 통신 케이블을 타고 전송될 때의 형상은 AnalogSensor 타입의 오브젝트들의 값이 순서대로 인코딩되어 차례 차례 나열되어 있다는 의미로 볼 수 있습니다. 즉, UM3 프로토콜 권고안은 데이터를 송신하거나 수신하기 위한 방법을 정의하기 위해 클래스 타입과 애트리뷰트 등을 정의하는 것일 뿐, 해당 클래스나 애트리뷰트가 컴퓨터에서 어떤 방식으로 구현되어야 하는 가에 관해서는 제한을 두지 않습니다.

The analogSensorList attribute is UM3 SEQUENCE OF type, and it is implemented as linked list type when programming languages like C, C++, C#, or Java are used. This recommendation, however, does not put any restriction on implementation method of the corresponding attribute or programming language. A more accurate description of the UM3 SEQUENCE OF type is that when the values of attributes of the corresponding analogSensorList are encoded and transmitted over the communication cable, the values of the objects of AnalogSensor type are encoded in order, one by one, and listed in sequence. UM3 protocol defines class types and attributes to define the method to send or receive date, and it does not put restriction on the method of implementation of classes and attributes in target computers.

10.3.51. binarySensorList Attribute

analogSensor 애트리뷰트의 경우와 마찬가지로 현재의 게이트웨이 하위레벨에 연결되어 있는 BinarySensor 의 오브젝트들을 그 값으로 갖고 있는 애트리뷰트 입니다. UM3 SEQUENCE OF 타입으로 정의됩니다.

This attribute has BinarySensor objects connected to the lower level of the current gateway similar to the analogSensor attribute. It is defined with UM3 SEQUENCE OF type.

10.3.52. accumulatedAnalogSensorList Attribute

현재 게이트웨이 하위 레벨에 연결되어 있는 AccumulatedAnalogSensor 의 오브젝트들을 갖고 있는 애트리뷰트 입니다.

This contains AccumulatedAnalogSensor objects connected to the lower level of the current gateway device.

10.3.53. analogControlList Attribute

현재 게이트웨이 하위 레벨에 연결되어 있는 AnalogControl 의 오브젝트들을 갖고 있는 애트리뷰트 입니다.

This contains AnalogControl objects connected to the lower level of the current gateway device.

10.3.54. binaryControlList Attribute

현재 게이트웨이 하위 레벨에 연결되어 있는 BinaryControl 의 오브젝트들을 갖고 있는 애트리뷰트 입니다.

This contains BinaryControl objects connected to the lower level of the current gateway device.

10.3.55. multiStateSensorList Attribute

현재 게이트웨이 하위 레벨에 연결되어 있는 MultiStateSensor 의 오브젝트들을 갖고 있는 애트리뷰트 입니다.

This contains MultiStateSensor objects connected to the lower level of the current gateway device.

10.3.56. multiStateControlList Attribute

현재 게이트웨이 하위 레벨에 연결되어 있는 MultiStateControl 의 오브젝트들을 갖고 있는 애트리뷰트 입니다.

This contains MultiStateControl objects connected to the lower level of the current gateway device.

10.3.57. operationalTimeInfo Attribute

서비스 운영과 관련된 기간, 시간 등의 정보를 기록합니다.

This contains the information related to service operation including period and time.

10.3.58. presentValue Attribute

게이트웨이 장치의 현재의 상태 값을 나타내는 애트리뷰트입니다. presentValue 는 PresentGatewayValue 클래스 타입의 애트리뷰트 입니다.

This indicates the current status of the gateway device. The presentValue is an attribute of PresentGatewayValue class type.

10.3.59. presentGatewayStatus Attribute

게이트웨이 장치에 대한 서비스 제공 상태, 점검중 상태 등의 상태를 나타내기 위한 애트리뷰트 입니다.

This indicates the status of service for the gateway device such as in-service or maintenance.

10.3.60. maxAPDU Attribute

게이트웨이가 한 번에 송수신 가능한 UM3 APDU 의 최대 길이 즉, 버퍼 (buffer) 의 크기를 나타냅니다. 이는 일반적인 소켓 프로그래밍에서 read() 혹은 write() 함수를 이용해 소켓으로부터 데이터를 읽거나 쓰기 위해 사용하는, 개발자가 임의로 정할 수 있는 byte array 의 크기를 의미합니다.

This represents the size of buffer that determines the maximum length of UM3 APDU that can be transmitted or received in one attempt. Developers can determine the size of byte array that, and the buffer is used to write or read to and from the socket using the read() or write() function in general socket programming.

10.3.61. sereialNumber Attribute

게이트웨이 장치의 시리얼 번호를 기록하는 애트리뷰트입니다. 시리얼 번호의 형식은 제조사 혹은 공급사에 따라 달라질 수 있으며, 본 권고안이 정의하는 UM3 프로토콜은 serialNumber 애트리뷰트의 타입을 UM3CharacterString 타입으로 정의합니다.

This contains the serial number of the gateway device. The format of the serial number can change depending on the manufacturer or supplier, and the type of serialNumber attribute in UM3 protocol of this recommendation is defined with UM3CharacterString type.

10.3.62. ACTION

Gateway 클래스가 상속받는 UM3Base 클래스의 액션을 그대로 갖고 있습니다.

This contains actions of the UM3Base class that the Gateway class inherits.

10.4. CompanyInfo Class

CompanyInfo 클래스는 회사명, 연락처 등의 애트리뷰트로 구성됩니다. CompanyInfo 클래스는 UM3Base 클래스로부터 그 애트리뷰트와 액션을 상속 받습니다.

The CompanyInfo class contains the company name and contact as its attributes. The CompanyInfo class inherits the attributes and actions for the UM3Base class.

표 8-CompanyInfo 클래스의 애트리뷰트 구성 [Configuration of attributes of CompanyInfo class]

Attribute	Type	M/O
name	UM3CharacterString	M
phoneNumber	UM3CharacterString	M

faxNumber	UM3CharacterString	O
email	UM3CharacterString	M
address	UM3CharacterString	O

CompanyInfo 클래스의 UM3 CLASS DEFINITION SYNTAX 로의 표현은 다음과 같습니다.

CompanyInfo class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
CompanyInfo UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며, name 애트리뷰트로
        특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by the name attribute;

    ATTRIBUTE NAME AS
        Name                        name,
        phoneNumber                 phoneNumber,
        faxNumber                   faxNumber,
        Email                       email,
        address                     address;
    ACTION DEFINED AS
        None;
    BEHAVIOR
        회사관련 정보를 저장;
        This contains company related information;
;;
```

다음은 UM3Base 클래스의 ASN.1 정규표현식에 의한 정의입니다.

CompanyInfo can be defined as follows by using ASN.1 regular expressions as follows.

```
CompanyInfo ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &name                       UM3CharacterString,
```

```
        &phoneNumber                UM3CharacterString,
        &faxNumber                  UM3CharacterString OPTIONAL,
        &email                      UM3CharacterString,
        &address                    UM3CharacterString OPTIONAL
}
```

10.4.1. name Attribute

회사의 이름을 나타냅니다.

This contains name of the company.

10.4.2. phoneNumber Attribute

회사의 전화번호를 나타냅니다.

This contains the telephone number of the company.

10.4.3. faxNumber Attribute

회사의 팩스 번호를 나타냅니다.

This contains the fax number of the company.

10.4.4. email Attribute

회사의 대Table 이메일 주소 정보를 나타냅니다.

This contains the representative e-mail address information of the company.

10.4.5. address Attribute

회사의 주소정보를 나타냅니다.

This contains the address of the company.

10.4.6. ACTION

CompanyInfo 클래스에서 별도로 정의된 액션은 없습니다.

There is no action separately defined in the CompanyInfo class.

10.5. PersonInfo Class

PersonInfo 클래스는 사람의 정보를 저장하고 관리하기 위한 클래스이며 주로 연락처 등을 중심으로 그 정보가 저장됩니다. 또한 PersonInfo 클래스는 모든 애트리뷰트들이 UM3CharacterString 으로 정의되어 있습니다.

본 권고안이 정의하고 있는 여러 클래스 타입들 중 PersonInfo 와 같은 클래스 오브젝트의 정보는 주로 서버측에 저장되어 유지관리되며 특별한 경우가 아닌 이상 게이트웨이에 그 정보가 저장되는 경우는 없는 것으로 정의합니다. 이는 앞서 정의한 CompanyInfo 클래스도 마찬가지 입니다. 단, 이러한 구현 방식은 하나의 예일 뿐이며 각각의 상황에 따라 다른 방식으로 구현할 수 있음을 밝혀둡니다.

The PersonalInfo class is for storing and managing the information of a person, and it is mainly used for contact information. All attributes of the PersonalInfo class are defined with UM3CharacterString.

Among the various class types defined in this recommendation, the information of class objects like PersonalInfo is stored and managed from the server side, and they should not be stored in the gateway device unless otherwise specified. The same rule applies to the CompanyInfo class that was defined earlier. This implementation, however, is only one example, and the actual implementation can be different depending on the situation.

표 9-PersonInfo 클래스의 애트리뷰트 구성 [Attributes of PersonInfo class]

Attribute	Type	M/O
name	UM3CharacterString	M
phoneNumber	UM3CharacterString	O
cellPhoneNumber	UM3CharacterString	M
email	UM3CharacterString	M
address	UM3CharacterString	O
companyName	UM3CharacterString	O

다음은 PersonInfo 클래스에 대한 UM3 CLASS DEFINITION SYNTAX 로의 정의입니다.

PersonInfo class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PersonInfo UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분하며, name 과
            cellPhoneNumber 애트리뷰트로 특징지워짐;
```

```
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and it
        is characterized by the name attribute;
    ATTRIBUTE NAME AS
        name                                name,
        phoneNumber                         phoneNumber,
        cellPhoneNumber                     cellPhoneNumber,
        email                               email,
        address                             address,
        companyName                         companyName;
    ACTION DEFINED AS
        None;
    BEHAVIOR
        인물관련 연락처 정보를 저장;
        This contains contact information related to a person;
;;
```

다음은 UM3Base 클래스의 ASN.1 정규표현식에 의한 정의입니다.

PersonInfo can be defined as follows by using ASN.1 regular expressions as follows.

```
PersonInfo ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &name                           UM3CharacterString,
        &phoneNumber                    UM3CharacterString OPTIONAL,
        &cellPhoneNumber                UM3CharacterString,
        &email                          UM3CharacterString,
        &address                        UM3CharacterString OPTIONAL,
        &companyName                    UM3CharacterString OPTIONAL
}
```

10.5.1. name Attribute

사람의 이름을 저장하기 위한 애트리뷰트 입니다.

This attribute contains the name of a person.

10.5.2. phoneNumber Attribute

유선전화 번호를 저장하기 위한 애트리뷰트입니다.

This attribute contains the phone number of a person.

10.5.3. cellPhoneNumber Attribute

휴대전화번호를 저장하기 위한 애트리뷰트 입니다.

This attribute contains the mobile phone number of a person.

10.5.4. email Attribute

이메일 정보를 저장하기 위한 애트리뷰트 입니다.

This attribute contains the e-mail information of a person.

10.5.5. address Attribute

주소정보를 저장하기 위한 애트리뷰트 입니다.

This attribute contains address information.

10.5.6. companyName Attribute

사람이 속한 회사이름을 저장하기 위한 애트리뷰트 입니다.

This attribute contains the name of the company to which a person belongs.

10.5.7. ACTION

PersonInfo 클래스에는 별도로 정의된 액션이 없습니다.

There is no action separately defined in the PersonInfo class.

10.6. DeviceOperationalCondition Class

특정 장치가 정상적으로 동작하기 위해 필요한 환경에 관한 정보를 갖고 있는 클래스입니다. 특히 게이트웨이, 센서노드, 컨트롤노드, 컴퓨터 등의 정상적인 동작을 위한 정보를 저장합니다.

This class that contains information about the environment required for normal operation of the devices. It particularly contains the information for normal operation of the gateway device, sensor node, control node, and computer.

표 10-DeviceOperationalCondition 클래스의 애트리뷰트 구성 [Attributes of DeviceOperationalCondition class]

Attribute	Type	M/O
temperatureMax	UM3Real	M
temperatureMin	UM3Real	M
ratedVoltage	UM3UnsignedInteger32	M
powerConsumption	UM3UnsignedInteger32	M
humidityMax	UM3UnsignedInteger16	O

다음은 DeviceOperationalCondition 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The DeviceOperationalCondition class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
DeviceOperationalCondition UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며, 습도를 제외한 4개의
            애트리뷰트 값으로 특징 지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized by four attributes excluding humidity;

        ATTRIBUTE NAME AS
            temperatureMax                  temperatureMax,
            temperatureMin                  temperatureMin,
            ratedVoltage                    ratedVoltage,
            powerConsumption                powerConsumption,
            humidityMax                     humidityMax;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            특정 장치의 정상동작을 위한 환경정보를 저장;
            This contains environment information required for normal operation of a specific device.
;;
```

다음은 UM3Base 클래스의 ASN.1 정규표현식에 의한 정의입니다.

DeviceOperationalCondition can be defined as follows by using ASN.1 regular expressions.

```
DeviceOperationalCondition ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &temperatureMax                 UM3Real,
        &temperatureMin                 UM3Real,
        &ratedVoltage                   UM3UnsignedInteger32,
        &powerConsumption               UM3UnsignedInteger32,
        &humidityMax                    UM3UnsignedInteger16 OPTIONAL
}
```

10.6.1. temperatureMax Attribute

정상동작을 보장하는 허용 온도의 상한값을 나타냅니다. 단위는 °C 입니다.

This contains the upper limit of the temperature that guarantees normal operation. The unit is °C.

10.6.2. temperatureMin Attribute

정상동작을 보장하는 허용 온도의 하한값을 나타냅니다. 단위는 °C 입니다.

This contains the lower limit of the temperature that guarantees normal operation. The unit is °C.

10.6.3. ratedVoltage Attribute

정격전압을 나타냅니다. 단위는 V (Volt) 입니다.

This contains the rated voltage. The unit is V (Volt).

10.6.4. powerConsumption Attribute

정격소비전력을 나타냅니다. 단위는 W (Watt) 입니다.

This contains the rated power consumption. The unit is W (Watt).

10.6.5. humidityMax Attribute

정상동작을 위한 허용 습도의 상한값을 나타냅니다. 단위는 % 입니다.

This contains the upper limit of humidity that guarantees normal operation. The unit is %.

10.6.6. ACTION

DeviceOperationalCondition 클래스에는 별도로 정의된 액션이 없습니다.

There is no action separately defined in the DeviceOperationalCondition class.

10.7. InstalledBattery Class

배터리의 사양 정보를 저장하기 위한 클래스입니다. 배터리는 주로 게이트웨이, 센서노드 혹은 컨트롤 노드에 장착되는 배터리를 대상으로 합니다. 그러나 경우에 따라 센서 혹은 컨트롤러 뿐만 아니라 서버, 네트워크 장비 등과 같은 컴퓨팅 자원에 장착되어 있는 모든 배터리가 InstalledBattery 클래스로 모델링 될 수 있습니다. 또한 이러한 형태의 배터리는 자동차, 모바일 디바이스 등 거의 모든 형태의 배터리의 모델링에 활용될 수 있습니다.

This class is for storing battery specification information. Target batteries are mainly the ones used in the gateway device, sensor node, and control node. Any batteries installed in the computing resources like server or network equipment as well as sensor and controller can be modeled with InstalledBattery class. It can also be used to model any type of batteries including automobile and mobile devices.

표 11-InstalledBattery 클래스의 애트리뷰트 구성 [Attributes of InstalledBattery]

Attribute	Type	M/O
temperatureMax	UM3Real	O
temperatureMin	UM3Real	O
isBackupBattery	UM3Boolean	O
isChargable	UM3Boolean	O
ratedVoltage	UM3UnsignedInteger32	M
ratedCurrent	UM3Real	M
type	UM3CharacterString	O
size	UM3CharacterString	M
weight	UM3Real	O
presentVoltage	UM3Real	M
remainingBatteryTime	UM3DateTime	O

다음은 InstalledBattery 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The InstalledBattery class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

InstalledBattery UM3 OBJECT CLASS

DERIVED FROM
UM3Base;
CHARACTERIZED BY
um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며, type 애트리뷰트의 값으로 특징지워짐;
um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and they are characterized by type attribute;
ATTRIBUTE NAME AS

temperatureMax	temperatureMax,
temperatureMin	temperatureMin,
isBackupBattery	isBackupBattery,
isChargable	isChargable,
ratedVoltage	ratedVoltage,
ratedCurrent	ratedCurrent,
type	type,
size	size,
weight	weight,
presentVoltage	presentVoltage,
remainingBatteryTime	remainingBatteryTime;

ACTION DEFINED AS
None;
BEHAVIOR
배터리의 사양과 관련된 정보를 저장;
This contains information related to battery specification.
;;

다음은 InstalledBattery 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The InstalledBattery class can be defined as follows by using ASN.1 regular expressions.

InstalledBattery ::= UM3 CLASS {

&um3ClassIdentifier	**UM3ClassIdentifier,**
&um3ObjectName	**UM3ObjectName,**
&um3ObjectNameAlias	**UM3CharacterString OPTIONAL,**
&um3ObjectDescription	**UM3CharacterString OPTIONAL,**
&um3AttributeList	**UM3ObjectList,**
&temperatureMax	**UM3Real OPTIONAL,**
&temperatureMin	**UM3Real OPTIONAL,**
&isBackupBattery	**UM3Boolean OPTIONAL,**
&isChargable	**UM3Boolean OPTIONAL,**
&ratedVoltage	**UM3UnsignedInteger32,**
&ratedCurrent	**UM3Real,**
&type	**UM3CharacterString OPTIONAL,**
&size	**UM3CharacterString,**
&weight	**UM3Real OPTIONAL,**

```
        &presentVoltage                 UM3Real,
        &remainingBatteryTime           UM3DateTime OPTIONAL
}
```

10.7.1. temperatureMax Attribute

정상동작을 보장하는 허용 온도의 상한값을 나타냅니다. 단위는 °C 입니다.

This contains the upper limit of the temperature that guarantees normal operation. The unit is °C.

10.7.2. temperatureMin Attribute

정상동작을 보장하는 허용 온도의 하한값을 나타냅니다. 단위는 °C 입니다.

This contains the lower limit of the temperature that guarantees normal operation. The unit is °C.

10.7.3. isBackupBattery Attribute

해당 배터리가 백업용 보조배터리인지의 여부를 나타냅니다. 해당 배터리가 백업용 보조 배터리일 경우 해당 애트리뷰트의 값은 TRUE 로 저장됩니다.

This indicates whether the current battery is an aux battery for backup. The value of this attribute is TRUE when the corresponding battery is an aux battery for backup.

10.7.4. isChargable Attribute

충전이 가능한 충전지 여부를 나타냅니다. 해당 애트리뷰트의 값이 TRUE 일 경우 충전지임을 나타내게 됩니다.

This indicates whether the battery is rechargeable. The corresponding battery is rechargeable when the value of this attribute is TRUE.

10.7.5. ratedVoltage Attribute

정격전압 혹은 공칭전압을 나타내며 그 단위는 V (Volt) 입니다.

This contains the rated or nominal voltage. The unit is V (Volt).

10.7.6. ratedCurrent Attribute

정격전류 혹은 용량을 나타내며 그 단위는 mAh 입니다.

This contains the rated current or capacity. The unit is mAh.

10.7.7. type Attribute

충전소자의 종류에 따른 배터리의 종류를 나타냅니다. 예를 들어 니켈카드뮴 배터리의 경우 해당 애트리뷰트의 값은 “NiCd” 가 저장됩니다.

This indicates the type of battery based on the material used for storing energy. For example, “NiCd” is stored in this attribute for Nickel Cadmium batteries.

10.7.8. size Attribute

길이, 폭, 높이 등 배터리의 크기를 UM3CharacterString 타입의 정보로 저장합니다.

The size of battery including length, width, and height is stored as UM3CharacterString type.

10.7.9. weight Attribute

배터리의 중량 혹은 무게를 나타내며 그 단위는 g (gram) 입니다.

This contains the weight of the battery. The unit is g (gram).

10.7.10. presentVoltage Attribute

현재 배터리의 출력전압을 나타내며 그 단위는 V (Volt) 입니다.

This contains the output voltage of the current battery. The unit is V (Volt).

10.7.11. remainingBatteryTime Attribute

배터리의 사용가능 시간을 나타내며 그 단위는 sec 입니다.

This contains the remaining battery time. The unit is sec.

10.7.12. ACTION

InstalledBattery 클래스에서 별도로 정의된 액션은 없습니다.

There is no action separately defined in the InstalledBattery class.

10.8. InstalledEnvironment Class

InstalledEnvironment 클래스는 특정 장치가 설치된 환경정보를 나타내기 위해 사용됩니다. 실외 혹은 실내, 설치 장소 등 오프라인에서 설치된 장치 주변의 환경정보를 직접 입력하여 필요할 때 해당 정보에 접근하기 위해 사용하는 클래스입니다.

The InstalledEnvironment class is used to store information about environment where a specific device is installed. It is used for entering information about the environment where equipment is installed such as indoor, outdoor, or installation location, and to use such information when necessary.

표 12-InstalledEnvironment 클래스의 애트리뷰트 구성 [Attributes of InstalledEnvironment]

Attribute	Type	M/O
isIndoor	UM3Boolean	M
hasEnclosure	UM3Boolean	M
isWaterproof	UM3Boolean	M
altitude	UM3Integer16	O
isRackMounted	UM3Boolean	O
size	UM3CharacterString	O
pointDescription	UM3CharacterString	O
installedDate	UM3DateTime	M

다음은 InstalledEnvironment 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

InstalledEnvironment class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
InstalledEnvironment UM3 OBJECT CLASS
    DERIVED FROM
        None;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 주로 installedDate 로
        특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are mainly characterized by the installedDate attribute;
    ATTRIBUTE NAME AS
        isIndoor                    isIndoor,
        hasEnclosure                hasEnclosure,
        isWaterproof                isWaterproof,
        Altitude                    altitude,
        isRackMounted               isRackMounted,
```

Size size,
pointDescription pointDescription,
installedDate installedDate;
ACTION DEFINED AS
None;
BEHAVIOR
특정장치의 설치 환경 및 주변환경 정보를 저장;
This contains the information about the installation environment and surrounding environment of specific equipment.
;;

다음은 InstalledEnvironment 클래스의 ASN.1 정규표현식에 의한 정의입니다.

InstalledEnvironment class can be defined as follows by using ASN.1 regular expressions.

```
InstalledEnvironment ::= UM3 CLASS {
        &um3ClassIdentifier          UM3ClassIdentifier,
        &um3ObjectName               UM3ObjectName,
        &um3ObjectNameAlias          UM3CharacterString OPTIONAL,
        &um3ObjectDescription        UM3CharacterString OPTIONAL,
        &um3AttributeList            UM3ObjectList,
        &isIndoor                    UM3Boolean,
        &hasEnclosure                UM3Boolean,
        &isWaterproof                UM3Boolean,
        &altitude                    UM3Integer16 OPTIONAL,
        &isRackMounted               UM3Boolean OPTIONAL,
        &size                        UM3CharacterString OPTIONAL,
        &pointDescription            UM3CharacterString OPTIONAL,
        &installedDate               UM3DateTime
}
```

10.8.1. isIndoor Attribute

특정 장치의 설치 장소가 실내인지 실외인지를 나타내는 애트리뷰트입니다. 해당 애트리뷰트의 값이 TRUE 인 경우, 설치장소는 실내임을 나타내게 됩니다.

This attribute indicates whether the installation location is indoors or outdoors. The installation is indoors if the value of this attribute is TRUE.

10.8.2. hasEnclosure Attribute

특정 장치를 보호하기 위한 함체가 설치되어 있는지를 나타내기 위한 애트리뷰트입니다. 해당 애트리

뷰트의 값이 TRUE 인 경우, 장치는 함체 내부에 설치되어 있음을 뜻합니다.

This attribute indicates whether an enclosure to protect specific device is installed. If the value of this attribute is TRUE, then it means that the device is installed inside a protective enclosure.

10.8.3. isWaterProof Attribute

방수 기능의 유무를 나타내는 애트리뷰트이며, 그 값이 TRUE 일경우에는 방수가능을 의미하며 FALSE 일 경우에는 방습가능을 의미합니다. 상황에 따라 방수와 방습이 모두 지원되지 않는 장치의 경우에는 해당 애트리뷰트의 값을 단순히 FALSE 로 정의하여 나타낼 수도 있습니다.

This indicates whether the device has a waterproof function. The device is waterproof if this value is TRUE, and it is moisture proof if it is FALSE. If a device is neither waterproof or moisture proof, then it can simply be defined as FALSE depending on the situation.

10.8.4. altitude Attribute

장치가 설치된 고도를 나타내며 그 단위는 m (meter) 입니다.

This indicates the altitude of the place where the device is installed. The unit is m (meter).

10.8.5. isRackMounted Attribute

설치된 방법이 랙 마운트 형태로 설치되었는지의 여부를 나타냅니다. 해당 애트리뷰트의 값이 TRUE 일 경우 해당 장치는 랙 마운트 형태로 설치되었음을 뜻하게 됩니다.

This indicates whether it is installed in a rack mount type. When the value of this attribute is TRUE, it means that the corresponding device is rack mounted.

10.8.6. size Attribute

해당 장치의 크기 정보 즉, 가로, 세로, 높이 정보를 UM3CharacterString 타입으로 저장합니다.

This contains the size information of the corresponding device including width, length, and height as UM3CharacterString type.

10.8.7. pointDescription Attribute

위도 및 경도, 장소에 대한 설명 등 설치장소에 관한 정보를 저장합니다.

This contains the information about installation location including latitude, longitude, and description of the location.

10.8.8. installedDate Attribute

장치를 설치한 일시를 나타내며 경우에 따라 서비스를 시작한 일시의 의미로도 사용할 수 있습니다.

This contains the date of installation, and it can be used as the service start date depending on the scenario.

10.8.9. ACTION

InstalledEnvironment 클래스에서 별도로 정의된 액션은 없습니다.

There is no action separately defined in the InstalledEnvironment class.

10.9. InstalledCPU Class

InstalledCPU 클래스는 게이트웨이, 퍼스널컴퓨터, 서버 등의 장치에 장착된 중앙처리장치 (CPU) 의 정보를 기록하기 위한 클래스입니다.

The InstalledCPU class is for specifying the information of the CPU installed in various devices like gateway, personal computer, and server.

표 13-InstalledCPU 클래스의 애트리뷰트 구성 [Attributes of InstalledCPU class]

Attribute	Type	M/O
numberOfCPU	UM3UnsignedInteger16	M
Manufacturer	UM3CharacterString	M
Model	UM3CharacterString	M
Speed	UM3CharacterString	M
busWidth	UM3UnsignedInteger16	M

다음은 InstalledCPU 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

InstalledCPU class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
InstalledCPU UM3 OBJECT CLASS
    DERIVED FROM
        None;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 전체 애트리뷰트의 값
        으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
```

they are characterized by entire attributes;
ATTRIBUTE NAME AS
numberOfCPU numberOfCPU,
Manufacturer manufacturer,
Model model,
Speed speed,
busWidth busWidth;
ACTION DEFINED AS
None;
BEHAVIOR
장착된 CPU 정보를 저장하고 관리;
This stores and manages information of the installed CPU;
;;

다음은 InstalledCPU 클래스의 ASN.1 정규표현식에 의한 정의입니다.

InstalledCPU class can be defined as follows by using ASN.1 regular expressions.

```
InstalledCPU ::= UM3 CLASS {
        &um3ClassIdentifier          UM3ClassIdentifier,
        &um3ObjectName               UM3ObjectName,
        &um3ObjectNameAlias          UM3CharacterString OPTIONAL,
        &um3ObjectDescription        UM3CharacterString OPTIONAL,
        &um3AttributeList            UM3ObjectList,
        &numberOfCPU                 UM3UnsignedInteger16,
        &manufacturer                UM3CharacterString,
        &model                       UM3CharacterString,
        &speed                       UM3CharacterString,
        &busWidth                    UM3UnsignedInteger16
}
```

10.9.1. numberOfCPU Attribute

장치에 장착된 CPU 의 개수를 나타냅니다. 다만, 외형상 한 개의 CPU 에 코어 개념의 소자가 복수로 장착된 CPU 칩에 관한 정보 등은 상황에 따라 1 혹은 4 등으로 적절하게 표현할 수 있습니다.

This indicates the number of CPUs installed in the device. When a CPU has more than one core in one physical device, it can be specified as 1 or 4, depending on the situation.

10.9.2. manufacturer Attribute

CPU 의 제조사 정보를 저장합니다.

This contains the information about the CPU manufacturer.

10.9.3. model Attribute

CPU 의 모델 관련 정보를 기록합니다.

This contains the information related to the CPU model.

10.9.4. speed Attribute

CPU 의 처리속도를 UM3CharacterString 타입으로 정의합니다.

The clock speed of the CPU is defined with UM3CharacterString type.

10.9.5. busWidth Attribute

CPU 의 데이터 처리 단위를 나타냅니다. 그 단위는 bit 이며 UM3UnsignedInteger16 타입으로 나타냅니다.

This indicates the data processing unit of CPU. The unit is bits, and it is specified as UM3UnsignedInteger16 type.

10.9.6. ACTION

InstalledCPU 클래스에서 별도로 정의된 액션은 없습니다.

There is no action separately defined in the InstalledCPU class.

10.10. InstalledMemory Class

장치에 장착되어있는 메모리와 관련된 정보를 기록하고 저장 관리하기 위한 클래스입니다.

This class is for storing and managing the information related to the memory installed in the device.

표 14-InstalledMemory 클래스의 애트리뷰트 구성 [Attributes of InstalledMemory class]

Attribute	Type	M/O
installedMemorySize	UM3UnsignedInteger32	M
maximumExpandableSize	UM3UnsignedInteger32	M
accessTime	UM3Real	O
Manufacturer	UM3CharacterString	O
numberOfMemorySlot	UM3UnsignedInteger16	O

currentMemorySizePerSlot	UM3UnsignedInteger32	O

다음은 InstalledMemory 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

InstalledMemory class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
InstalledMemory UM3 OBJECT CLASS
        DERIVED FROM
            None;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 메모리의 전체용량과
            확장 가능 용량으로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized by the total capacity of memory and extensible capacity;
        ATTRIBUTE NAME AS
            installedMemorySize                 installedMemorySize,
            maximumExpandableSize               maximumExpandableSize,
            accessTime                          accessTime,
            manufacturer                        manufacturer,
            numberOfMemorySlot                  numberOfMemorySlot,
            currentMemorySizePerSlot            currentMemorySizePerSlot;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            장착된 메모리 정보와 확장 가능 정보 등을 저장하고 관리;
            This stores and manages the information of installed memory and extensible capacity
            information.
;;
```

다음은 InstalledMemory 클래스의 ASN.1 정규표현식에 의한 정의입니다.

InstalledMemory class can be defined as follows by using ASN.1 regular expressions.

```
InstalledMemory ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &installedMemorySize            UM3UnsignedInteger32,
        &maximumExpandableSize          UM3UnsignedInteger32,
        &accessTime                     UM3Real OPTIONAL,
```

```
        &manufacturer                   UM3CharacterString OPTIONAL,
        &numberOfMemorySlot             UM3UnsignedInteger16 OPTIONAL,
        &currentMemorySizePerSlot       UM3UnsignedInteger32 OPTIONAL
}
```

10.10.1. installedMemorySize Attribute

설치된 메모리의 용량을 나타내기 위한 애트리뷰트 입니다. 그 단위는 킬로바이트 (Kbyte) 입니다.

This attribute contains the capacity of installed memory. The unit is Kbytes.

10.10.2. maximumExpandableSize Attribute

설치된 메모리의 용량을 나타내기 위한 애트리뷰트 입니다. 그 단위는 킬로바이트 (Kbyte) 입니다.

This attribute contains the maximum expandable size of the memory. The unit is Kbytes.

10.10.3. accessTime Attribute

설치된 메모리의 R/W 속도를 나타내며 그 단위는 마이크로 세컨드 (μS) 입니다.

This indicates the R/W speed of the installed memory. The unit is microseconds (μS).

10.10.4. manufacturer Attribute

설치된 메모리의 제조사 정보를 기록합니다.

This contains the information about manufacturer of installed memory.

10.10.5. numberOfMemorySlot Attribute

장치의 메인보드가 지원하는 메모리 장착 가능 슬롯의 개수를 나타냅니다.

This contains the number of slots in the main board that support memory installation.

10.10.6. currentMemorySizePerSlot Attribute

현재 메모리가 장착되어 사용중인 슬롯의 개수를 나타냅니다.

This indicates the number of slots where memory is currently installed and used.

10.10.7. ACTION

InstalledMemory 클래스에는 별도로 정의된 액션이 없습니다.

There is no action separately defined in the InstalledMemory class.

10.11. PresentBatteryValue Class

PresentBatteryValue 클래스는 장치에 장착된 배터리의 현재 상태값을 나타내기 위해 사용되는 클래스입니다. 주로 배터리의 잔량, 현재 전압의 크기 등의 관한 정보를 저장 관리합니다.

The PresentBatteryValue class is used to indicate the current status of the battery installed in the device. It usually stores information like remaining battery capacity and the battery's current voltage output.

표 15-PresentBatteryValue 클래스의 애트리뷰트 구성 [Attributes of PresentBatteryValue class]

Attribute	Type	M/O
presentVoltage	UM3Real	M
previousVoltage	UM3Real	M
remainingBatteryTime	UM3DateTime	M

다음은 PresentBatteryValue클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentBatteryValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PresentBatteryValue UM3 OBJECT CLASS
    DERIVED FROM
        None;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 배터리의 현재 값과
        잔량 혹은 사용가능 시간으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by the current value of battery, remaining capacity, or useable time;
    ATTRIBUTE NAME AS
        presentVoltage                  presentVoltage,
        previousVoltage                 previousVoltage,
        remainingBatteryTime            remainingBatteryTime;
    ACTION DEFINED AS
        None;
    BEHAVIOR
```

배터리의 현재 상태 값 등을 저장 관리;
This stores and manages the values related to the current status of battery.
;;

다음은 PresentBatteryValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

PresentBatteryValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentBatteryValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentVoltage                 UM3Real,
        &previousVoltage                UM3Real,
        &remainingBatteryTime           UM3DateTime
}
```

10.11.1. presentVoltage Attribute

현재 배터리의 출력 전압을 나타내며 그 단위는 V (Volt) 입니다.

This indicates the instantaneous output voltage of the battery. The unit is V (Volt).

10.11.2. previousVoltage Attribute

현재 배터리의 출력값을 측정하기 직전의 이전 값을 나타내며 그 단위는 V (Volt) 입니다. 상기 presentVoltage 애트리뷰트 값과 previousVoltage 애트리뷰트의 값을 이용하여 배터리 제조사가 제공하지 않는 여러가지 정보의 계산에 필요한 연산을 수행할 수 있습니다.

This contains the value right before measuring the output value of the battery, and the unit is V (Volt). Various information that is not provided by the battery manufacturer can be extracted by using the presentVoltage attribute that is defined above and previousVoltage attribute value.

10.11.3. remainingBatteryTime Attribute

측정 순간부터 배터리가 완전히 방전되어 더 이상 사용할 수 없는 순간까지의 배터리 사용 가능 시간을 나타내며 그 타입은 UM3DateTime 입니다. 해당 애트리뷰트의 값의 단위는 초 (second) 이며 1970 년 1월 1일 0 시를 기준으로 하는 현재시각과는 다른 개념으로 사용됨에 유의해야 합니다.

This indicates the remaining time of the battery from the measurement point until the battery is completely drained, and the type is UM3DateTime. The unit of the value of the attribute is seconds. Caution is required as the concept is different from the current time, which is represented as elapsed time from 12:00 AM, Jan 1st, 1970.

10.11.4. ACTION

PresentBatteryValue 클래스에는 별도로 정의된 액션이 없습니다.

There is no action separately defined in the presentBatteryValue class.

10.12. InstalledSoftware Class

장치에 설치되어 있는 특정 소프트웨어에 관한 정보를 기록 관리하기 위한 클래스 입니다.

This class is for storing and managing the information of specific software installed in the device.

표 16-InstalledSoftware 클래스의 애트리뷰트 구성 [Attributes of InstalledSoftware class]

Attribute	Type	M/O
isOS	UM3Boolean	M
nameOfSoftware	UM3CharacterString	M
version	UM3CharacterString	M
updateDateTime	UM3DateTime	M
numberOfUpdateUpToNow	UM3UnsignedInteger16	O
manufacturer	UM3CharactgerString	O

InstalledSoftware 게이트웨이 장치, 하나의 보드에 게이트웨이와 센서소자가 장착된 경우 혹은 제어장치의 컨트롤러가 장치의 제어를 위한 회로를 포함하고 있는 경우, 해당 장치에 설치된 운영체제 (OS) 혹은 펌웨어 (firmware) 등의 정보를 나타냅니다.

When the gateway and sensor device are integrated in the same board or controller of control device includes the circuit for device control, the InstalledSoftware class contains the information about the operating system (OS) or firmware installed in the corresponding device.

다음은 InstalledSoftware 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The InstalledSoftware class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

InstalledSoftware UM3 OBJECT CLASS

DERIVED FROM
None;
CHARACTERIZED BY
um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 버전번호로 특징지워짐;
um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and they are characterized by version number;
ATTRIBUTE NAME AS

isOS	isOS,
nameOfSoftware	nameOfSoftware,
version	version,
updateDateTime	updateDatetime,
numberOfUpdateUpToNow	numberOfUpdateUpToNow,
manufacturer	manufacturer;

ACTION DEFINED AS
None;
BEHAVIOR
설치된 소프트웨어 관련 정보를 저장 관리;
This stores and manages the information related to installed software.
;;

다음은 InstalledSoftware 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The InstalledSoftware class can be defined as follows by using ASN.1 regular expressions.

```
InstalledSoftware ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &isOS                           UM3Boolean,
        &nameOfSoftware                 UM3CharacterString,
        &version                        UM3CharacterString,
        &updateDateTime                 UM3DateTime,
        &numberOfUpdateUpToNow          UM3UnsignedInteger16 OPTIONAL,
        &manufacturer                   UM3CharactgerString OPTIONAL
}
```

10.12.1. isOS Attribute

InstalledSoftware 클래스 오브젝트가 갖고 있는 정보가 운영체제 (OS) 에 관한 정보인지, 혹은 일반적인 소프트웨어 및 펌웨어 (firm ware) 에 관한 정보인지를 나타냅니다. UM3Boolean 타입이며 해당 애트

리뷰트의 값이 TRUE 일 경우 운영체제를 나타냅니다.

This indicates whether the information contained in the InstalledSoftware class object is about the version of the operating system (OS), general software, and firmware. It is UM3Boolean type, and it is the version of the operating system when the value of this attribute is TRUE.

10.12.2. nameOfSoftware Attribute

설치된 소프트웨어의 이름을 나타냅니다. 게이트웨이 장치의 경우 “arm-linux”, “micro-c-kernel” 등의 값을 가질 수 있으며 일반적인 컴퓨터의 경우 “HP-UX”, “Solaris”, “Windows server 2008”, “Linux”, “Microsoft Office Word”, “UM3 model communicator” 등의 일반적인 소프트웨어 이름 혹은 제조사가 명명한 이름을 그 값으로 갖게 됩니다.

This contains the name of the installed software. Gateway devices can be values like “arm-linux” or “micro-c-kernel”, and general computers may have general software names or names provided by manufacturers such as “HP-UX”, “Solraris”, “Windows server 2008”, “Linux”, “Microsoft Office Word”, and “UM3 model communicator”.

10.12.3. version Attribute

상기 애트리뷰트로 특징지워지고 구분되는 소프트웨어의 버전 번호를 나타냅니다.

The class is characterized with this attribute, including software version number.

10.12.4. updateDateTime Attribute

가장 최근의 업데이트 일시를 나타냅니다.

This contains the last updated date.

10.12.5. numberOfUpdateUpToNow Attribute

최초 설치 이후 현재까지 업데이트 한 회수를 나타냅니다.

This contains the number updates performed from initial installation until now.

10.12.6. manufacturer Attribute

해당 소프트웨어의 제조사 정보를 나타냅니다.

This contains the information about the manufacturer of the corresponding software.

10.12.7. ACTION

InstalledSoftware 클래스에는 별도로 정의된 액션이 없습니다.

There is no action defined in this class.

10.13. InstalledHardDrive Class

장치 혹은 컴퓨터에 설치된 하드디스크의 정보를 나타내기 위한 클래스 입니다.

This class is used to store the information of the hard disk installed in the device or computer.

표 17-InstalledHardDrive 클래스의 애트리뷰트 구성 [Attributes of InstalledHardDrive class]

Attribute	Type	M/O
diskSize	UM3UnsignedInteger32	M
speed	UM3CharacterString	O
manufacturer	UM3CharacterString	O
diskDiameter	UM3Real	M
hasReplaced	UM3Boolean	M
replacedDateTime	UM3DateTime	M
serialNumber	UM3CharacterString	M

다음은 InstalledHardDrive 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

InstalledHardDrive class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
InstalledHardDrive UM3 OBJECT CLASS
    DERIVED FROM
        None;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 용량으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by capacity;
    ATTRIBUTE NAME AS
        diskSize                    diskSize,
        Speed                       speed,
        manufacturer                manufacturer,
        diskDiameter                diskDiameter,
        hasReplaced                 hasReplaced,
        replacedDateTime            replacedDateTime,
```

serialNumber serialNumber;
ACTION DEFINED AS
None;
BEHAVIOR
설치된 하드디스크 사양 관련 정보를 저장 관리;
This stores and manages the information related to the specification of installed hard disk.
;;

다음은 InstalledHardDrive 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The InstalledHardDrive class can be defined as follows by using ASN.1 regular expressions.

```
InstalledHardDrive ::= UM3 CLASS {
        &um3ClassIdentifier          UM3ClassIdentifier,
        &um3ObjectName               UM3ObjectName,
        &um3ObjectNameAlias          UM3CharacterString OPTIONAL,
        &um3ObjectDescription        UM3CharacterString OPTIONAL,
        &um3AttributeList            UM3ObjectList,
        &diskSize                    UM3UnsignedInteger32,
        &speed                       UM3CharacterString OPTIONAL,
        &manufacturer                UM3CharacterString OPTIONAL,
        &diskDiameter                UM3Real,
        &hasReplaced                 UM3Boolean,
        &replacedDateTime            UM3DateTime,
        &serialNumber                UM3CharacterString
}
```

10.13.1. diskSize Attribute

장착된 하드디스크의 용량을 나타내며 그 단위는 MByte 입니다.

This indicates the capacity of the installed hard disk. The unit is MBytes.

10.13.2. speed Attribute

장착된 하드디스크의 RW 액세스 속도를 나타내며 그 단위는 rpm 입니다.

This indicates the RW access speed of the installed hard disk. The unit is rpm.

10.13.3. manufacturer Attribute

제조사의 정보를 나타냅니다.

This contains the information about manufacturer.

10.13.4. diskDiameter Attribute

하드디스크의 물리적 크기를 나타내며 그 단위는 inch 입니다.

This contains the physical size of the hard disk. The unit is inch.

10.13.5. hasReplaced Attribute

하드디스크의 교체 여부를 나타냅니다.

This indicates the status of hard disk replacement.

10.13.6. replacedDateTime Attribute

하드디스크가 교체 된 적이 있을 경우 교체일시를 나타냅니다.

This contains the date and time when the hard disk was replaced if it was replaced.

10.13.7. serialNumber Attribute

하드디스크의 일련번호를 나타내며 그 타입은 UM3CharacterString 입니다.

This contains the serial number of the hard disk. The type is UM3CharacterString.

10.13.8. ACTION

InstalledHardDrive 클래스에는 별도로 정의되 액션이 없습니다.

There is no action separately defined in the InstalledHardDrive class.

10.14. UM3TcpIpAddress Class

특장 장치의 TCP/IP 통신을 위한 주소와 포트번호 등의 정보를 갖고 있습니다.

This contains the information for TCP/IP communication of specific device such as address and port number.

표 18-UM3TcpIpAddress 클래스의 애트리뷰트 구성 [Attributes of UM3TcpIpAddress class]

Attribute	Type	M/O
ipAddressV4	UM3CharacterString	M

ipAddressV6	UM3CharacterString	O
portNumber	UM3UnsignedInteger16	M
tcpOrUdp	UM3Boolean	M

다음은 UM3TcpIpAddress 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

UM3TcpIpAddress class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
UM3TcpIpAddress UM3 OBJECT CLASS
        DERIVED FROM
            None;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 ip address 와 port 번
            호로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized by ip address and port number;
        ATTRIBUTE NAME AS
            ipAddressV4                         ipAddressV4,
            ipAddressV6                         ipAddressV6,
            portNumber                          portNumber,
            tcpOrUdp                            tcpOrUdp;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            TCP/IP 통신을 위한 주소 및 포트번호 등의 정보를 저장 관리;
            This stores and manages the information for TCP/IP communication such as address and port
            number.
;;
```

다음은 UM3TcpIpAddress 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The UM3TcpIpAddress class can be defined as follows by using ASN.1 regular expressions.

```
UM3TcpIpAddress ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &ipAddressV4                    UM3CharacterString,
```

```
        &ipAddressV6                UM3CharacterString OPTIONAL,
        &portNumber                 UM3UnsignedInteger16,
        &tcpOrUdp                   UM3Boolean
}
```

10.14.1. ipAddressV4 Attribute

IP 버전 4에 대한 IP Address 를 나타냅니다.

This contains IP address for IP version 4.

10.14.2. ipAddressV6 Attribute

IP 버전 6에 대한 IP Address 를 나타냅니다.

This contains IP address for IP version 6.

10.14.3. portNumber Attribute

소켓 통신을 위한 포트번호를 나타냅니다.

This contains the port number for socket communication.

10.14.4. tcpOrUdp Attribute

UM3 프로토콜의 APDU 송수신을 위한 방식이 TCP 방식인지 UDP 방식인지를 나타냅니다. 본 권고안은 모든 UM3 APDU 를 TCP 방식으로 송수신할 것을 권고합니다.

This indicates whether the APDU transmission and reception method of the UM3 protocol is TCP or UDP. This recommendation recommends the TCP method for transmission and reception of UM3 APDU.

10.14.5. ACTION

UM3TcpIpAddress 클래스에서 별도로 정의된 액션은 없습니다.

There is no action separately defined in the UM3TcpIpAddress class.

10.15. SensorBase Class

본 권고안이 정의하는 AnalogSensor, BinarySensor, AccumulatedAnalogSensor, MultistateSensor 등의 베이스 클래스입니다. 주로 구성관리 등을 위한 정보를 기록하고 관리하기 위한 애트리뷰트로 구성되어 있습니다.

The SensorBase class is a base class for AnalogSensor, BinarySensor, AccumulatedAnalogSensor, and MultistateSensor defined in this recommendation. It has the attributes for storing and managing the information, and is mainly for configuration management.

표 19-SensorBase 클래스의 애트리뷰트 구성 [Attributes of SensorBase class]

Attribute	Type	M/O
manufacturerInfo	CompanyInfo	O
vendorInfo	CompanyInfo	O
maintenancePersonel	PersonInfo	O
operationalCondition	DeviceOperationalCondition	O
hasBattery	UM3Boolean	M
batteryInfo	InstalledBattery	O
hasBackupBattery	UM3Boolean	M
backupBatteryInfo	InstalledBattery	O
presentBatteryValueInfo	PresentBatteryValue	M
presentBackupBatteryValueInfo	PresentBatteryValue	O
installedEnvironment	InstalledEnvironment	O
powerConsumption	UM3Real	M
levelInAConfigurationTree	UM3UnsignedInteger16	M
upperGatewayAddress	UM3TcpIpAddress	O
operationalTimeInfo	DeviceMaintenanceScheduleInfo	M
presentSensorStatus	DeviceOperationStatus	M
serialNumber	UM3CharacterString	M

다음은 SensorBase 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

SensorBase class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
SensorBase UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 AnalogSensor,
            BinarySensor 등을 위한 베이스 클래스로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized as a base class for AnalogSensor and BinarySensor;
        ATTRIBUTE NAME AS
            manufacturerInfo                    manufacturerInfo,
            vendorInfo                          vendorInfo,
            maintenancePersonel                 maintenancePersonel,
            operationalCondition                operationalCondition,
```

hasBattery hasBattery,
batteryInfo batteryInfo,
hasBackupBattery hasBackupBattery,
backupBatteryInfo backupBatteryInfo,
presentBatteryValueInfo presentBatteryValueInfo,
presentBackupBatteryValueInfo presentBackupBatteryValueInfo,
installedEnvironment installedEnvironment,
powerConsumption powerConsumption,
levelInAConfigurationTree levelInAConfigurationTree,
upperGatewayAddress upperGatewayAddress,
operationalTimeInfo operationalTimeInfo,
presentSensorStatus presentSensorStatus,
serialNumber serialNumber;

ACTION DEFINED AS
None;

BEHAVIOR
특정 소자로 데이터를 측정하고 이를 결선을 통해 상위 게이트웨이가 수집, 혹은 결선을 통해 상위 게이트웨이가 소자를 제어;
Data is measured with a specific device and collected by an upper level gateway through a cable, or a device is controlled by an upper level gateway through a cable;

;;

다음은 SensorBase 클래스의 ASN.1 정규표현식에 의한 정의입니다.

SensorBase class can be defined as follows by using ASN.1 regular expressions.

```
SensorBase ::= UM3 CLASS {
        &um3ClassIdentifier              UM3ClassIdentifier,
        &um3ObjectName                   UM3ObjectName,
        &um3ObjectNameAlias              UM3CharacterString OPTIONAL,
        &um3ObjectDescription            UM3CharacterString OPTIONAL,
        &um3AttributeList                UM3ObjectList,
        &manufacturerInfo                CompanyInfo OPTIONAL,
        &vendorInfo                      CompanyInfo OPTIONAL,
        &maintenancePersonel             PersonInfo OPTIONAL,
        &operationalCondition            DeviceOperationalCondition OPTIONAL,
        &hasBattery                      UM3Boolean,
        &batteryInfo                     InstalledBattery OPTIONAL,
        &hasBackupBattery                UM3Boolean,
        &backupBatteryInfo               InstalledBattery OPTIONAL,
        &presentBatteryValueInfo         PresentBatteryValue,
        &presentBackupBatteryValueInfo   PresentBatteryValue OPTIONAL,
        &installedEnvironment            InstalledEnvironment OPTIONAL,
        &powerConsumption                UM3Real,
        &levelInAConfigurationTree       UM3UnsignedInteger16,
```

```
        &upperGatewayAddress        UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo        DeviceMaintenanceScheduleInfo,
        &presentSensorStatus        DeviceOperationStatus,
        &serialNumber               UM3CharacterString
}
```

10.15.1. manufacturerInfo Attribute

센서 제조 회사의 정보를 갖고 있습니다.

This contains the information about the sensor manufacturer.

10.15.2. vendorInfo Attribute

센서 공급 회사의 정보를 갖고 있습니다.

This contains the information about the sensor vendor.

10.15.3. maintenancePersonel Attribute

센서 장치에 대한 유지보수 혹은 점검 담당자 정보를 갖고 있습니다.

This contains the information about the personnel responsible for maintenance and monitoring of sensor device.

10.15.4. operationalCondition Attribute

센서의 운용환경에 대한 정보를 갖고 있습니다.

This contains the information about the sensor’s operation environment.

10.15.5. hasBattery Attribute

센서를 위한 별도의 배터리 장치를 갖고 있는지를 나타냅니다. 그 값이 TRUE 일 경우 배터리가 장착되어 있음을 뜻하며, FALSE 일 경우 결선되어 있는 게이트웨이에서 전력을 공급함을 뜻 합니다.

This indicates whether there is a separate battery device for the sensor. When this value is TRUE, it means that a battery is installed. When it is FALSE, it means that power is supplied by the gateway connected by cable.

10.15.6. batteryInfo Attribute

배터리가 장착되어 있을 경우 해당 배터리의 정보를 기록합니다.

This contains the information about the battery when a battery is installed.

10.15.7. hasBackupBattery Attribute

백업용 보조 배터리가 장착되어 있는지의 여부를 나타냅니다.

This indicates whether an aux battery for backup is installed.

10.15.8. backupBatteryInfo Attribute

백업용 보조 배터리의 사양 등 여러가지 정보를 기록합니다.

This contains various information about the specification of the aux battery for backup.

10.15.9. presentBatteryValueInfo Attribute

현재 배터리의 잔량 등 상태정보 값을 갖게 됩니다.

It contains the status information such as remaining battery capacity.

10.15.10. presentBackupBatteryValueInfo Attribute

백업용 보조 배터리의 현재 상태값 들을 갖게 됩니다.

This contains the current value of the aux battery for backup.

10.15.11. installedEnvironment Attribute

센서가 설치된 환경정보를 나타냅니다. 예를 들어 센서의 사이즈 등은 installedEnvironment.size 애트리뷰트의 값을 이용하여 UM3CharacterString 타입으로 정의할 수 있습니다.

This contains the information about the environment where sensor is installed. For example, the size of the sensor can be defined in UM3CharacterString type by using the value of the installedEnvironment.size attribute.

10.15.12. powerConsumption Attribute

센서의 전력소모와 관련된 정보를 기록합니다.

This contains the information related to the sensor’s power consumption.

10.15.13. levelInAConfigurationTree Attribute

configuration tree 에서의 레벨을 나타냅니다. 앞서 정의한 바와 같이 configuration tree 는 information tree 혹은 containment tree 와 동일하거나 유사한 개념으로 사용됩니다.

This indicates the level in the configuration tree. As defined earlier, configuration is used with a similar or the same concept as the information tree of containment tree.

10.15.14. upperGatewayAddress Attribute

비록 센서가 결선으로 연결된 상위 레벨의 게이트웨이와 TCP/IP 기반의 통신을 하지 않는다 하더라도 AnalogSensor 오브젝트는 상위 게이트웨이의 TCP/IP 통신을 위한 주소를 갖고 있습니다. 해당 애트리뷰트는 선택요소 (optional) 로 정의되어 있습니다.

Even when the sensor does not communicate the upper level gateway based on TCP/IP through a cable, an AnalogSensor object has the address of an upper level gateway for TCP/IP communication. The corresponding attribute is defined as an optional field.

10.15.15. operationalTimeInfo Attribute

SensorMaintenanceScheduleInfo 클래스 타입의 애트리뷰트이며 점검주기, 예정 점검일시 등을 나타냅니다.

This is an attribute of the SensorMaintenanceScheduleInfo class type, and it contains the inspection period and scheduled inspection date.

10.15.16. presentSensorStatus Attribute

센서의 현재 상태를 나타냅니다. outOfService 의 경우 센서가 더 이상 센싱 서비스를 제공하지 않는다는 뜻 입니다. inService 는 현재 정상적인 형태로 센서가 동작 중이며 데이터 수집 서비스를 제공하고 있다는 뜻 입니다. inMaintenance 는 현재 유지보수를 위한 점검 중이며 다시 서비스가 재개될 예정임을 뜻 합니다. inFault 는 현재 고장상태이며 다시 정상화 할 계획임을 의미합니다. 결선으로 연결된 센서의 경우 communicationFailure 와 detached 는 동일한 의미로 사용될 수 있습니다. unknownCause 는 원인을 알 수 없는 비정상 상태임을 뜻 하며 해당 비정상 상태를 정상화 시킬 예정임을 그 뜻에 내포하고 있습니다.

This contains the current status of the sensor. When this value is outOfService, it means that no more sensing service is provided. When it is inService, it means that the sensor currently operates normally and provides the data collection service. inMaintenance means that the sensor is currently under maintenance, and service will resume later. inFault means that the sensor is out of order, and it will be corrected again. In case the sensor connected by cable, communicationFailure and detached can be used for same meaning. unknownCause means that the sensor is not working correctly due to an unknown reason, and it implies that the failure will be fixed.

10.15.17. serialNumber Attribute

시리얼 번호와 관련된 정보를 갖고 있는 애트리뷰트 입니다.

This attribute contains the information related to the serial number.

10.15.18. ACTION

SensorBase 클래스에서 별도로 다시 정의된 액션은 없습니다.

There is no action separately defined in the SensorBase class.

10.16. AnalogSensor Class

애널로그 방식의 센서를 나타냅니다. 본 권고안이 정의하는 애널로그 센서는 그 출력값이 0 혹은 1 로 구분되는 binary 방식과는 달리 애널로그 방식의 출력값을 갖는 장치를 의미합니다. 애널로그 출력값은 그 타입을 UM3Real, UM3IntegerXX, UM3UnsignedIntegerXX 타입으로 정의할 수 있습니다. 애널로그 출력값은 전압 혹은 전류와 같은 전기적 신호일 경우가 많으며 이는 analog input, analog output 단자와 연결됩니다.

This class is used for analog sensors. An analog sensor is defined as a device that has analog output value, unlike the binary type that produces 0 or 1 as output value in this recommendation. The type of analog output can be defined with UM3Real, UM3IntegerXX, or UM3UnsignedIntegerXX. Quite often than not, the analog output value represents an electrical signal like voltage or current, and they are connected to the analog input or analog output terminal.

본 권고안이 정의하는 센서노드 (sensor node) 는 센싱소자 (sensing component) 가 통신기능을 갖춘 전기적인 회로와 연결되어 스스로 데이터를 수신하거나 송신할 수 있는 경우를 뜻합니다. 즉, 전기적인 회로와 연결된 1 개 이상의 센서들의 집합을 센서노드로 정의합니다.

A sensor node defined in this recommendation is a sensing component that is connected to an electric circuit with communication function, and it can receive or transmit data autonomously. In other words, a set of one or more sensors that are connected electrically is defined as a sensor node.

그러나, 스스로 데이터를 송수신할 수 없는 즉, 공용 혹은 전용 게이트웨이 장치와의 전기적인 연결을 통해서만 데이터를 수신하거나 송신할 수 있는 센서 혹은 1 개 이상의 센서들을 본 권고안은 센서 혹은 센서의 집합으로 정의합니다. 이 때 게이트웨이 장치는 간단한 소형의 전기회로기판의 형태일 수도 있으며 일정규모 이상의 컴퓨터와 유사한 구성을 갖춘 게이트웨이와 같은 장치일 수도 있습니다.

One or more sensors that can only receive or transmit data through electrical connection with a shared or dedicated gateway device without the capability of internal data transfer capability are defined as a sensor or set of sensors in

this recommendation. In this case, a gateway device might be in the form of small electronic circuit board or full scale gateway device with a similar configuration as a general purpose computer.

따라서, 여러 개의 AI, AO, DI, DO 포트와 시리얼포트, Zigbee 포트, Ehternet 포트 등을 장착하고 있으며 AI, AO, DI, DO 포트에 연결된 센서를 전압 혹은 전류의 강도를 이용하여 제어하거나 데이터를 가져오는 게이트웨이가 있다면, 해당 게이트웨이와 더불어 게이트웨이에 연결된 모든 센서들을 합하여 하나의 센서노드라 칭하게 됩니다.

Therefore, if there is a gateway device that has an AI, AO, DI, DO port, serial port, Zigbee port, and Ethernet port, and it controls and collects data from the sensors connected to the AI, AO, DI, and DO ports by controlling the voltage or current, then all sensors connected to the gateway along with the corresponding gateway are called a sensor node.

따라서, AnalogSensor 클래스는 게이트웨이 장치와의 연결을 통해 데이터를 송수신하는, 자체적인 통신 기능을 갖고 있지 않은 소자 혹은 장치로 정의합니다.

센서소자에 대한 설명 즉, 센서타입 등의 정보는 UM3Base 클래스로부터 상속받은 um3ObjectDescription 애트리뷰트를 활용해야 합니다. 이는 AnalogSensor 클래스뿐만 아니라 본 권고안이 정의하는 모든 센서 및 컨트롤 관련 클래스에 동일하게 적용되어야 합니다.

Therefore, the AnalogSensor class is used to define a device or component that communicates through the connection with a gateway device without internal communication capability.

The um3ObjectDescription attribute inherited from the UM3Base class should be utilized for the description about the sensor component i.e. information about sensor type. It should apply not only to the AnalogSensor class, but also all sensor and control related classes defined in this recommendation.

표 20-AnalogSensor 클래스의 애트리뷰트 구성 [Attributes of AnalogSensor class]

Attribute	Type	M/O
presentValue	PresentAnalogSensorValue	M

다음은 AnalogSensor 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

AnalogSensor class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
AnalogSensor UM3 OBJECT CLASS
        DERIVED FROM
```

UM3Base, SensorBase;
CHARACTERIZED BY
um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 presentValue 애트리뷰트의 값으로 특징지워짐;
um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and they are characterized as the value of the presentValue attribute;
ATTRIBUTE NAME AS
presentValue presentValue;

ACTION DEFINED AS
None;
BEHAVIOR
특정 소자로 데이터를 측정하고 이를 결선을 통해 상위 게이트웨이가 수집, 혹은 결선을 통해 상위 게이트웨이가 소자를 제어;
Data is measured with a specific device and collected by an upper level gateway through a cable, or a device is controlled by an upper level gateway through a cable;

;;

다음은 AnalogSensor 클래스의 ASN.1 정규표현식에 의한 정의입니다.

AnalogSensor class can be defined as follows by using ASN.1 regular expressions.

```
AnalogSensor ::= UM3 CLASS {
    &um3ClassIdentifier              UM3ClassIdentifier,
    &um3ObjectName                   UM3ObjectName,
    &um3ObjectNameAlias              UM3CharacterString OPTIONAL,
    &um3ObjectDescription            UM3CharacterString OPTIONAL,
    &um3AttributeList                UM3ObjectList,
    &manufacturerInfo                CompanyInfo OPTIONAL,
    &vendorInfo                      CompanyInfo OPTIONAL,
    &maintenancePersonel             PersonInfo OPTIONAL,
    &operationalCondition            DeviceOperationalCondition OPTIONAL,
    &hasBattery                      UM3Boolean,
    &batteryInfo                     InstalledBattery OPTIONAL,
    &hasBackupBattery                UM3Boolean,
    &backupBatteryInfo               InstalledBattery OPTIONAL,
    &presentBatteryValueInfo         PresentBatteryValue,
    &presentBackupBatteryValueInfo   PresentBatteryValue OPTIONAL,
    &installedEnvironment            InstalledEnvironment OPTIONAL,
    &powerConsumption                UM3Real,
    &levelInAConfigurationTree       UM3UnsignedInteger16,
    &upperGatewayAddress             UM3TcpIpAddress OPTIONAL,
    &operationalTimeInfo             DeviceMaintenanceScheduleInfo,
    &presentSensorStatus             DeviceOperationStatus,
    &serialNumber                    UM3CharacterString,
    &location                        UM3CharacterString,
```

&presentValue PresentAnalogSensorValue
}

AnalogSensor 클래스는 SensorBase 클래스와 UM3Base 클래스로부터 그 속성을 상속받습니다. 즉, 상기 ASN.1 정규표현식으로 정의된 각 애트리뷰트의 상세정의는 SensorBase 클래스와 UM3Base 클래스를 참조해야 합니다.

The AnalogSensor class inherits the attributes from the SensorBase class and UM3Base class. That means that the SensorBase class and UM3Base class should be referred for the detailed definition of attributes defined with the above ASN.1 regular expression.

10.16.1. presentValue Attribute

센서의 현재 값을 나타내기 위해 사용되는 애트리뷰트 입니다. presentValue 애트리뷰트를 제외한 나머지 애트리뷰트들은 서비스 관리 영역의 애트리뷰트 들로 볼 수 있습니다.

This attribute is used to indicate the current value of the sensor. All other attributes except the presentValue attribute can be considered as the ones for the service management area.

10.16.2. ACTION

AnalogSensor 클래스에서 별도로 정의된 액션은 없습니다.

There is no action separately defined in the AnalogSensor class.

10.17. SensorMaintenanceScheduleInfo Class

서비스의 정상적인 운영을 위한 각종 장치들에 대한 오프라인 혹은 온라인 점검 주기, 점검 예정 시각 등에 관한 정보를 나타내는 클래스입니다.

This class is used to store information required for normal operation of the service such as the offline or online inspection period and scheduled inspection time.

표 21-SensorMaintenanceScheduleInfo 클래스의 애트리뷰트 구성 [Attributes of SensorMaintenanceScheduleInfo class]

Attribute	Type	M/O
inServiceDateTime	UM3DateTime	M
latestMaintenanceDateTime	UM3DateTime	O
nextMaintenanceDateTime	UM3DateTime	O
durablePeriod	UM3DateTime	M
maintenancePeriod	UM3DateTime	O

numberOfRewired	UM3UnsignedInteger16	O
detachScheduleInfo	UM3DateTime	O

다음은 SensorMaintenanceScheduleInfo 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The SensorMaintenanceScheduleInfo class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
SensorMaintenanceScheduleInfo UM3 OBJECT CLASS
        DERIVED FROM
            None;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 점검주기, 점검일자,
            rewire 등의 일정으로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized by inspection period, inspection date, and rewire schedule;
        ATTRIBUTE NAME AS
            inServiceDateTime                   inServiceDateTime,
            latestMaintenanceDateTime           latestMaintenanceDateTime,
            nextMaintenanceDateTime             nextMaintenanceDateTime,
            durablePeriod                       durablePeriod,
            maintenancePeriod                   maintenancePeriod,
            numberOfRewired                     numberOfRewired,
            detachScheduleInfo                  detachScheduleInfo;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            유지보수 등을 위한 주요 일정정보를 저장 관리;
            This stores and manages the major schedule information for maintenance;
;;
```

다음은 SensorMaintenanceScheduleInfo 클래스의 ASN.1 정규표현식에 의한 정의입니다.

SensorMaintenanceScheduleInfo class can be defined as follows by using ASN.1 regular expressions.

```
SensorMaintenanceScheduleInfo ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
```

```
        &um3ObjectDescription          UM3CharacterString OPTIONAL,
        &um3AttributeList              UM3ObjectList,
        &inServiceDateTime             UM3DateTime,
        &latestMaintenanceDateTime     UM3DateTime OPTIONAL,
        &nextMaintenanceDateTime       UM3DateTime OPTIONAL,
        &durablePeriod                 UM3DateTime,
        &maintenancePeriod             UM3DateTime OPTIONAL,
        &numberOfRewired               UM3UnsignedInteger16 OPTIONAL,
        &detachScheduleInfo            UM3DateTime OPTIONAL
}
```

10.17.1. inServiceDateTime Attribute

해당 애트리뷰트를 갖고 있는 클래스로 모델링한 특정 장치의 서비스 시작 일시를 나타냅니다.

This contains the start date of the specific device modeled by the class with the corresponding attributes.

10.17.2. latestMaintenanceDateTime Attribute

최근 마지막으로 유지보수활동 혹은 점검을 실시한 일시를 나타냅니다.

This contains the date of the last maintenance activity or inspection.

10.17.3. nextMaintenanceDateTime Attribute

다음 유지보수활동에 따른 점검 예정 일시를 나타냅니다.

This contains the schedule for the next maintenance activity.

10.17.4. durablePeriod Attribute

해당 애트리뷰트는 inServiceDateTime 애트리뷰트의 값으로 기록된 일시로부터 사용가능한 기간 혹은 감가상각비가 0 이 되는 기간을 나타냅니다.

This contains the period from the date and time in the inServiceDateTime for the usable period or until the depreciated value becomes 0.

10.17.5. maintenancePeriod Attribute

점검주기를 나타내며 UM3DateTime 타입으로 기록합니다. 해당 애트리뷰트는 초 단위로 저장되며 적절한 단위로 변경 후 사용해야 합니다.

This contains the maintenance schedule, stored in UM3DateTime type. It is stored in the unit of seconds, and it

should be converted to the proper format before use.

10.17.6. numberOfRewired Attribute

inServiceDateTime 애트리뷰트에 기록된 서비스 시작일로부터 지금까지 다시 결선을 한 즉, detach 한 후 재 결선을 한 회수를 나타냅니다.

This indicates the frequency of rewiring after detaching, from the service start date recorded in the inServiceDateTime attribute until now.

10.17.7. detachScheduleInfo Attribute

다음 결선해제 혹은 detach 관련 일정을 나타내는 애트리뷰트 입니다.

This contains the schedule for the next detach.

10.17.8. ACTION

SensorMaintenanceScheduleInfo 클래스에서 새롭게 정의된 액션은 없습니다.

There is no action separately defined in the SensorMaintenanceScheduleInfo class.

10.18. PresentAnalogSensorValue Class

애널로그 방식의 센서를 운용할 때 해당 센서가 측정한 현재의 값을 나타내기 위한 클래스 타입 입니다.

This class is used to represent the current value measured by the corresponding sensor when an analog type sensor is used.

표 22-PresentAnalogSensorValue 클래스의 애트리뷰트 구성 [Attributes of PresentAnalogSensorValue class]

Attribute	Type	M/O
presentValue	UM3Real	M
presentValueTimestamp	UM3DateTime	M
previousValue	UM3Real	M
previousValueTimestamp	UM3DateTime	M
maxPresentValue	UM3Real	M
minPresentValue	UM3Real	M
resolution	UM3Real	M
units	UM3CharacterString	M
updatePeriod	UM3DateTime	M
acceptableMaxPresentValue	UM3Real	M

acceptableMinPresentValue	UM3Real	M

다음은 PresentAnalogSensorValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentAnalogSensorValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PresentAnalogSensorValue UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 값, 현재의 값
            을 측정한 시각, 허용최대값, 허용최소값으로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized by current value, time that current value is taken, upper limit value, and
            lower limit value;
        ATTRIBUTE NAME AS
            presentValue                        presentValue,
            presentValueTimestamp               presentValueTimestamp,
            previousValue                       previousValue,
            previousValueTimestamp              previousValueTimestamp,
            maxPresentValue                     maxPresentValue,
            minPresentValue                     minPresentValue,
            resolution                          resolution,
            units                               units,
            updatePeriod                        updatePeriod,
            acceptableMaxPresentValue           acceptableMaxPresentValue,
            acceptableMinPresentValue           acceptableMinPresentValue;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            애널로그 방식 센서의 측정 값 관련 정보를 저장하고 관리;
            This stores and manages the information related to the measured value of analog type sensor.
;;
```

다음은 PresentAnalogSensorValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentAnalogSensorValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentAnalogSensorValue ::= UM3 CLASS {
        &um3ClassIdentifier                 UM3ClassIdentifier,
```

```
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &presentValue               UM3Real,
        &presentValueTimestamp      UM3DateTime,
        &previousValue              UM3Real,
        &previousValueTimestamp     UM3DateTime,
        &maxPresentValue            UM3Real,
        &minPresentValue            UM3Real,
        &resolution                 UM3Real,
        &units                      UM3CharacterString,
        &updatePeriod               UM3DateTime,
        &acceptableMaxPresentValue  UM3Real,
        &acceptableMinPresentValue  UM3Real
}
```

10.18.1. presentValue Attribute

해당 애트리뷰트의 타입은 UM3Real 타입이며 가장 최근에 센서가 측정한 값을 나타냅니다.

This attribute is UM3Real type, and it contains the value most recently measured by the sensor.

10.18.2. presentValueTimestamp Attribute

상기 presentValue 애트리뷰트의 측정일시를 나타냅니다.

This contains the date and time when the above presentValue was measured.

10.18.3. previousValue Attribute

해당 애트리뷰트의 타입은 UM3Real 타입이며 presentValue 애트리뷰트의 값을 측정하기 전의 presentValue 를 나타냅니다. 에이전트나 혹은 매니저는 presentValue 를 획득하였을 경우 현재 presentValue 를 previousValue 애트리뷰트의 값으로 갱신하고 presentValue 에 최근에 읽어들인 값을 기록합니다.

This attribute is UM3Real type, and it contains the presentValue before measuring the value of the presentValue attribute. When an agent or manager obtains the presentValue, it updates the presentValue with the previousValue and stores the last read value into the presentValue.

10.18.4. previousValueTimestamp Attribute

상기 previousValue 를 읽어들인 일시를 나타내며 presentValueTimestamp 의 값을 previousValueTimestamp

로 복사한 후 presentValueTimestamp 값을 기록합니다.

This contains the date and time when the above previousValue was read. The value of presentValueTimestamp is copied to the previousValueTimestamp, and the value of presentValueTimesamp is stored.

10.18.5. maxPresentValue Attribute

측정을 시작한 이후 혹은 서비스를 시작한 이후 지금까지 측정한 값들 중 최대값을 뜻합니다.

This contains the maximum value among the measurement values from the start of measurement or start of service until now.

10.18.6. minPresentValue Attribute

측정을 시작한 이후 혹은 서비스를 시작한 이후 지금까지 측정한 값들 중 최소값을 뜻합니다.

This contains the minimum value among the measurement values from the start of measurement or start of service until now.

10.18.7. resolution Attribute

분해능을 나타냅니다. 즉, 측정값의 정밀도 혹은 구분가능한 최소단위의 값을 의미합니다.

This contains the resolution, which means the precision of measurement value or value of the minimum unit that can be resolved.

10.18.8. units Attribute

애널로그 값의 단위를 나타냅니다. 타입은 UM3CharacterString 타입입니다.

This contains the unit of analog value. The type is UM3CharacterString.

10.18.9. updatePeriod Attribute

presentValue 를 업데이트하기 위한 주기를 나타냅니다. 타입은 UM3DateTime 이며 그 단위는 초 입니다.

This contains the period to update presentValue. The type is UM3DateTime, and the unit is seconds.

10.18.10. acceptableMaxPresentValue Attribute

presentValue 애트리뷰트의 값이 갖고 있을 수 있는 최대값을 의미합니다. presentValue 애트리뷰트의 값

이 maxPresentValue 애트리뷰트의 값보다 더 클경우에는 센서 혹은 다른 장치의 고장일 확률이 매우 높습니다. 즉, 서비스 관리 분야의 이벤트관리 혹은 인시던트 관리 분야를 위한 애트리뷰트로 볼 수 있습니다.

해당 애트리뷰트는 임계치 (threshold value) 와는 다른 개념임에 주의해야 합니다.

This contains the maximum value that the value of the presentValue attribute is allowed to have. If the value of presentvalue attribute is greater than the value of maxPresentValue attribute, then it is likely that a sensor or other device is malfunctioning. It can be considered as an attribute for event management or incident management in the service management area.

Caution is required because this attribute is a different concept than the threshold value.

10.18.11. acceptableMinPresentValue Attribute

presentValue 애트리뷰트의 값이 갖고 있을 수 있는 최소값을 의미합니다. 해당 값 보다 작은 값을 갖고 있을 경우 센서소자의 고장을 의심할 수 있거나 혹은 측정값의 신뢰도를 의심할 수 있는 상태임을 뜻합니다.

This contains the minimum value that the value of the presentValue attribute is allowed to have. If the value of the presentValue attribute is smaller than this, then sensor failure is likely or the measurement data may not be trusted.

10.18.12. ACTION

PresentAnalogSensorValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentAnalogSensorValue class.

10.19. BinarySensor Class

출력 값을 0 혹은 1 로 모델링 할 수 있는 센싱소자를 나타냅니다. 즉, 전압 혹은 전류를 출력하며 해당 전압 혹은 전류 값이 하위 값 (low value) 과 상위 값 (high value) 으로 구성되는 센서를 나타냅니다. AnalogSensor 와 마찬가지로 BinarySensor 클래스 또한 자체적인 통신 기능을 갖추지 않은 소자 혹은 장치를 나타냅니다. BinarySensor 클래스의 애트리뷰트는 presentValue 애트리뷰트의 클래스 타입만을 제외하고 모든 애트리뷰트가 동일한 형태와 타입들로 정의됩니다.

This class is used for a sensing component that can be modeled as 0 or 1 for its output value. In other words, the sensor that produces the current or voltage output corresponding to the low value or high value can be modeled. As in the AnalogSensor, the BinarySensor class is also used for a device or component that does not have internal communication capability. All attributes of the BinarySensor class except the class type of the presentValue attribute can be defined as the same form and type.

표 23-BinarySenosr 클래스의 애트리뷰트 구성 [Attributes of BinarySensor class]

Attribute	Type	M/O
presentValue	PresentBinarySensorValue	M

다음은 BinarySensor 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The BinarySensor class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
BinarySensor UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base, SensorBase;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 presentValue 애트리뷰
            트의 값으로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized as the value of the presentValue attribute;
        ATTRIBUTE NAME AS
            presentValue                        presentValue;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            특정 소자로 데이터를 측정하고 이를 결선을 통해 상위 게이트웨이가 수집, 혹은 결선을 통
            해 상위 게이트웨이가 소자를 제어, 출력값은 low 혹은 high 로 구성;
            Data is measured with a specific device and collected by an upper level gateway through a
            cable, or device is controlled by upper level gateway through a cable, and output value has two
            states of low or high.
;;
```

다음은 BinarySensor 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The BinarySensor class can be defined as follows by using ASN.1 regular expressions.

```
BinarySensor ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
```

```
        &um3AttributeList                UM3ObjectList,
        &manufacturerInfo                CompanyInfo OPTIONAL,
        &vendorInfo                      CompanyInfo OPTIONAL,
        &maintenancePersonel             PersonInfo OPTIONAL,
        &operationalCondition            DeviceOperationalCondition OPTIONAL,
        &hasBattery                      UM3Boolean,
        &batteryInfo                     InstalledBattery OPTIONAL,
        &hasBackupBattery                UM3Boolean,
        &backupBatteryInfo               InstalledBattery OPTIONAL,
        &presentBatteryValueInfo         PresentBatteryValue,
        &presentBackupBatteryValueInfo   PresentBatteryValue OPTIONAL,
        &installedEnvironment            InstalledEnvironment OPTIONAL,
        &powerConsumption                UM3Real,
        &levelInAConfigurationTree       UM3UnsignedInteger16,
        &upperGatewayAddress             UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo             DeviceMaintenanceScheduleInfo,
        &presentSensorStatus             DeviceOperationStatus,
        &serialNumber                    UM3CharacterString,
        &presentValue                    PresentBinarySensorValue
}
```

BinarySensor 클래스도 AnalogSensor 와 마찬가지로 UM3Base 클래스와 SensorBase 클래스로부터 그 속성을 상속받는 것으로 정의되어 있습니다.

As in the AnalogSensor class, the BinarySensor class inherits the attributes from the SensorBase class and UM3Base class.

10.19.1. presentValue Attribute

센서의 현재 값을 갖고 있는 애트리뷰트 입니다. presentValue 애트리뷰트를 제외한 나머지 애트리뷰트들은 서비스 관리 영역의 애트리뷰트들로 볼 수 있으며 AnalogSensor 와 그 구성이 동일합니다.

This attribute is used to indicate the current value of the sensor. All other attributes except the presentValue attribute can be considered as the ones for the service management area, and the configuration is the same as AnalogSensor.

10.19.2. ACTION

BinarySensor 클래스에서 별도로 다시 정의된 액션은 없습니다.

There is no action separately defined in the BinarySensor class.

10.20. PresentBinarySensorValue Class

디지틀 방식의 센서를 운용할 때 해당 센서가 측정한 현재의 값을 나타내기 위한 클래스 타입 입니다.

This class is used to store the current value measured by the corresponding digital type sensor.

표 24-PresentBinarySensorValue 클래스의 애트리뷰트 구성 [Attributes of PresentBinarySensorValue class]

Attribute	Type	M/O
presentValue	UM3Boolean	M
presentValueTimestamp	UM3DateTime	M
previousValue	UM3Boolean	M
previousValueTimestamp	UM3DateTime	M
updatePeriod	UM3DateTime	M
stateChangedCount	UM3UnsignedInteger16	M
stateCountStartedDateTime	UM3DateTime	M

다음은 PresentBinarySensorValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentBinarySensorValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PresentBinarySensorValue UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 값, 현재의 값
        을 측정한 시각, 현재값의 변화 횟수 등으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by current value, time when the current value is taken, and frequency of
        changes in the current value;
    ATTRIBUTE NAME AS
        presentValue                    presentValue,
        presentValueTimestamp           presentValueTimestamp,
        previousValue                   previousValue,
        previousValueTimestamp          previousValueTimestamp,
        updatePeriod                    updatePeriod,
        stateChangedCount               stateChangedCount,
        stateCountStartedDateTime       stateCountStartedDateTime;
    ACTION DEFINED AS
        None;
    BEHAVIOR
        바이너리 방식 센서의 측정 값 관련 정보를 저장하고 관리;
```

This stores and manages the information related to the measurement value of the binary type sensor.
;;

다음은 PresentBinarySensorValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentBinarySensorValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentBinarySensorValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3Boolean,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3Boolean,
        &previousValueTimestamp         UM3DateTime,
        &updatePeriod                   UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}
```

10.20.1. presentValue Attribute

해당 애트리뷰트의 타입은 UM3Boolean 타입이며 가장 최근에 센서가 측정한 값을 나타냅니다.

This attribute is UM3Boolean type, and it contains the most recently value measured by the sensor.

10.20.2. presentValueTimestamp Attribute

상기 presentValue 애트리뷰트의 측정일시를 나타냅니다.

This contains the date and time when the value of the above presentValue attribute is measured.

10.20.3. previousValue Attribute

해당 애트리뷰트의 타입은 UM3Boolean 타입이며 presentValue 애트리뷰트의 값을 측정하기 전의 presentValue 를 나타냅니다. 에이전트나 혹은 매니저는 presentValue 를 획득하였을 경우 현재 presentValue 를 previousValue 애트리뷰트의 값으로 갱신하고 presentValue 에 최근에 읽어들인 값을 기록합니다.

This attribute is UM3Boolean type, and it contains the presentValue before measuring the value of the presentValue attribute. When an agent or manager obtains the presentValue, it updates the presentValue with previousValue and stores the last read value into presentValue.

10.20.4. previousValueTimestamp Attribute

상기 previousValue 를 읽어들인 일시를 나타내며 presentValueTimestamp 의 값을 previousValueTimestamp 로 복사한 후 presentValueTimestamp 값을 기록합니다.

This contains the date and time when the above previousValue was read. The value of presentValueTimestamp is copied to the previousValueTimestamp, and the value of presentValueTimesamp is stored.

10.20.5. updatePeriod Attribute

presentValue 애트리뷰트의 값을 갱신하기 위한 측정 주기를 나타내며 그 단위는 초 입니다.

It contains the period of measurement to update the value of presentValue attribute, and the unit is one second.

10.20.6. stateChangedCount Attribute

측정을 시작한 이후 지금까지 바이너리 값이 변화한 횟수를 나타냅니다. 측정을 시작한 시점은 stateCountStartedDateTime 애트리뷰트에 기록됩니다.

This contains the frequency of changes in binary value from the start of measurement until now. The time when the measurement was started is stored in the stateCountStartedDateTime attribute.

10.20.7. stateCountStartedDateTime Attribute

상기 stateChangedCount 애트리뷰트의 값 즉, presentValue 애트리뷰트의 값이 변화한 횟수를 세기 시작한 시점을 나타냅니다. 거의 대부분 해당 센서의 상태를 리셋한 시점과 동일한 시점을 가리키게 됩니다.

This contains the time when the measurement of value of stateChangedCount attribute, frequency of changes in value of the presentValue attributes, is started. In most cases, it is the same as the type when the corresponding sensor is reset.

10.20.8. ACTION

PresentBinarySensorValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentBinarySensorValue class.

10.21. AccumulatorSensor Class

AccumulatorSensor 클래스는 펄스 혹은 특정 이벤트가 발생하는 횟수를 누적하여 그 값을 출력하는 센서입니다. 주로 적산계량을 위해 사용되며 대부분 센서소자로부터의 출력이 아닌 센서소자와 연결된 인터페이스 회로 등에서 출력되는 값을 의미합니다.

The AccumulatorSensor class is for the sensor that accumulates the frequency of pulse or specific event and produces the corresponding value. It is usually used for accumulated metering, which means that the output is from the interface circuit connected to the sensor device instead of taking the output from a sensor component.

표 25-AccumulatorSensor 클래스의 애트리뷰트 구성 [Attributes of AccumulatorSensor class]

Attribute	Type	M/O
presentValue	PresentAccumulatorSensorValue	M

다음은 AccumulatorSensor 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

AccumulatorSensor class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
AccumulatorSensor UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base, SensorBase;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 presentValue 애트리뷰
        트의 값으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized as the value of the presentValue attribute;
    ATTRIBUTE NAME AS
        presentValue                        presentValue;

    ACTION DEFINED AS
        None;
    BEHAVIOR
        특정 소자로 펄스데이터를 측정하여 발생 횟수를 누적저장하고 이를 결선을 통해 상위 게이
        트웨이가 수집, 혹은 결선을 통해 상위 게이트웨이가 소자를 제어;
        Pulse data is measured with a specific device, frequency is accumulated and stored, and data is
        collected by and upper level gateway through a cable, or a device is controlled by an upper level
        gateway through a cable.
;;
```

다음은 AccumulatorSensor 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The AccumulatorSensor class can be defined as follows by using ASN.1 regular expressions.

```
AccumulatorSensor ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentSensorStatus            DeviceOperationStatus,
        &serialNumber                   UM3CharacterString,
        &presentValue                   PresentAccumulatorSensorValue
}
```

AccumulatorSensor 클래스는 SensorBase 클래스와 UM3Base 클래스로부터 그 속성을 상속받습니다. 앞서 정의한 AnalogSensor 와 마찬가지로 상기 ASN.1 정규표현식으로 정의된 각 애트리뷰트의 상세정의는 SensorBase 클래스와 UM3Base 클래스를 참조해야 합니다.

The AccumulatorSensor class inherits attributes from the SensorBase class and UM3Base class. Like the AnalogSensor that was defined earlier, the SensorBase class and UM3Base class should be referred for the detailed definition of attributes defined with the above ASN.1 regular expression.

10.21.1. presentValue Attribute

센서의 현재 값을 갖고 있는 애트리뷰트이며 그 타입은 PresentAccumulatorSensorValue 클래스 타입입니다.

This attribute is used to indicate the current value of the sensor, and the type is PresentAccumulatorSensorValue class type.

10.21.2. ACTION

AccumulatorSensor 클래스에서 별도로 다시 정의된 액션은 없습니다.

There is no action separately defined in the AccumulatorSensor class.

10.22. PresentAccumulatorSensorValue Class

펄스 형태의 발생횟수를 지속적으로 축적하여 그 출력값으로 출력하는 방식의 센서를 운용할 때 해당 센서가 측정한 현재의 값을 나타내기 위한 클래스 타입 입니다. 실제로는 센서소자가 이와 같은 축적 값을 출력하는 것이 아니라 센서와 연결된 컨트롤러 혹은 전기적 회로가 축적값을 저장하는 등의 연산과정을 수행하게 됩니다.

This class type is used to store the current value measured by the sensor when the sensor continuously accumulates the frequency of pulse type data and produces the accumulated value as output. Sensing component does not produce the accumulated output. Rather, the controller or electronic circuit connected to the sensor performs the computation to accumulate the values.

표 26-PresentAccumulatorSensorValue 클래스의 애트리뷰트 구성
[Attributes of PresentAccumulatorSensorValue class]

Attribute	Type	M/O
presentValue	UM3Real	M
presentValueTimestamp	UM3DateTime	M
previousValue	UM3Real	M
previousValueTimestamp	UM3DateTime	M
minPresentValue	UM3Real	M
minPresentValueTimestamp	UM3DateTime	M
units	UM3CharacterString	M
updatePeriod	UM3DateTime	M
acceptableMaxPresentValue	UM3Real	M

다음은 PresentAccumulatorSensorValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentAccumulatorSensorValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as

follows.

PresentAccumulatorSensorValue UM3 OBJECT CLASS
DERIVED FROM
UM3Base;
CHARACTERIZED BY
um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 값, 현재의 값을 측정한 시각 등으로 특징지워짐;
um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and they are characterized by current value and time that current value is taken;
ATTRIBUTE NAME AS

presentValue	presentValue,
presentValueTimestamp	presentValueTimestamp,
previousValue	previousValue,
previousValueTimestamp	previousValueTimestamp,
minPresentValue	minPresentValue,
minPresentValueTimestamp	minPresentValueTimestamp,
units	units,
updatePeriod	updatePeriod,
acceptableMaxPresentValue	acceptableMaxPresentValue;

ACTION DEFINED AS
None;
BEHAVIOR
적산 방식 센서의 측정 값 관련 정보를 저장하고 관리;
This stores and manages the information related to the values measured by the accumulator type sensor.
;;

다음은 PresentAccumulatorSensorValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentAccumulatorSensorValue class can be defined as follows by using ASN.1 regular expressions.

PresentAccumulatorSensorValue ::= UM3 CLASS {

&um3ClassIdentifier	**UM3ClassIdentifier,**
&um3ObjectName	**UM3ObjectName,**
&um3ObjectNameAlias	**UM3CharacterString OPTIONAL,**
&um3ObjectDescription	**UM3CharacterString OPTIONAL,**
&um3AttributeList	**UM3ObjectList,**
&presentValue	**UM3Real,**
&presentValueTimestamp	**UM3DateTime,**
&previousValue	**UM3Real,**
&previousValueTimestamp	**UM3DateTime,**
&minPresentValue	**UM3Real,**

```
        &minPresentValueTimestamp       UM3DateTime,
        &units                          UM3CharacterString,
        &updatePeriod                   UM3DateTime,
        &acceptableMaxPresentValue      UM3Real
}
```

10.22.1. presentValue Attribute

해당 애트리뷰트의 타입은 UM3Real 타입이며 가장 최근에 센서가 측정한 값을 나타냅니다.

This attribute is UM3Real type, and it contains the value most recently measured by the sensor.

10.22.2. presentValueTimestamp Attribute

상기 presentValue 애트리뷰트의 측정일시를 나타냅니다.

This contains the date and time when the above presentValue was measured.

10.22.3. previousValue Attribute

해당 애트리뷰트의 타입은 UM3Real 타입이며 presentValue 애트리뷰트의 값을 측정하기 전의 presentValue 를 나타냅니다. 에이전트나 혹은 매니저는 presentValue 를 획득하였을 경우 현재 presentValue 를 previousValue 애트리뷰트의 값으로 갱신하고 presentValue 에 최근에 읽어들인 값을 기록합니다.

This attribute is UM3Real type, and it contains the presentValue before measuring the value of the presentValue attribute. When an agent or manager obtains presentValue, it updates the presentValue with previousValue and stores the last read value into presentValue.

10.22.4. previousValueTimestamp Attribute

상기 previousValue 를 읽어들인 일시를 나타내며 presentValueTimestamp 의 값을 previousValueTimestamp 로 복사한 후 presentValueTimestamp 값을 기록합니다.

This contains the date and time when the above previousValue was read. The value of presentValueTimestamp is copied to the previousValueTimestamp, and the value of presentValueTimesamp is stored.

10.22.5. minPresentValue Attribute

측정을 시작한 이후 혹은 서비스를 시작한 이후 지금까지 측정한 값들 중 최소값을 뜻합니다. 이는

측정을 시작한 의미있는 시점에서 센서의 누적값이 얼마인가를 나타내는 용도로 사용되기도 합니다.

This contains the minimum value among the measurement values from the start of measurement or start of service until now. It is also used for the purpose of showing the accumulated value at a meaningful point when the measurement was started.

10.22.6. minPresentValueTimestamp Attribute

상기 minPresentValue 애트리뷰트가 측정된 일시를 나타냅니다.

This contains date and time when the above minPresentValue attribute value is measured.

10.22.7. units Attribute

축적 값의 단위를 나타냅니다. 타입은 UM3CharacterString 타입입니다.

This contains the unit of accumulated value. The type is UM3CharacterString.

10.22.8. updatePeriod Attribute

presentValue 를 업데이트하기 위한 주기를 나타냅니다. 타입은 UM3DateTime 이며 그 단위는 초 입니다.

This contains the period to update presentValue. The type is UM3DateTime and the unit is seconds.

10.22.9. acceptableMaxPresentValue Attribute

presentValue 애트리뷰트의 값이 갖고 있을 수 있는 최대값을 의미합니다. 적산 방식의 센서장치의 경우 해당 애트리뷰트의 의미는 측정할 수 있는 최대치를 의미하며 다시 리셋되어 0 으로부터 혹은 그 이하의 값으로부터 누적연산을 다시 시작해야 한다는 의미이기도 합니다.

해당 애트리뷰트는 임계치 (threshold value) 와는 다른 개념임에 주의해야 합니다.

This contains the maximum value that the value of the presentValue attribute is allowed to have. This means the maximum value that can be measured in case of an accumulation type sensor device, and the value should be reset to 0 or the accumulation should start from the lower value.

Caution is required because this attribute is a different concept than the threshold value.

10.22.10. ACTION

PresentAccumulatorSensorValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentAccumulatorSensorValue class.

10.23. MultiStateSensor Class

MultiStateSensor 클래스는 그 출력값을 2 개 혹은 그 이상의 단계로 나누어 출력하는 물리적 자원을 모델링한 결과입니다. 각 단계별 명칭은 자유롭게 정의할 수 있으며 그 개수는 가급적 10 개에서 20 개 사이의 단계로만 정의하도록 권고합니다.

The MultiStateSensor class is for modeling physical resources that produce output in two or more states. The name for each state can be freely defined, and it is recommended to limit the number of states between 10 and 20.

한 가지 유의할 점은 MultiStateSensor 의 단계는 순차적으로 증가 혹은 감소하는 값이 아니라는 점입니다. 이는 0 으로부터 시작하여 차례로 1, 2 등으로 그 값을 증가시키는 단계별 증가 혹은 단계별 -1, -2 등의 감소 등과는 그 형식이 다른 형태임에 유의해야 합니다.

One thing to be noted here is that the state of MultiStateSensor does not increase or decrease sequentially. It is different from the sequential state, which increases the value to 1, 2, ... from 0, or decreasing the value by -1, -2, …

The attributes of the MultiStateSensor class are as follows.

표 27-MultiStateSenosr 클래스의 애트리뷰트 구성 [Attributes of MultiStateSensor class]

Attribute	Type	M/O
presentValue	PresentMultiStateSensorValue	M

다음은 MultiStateSensor 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The MultiStateSensor class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
MultiStateSensor UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base, SensorBase;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 presentValue 애트리뷰
        트의 값 즉, 단계별 상태 값으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized as the value of the presentValue attribute i.e. value of each state;
    ATTRIBUTE NAME AS
        presentValue                    presentValue;
    ACTION DEFINED AS
```

None;
BEHAVIOR
특정 소자로 데이터를 측정하고 이를 결선을 통해 상위 게이트웨이가 수집, 혹은 결선을 통해 상위 게이트웨이가 소자를 제어, 출력값은 정의된 각 단계별 값으로 구성;
Data is measured with a specific device and collected by an upper level gateway through a cable, or a device is controlled by upper level gateway through a cable, and the output value has predefined states.
;;

다음은 MultiStateSensor 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The MultiStateSensor class can be defined as follows by using ASN.1 regular expressions.

```
MultiStateSensor ::= UM3 CLASS {
        &um3ClassIdentifier                UM3ClassIdentifier,
        &um3ObjectName                     UM3ObjectName,
        &um3ObjectNameAlias                UM3CharacterString OPTIONAL,
        &um3ObjectDescription              UM3CharacterString OPTIONAL,
        &um3AttributeList                  UM3ObjectList,
        &manufacturerInfo                  CompanyInfo OPTIONAL,
        &vendorInfo                        CompanyInfo OPTIONAL,
        &maintenancePersonel               PersonInfo OPTIONAL,
        &operationalCondition              DeviceOperationalCondition OPTIONAL,
        &hasBattery                        UM3Boolean,
        &batteryInfo                       InstalledBattery OPTIONAL,
        &hasBackupBattery                  UM3Boolean,
        &backupBatteryInfo                 InstalledBattery OPTIONAL,
        &presentBatteryValueInfo           PresentBatteryValue,
        &presentBackupBatteryValueInfo     PresentBatteryValue OPTIONAL,
        &installedEnvironment              InstalledEnvironment OPTIONAL,
        &powerConsumption                  UM3Real,
        &levelInAConfigurationTree         UM3UnsignedInteger16,
        &upperGatewayAddress               UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo               DeviceMaintenanceScheduleInfo,
        &presentSensorStatus               DeviceOperationStatus,
        &serialNumber                      UM3CharacterString,
        &presentValue                      PresentMultiStateSensorValue
}
```

MultiStateSensor 클래스도 AnalogSensor, BinarySensor 와 마찬가지로 UM3Base 클래스와 SensorBase 클래스로부터 그 속성을 상속받는 것으로 정의되어 있습니다.

As in the AnalogSensor class, the MultiStateSensor class inherits the attributes from the SensorBase class and UM3Base class.

10.23.1. presentValue Attribute

센서의 현재 값을 갖고 있는 애트리뷰트 입니다. presentValue 애트리뷰트를 제외한 나머지 애트리뷰트들은 서비스 관리 영역의 애트리뷰트들로 볼 수 있으며 AnalogSensor 와 그 구성이 동일합니다.

This attribute is used to indicate the current value of the sensor. All other attributes except the presentValue attribute can be considered as the ones for the service management area, and the configuration is the same as AnalogSensor.

10.23.2. ACTION

MultiStateSensor 클래스에서 별도로 다시 정의된 액션은 없습니다.

There is no action separately defined in the MultiStateSensor class.

10.24. PresentMultiStateSensorValue Class

PresentMultiStateSensorValue 클래스는 다단계의 출력값을 갖는 MultiStateSensor 의 현재 상태값을 저장 관리하기 위한 클래스 타입입니다. 현재의 상태 값을 저장하기 위해서는 센서가 갖고 있는 상태 값의 구분단계에 관한정보, 현재의 값, 이전의 상태의 값 등에 관한정보를 함께 갖고 있어야 합니다.

다음 표 28-PresentMultiStateSensorValue 클래스의 애트리뷰트 구성[는 PresentMultiStateSensorValue 클래스의 애트리뷰트 구성을 나타내고 있습니다.

This class is used to store the current states of the MultiStateSensor that produces multiple states output. To store the current state, the information about the classification of stages for each state, current value, and previous state of the sensor is also required.

Table 28 below shows the attributes of the PresentMultiStateSensorValue class.

표 28-PresentMultiStateSensorValue 클래스의 애트리뷰트 구성
[Attributes of PresentMultiStateSensorValue class]

Attribute	Type	M/O
presentValue	UM3UnsignedInteger16	M
presentValueTimestamp	UM3DateTime	M
previousValue	UM3UnsignedInteger16	M
previousValueTimestamp	UM3DateTime	M
numberOfStates	UM3UnsignedInteger16	M
stateNames	UM3 SEQUENCE OF UM3CharacterString	M
updatePeriod	UM3DateTime	M
stateChangedCount	UM3UnsignedInteger16	M
stateCountStartedDateTime	UM3DateTime	M

다음은 PresentMultiStateSensorValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentMultiStateSensorValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PresentMultiStateSensorValue UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 값, 상태의 이
            름, 상태의 개수 등으로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized by current value, name of state, and number of states;
        ATTRIBUTE NAME AS
            presentValue                        presentValue,
            presentValueTimestamp               presentValueTimestamp,
            previousValue                       previousValue,
            previousValueTimestamp              previousValueTimestamp,
            numberOfStates                      numberOfStates,
            stateNames                          stateNames,
            updatePeriod                        updatePeriod,
            stateChangedCount                   stateChangedCount,
            stateCountStartedDateTime           stateCountStartedDateTime;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            단계별 상태 값을 갖는 센서의 측정 값 관련 정보를 저장하고 관리;
            This stores and manages the information related to the measurement value of sensor that has
            state values by stage;
;;
```

다음은 PresentMultiStateSensorValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentMultiStateSensorValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentMultiStateSensorValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3UnsignedInteger16,
        &presentValueTimestamp          UM3DateTime,
```

```
        &previousValue                  UM3UnsignedInteger16,
        &previousValueTimestamp         UM3DateTime,
        &numberOfStates                 UM3UnsignedInteger16,
        &stateNames                     UM3 SEQUENCE OF UM3CharacterString,
        &updatePeriod                   UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}
```

10.24.1. presentValue Attribute

해당 애트리뷰트의 타입은 UM3UnsignedInteger16 타입이며 가장 최근에 센서가 측정한 값을 나타냅니다. presentValue 애트리뷰트의 값은 numberOfStates 애트리뷰트의 값보다 클 수 없으며 numberOfStates 애트리뷰트의 값이 *n* 이라면 그 값의 범위는 *0* 에서 *(n − 1)* 로 정의됩니다.

This attribute is UM3UnsignedInteger16 type, and it contains the value most recently measured by the sensor. The value of the presentValue attribute cannot be greater than the value of the numberOfState attribute. When the value of the numberOfState attribute is *n*, the range of the value can be defined as *0 ~ (n − 1)*.

10.24.2. presentValueTimestamp Attribute

상기 presentValue 애트리뷰트의 측정일시를 나타냅니다.

This contains the date and time when the above presentValue was measured.

10.24.3. previousValue Attribute

presentValue 애트리뷰트의 값을 측정하기 전의 presentValue 를 나타냅니다. 에이전트나 혹은 매니저는 presentValue 를 획득하였을 경우 현재 presentValue 를 previousValue 애트리뷰트의 값으로 갱신하고 presentValue 에 최근에 읽어들인 값을 기록합니다.

This contains the presentValue before measuring the value of the presentValue attribute. When an agent or manager obtains presentValue, it updates presentValue with previousValue and stores the last read presentValue.

10.24.4. previousValueTimestamp Attribute

상기 previousValue 를 읽어들인 일시를 나타내며 presentValueTimestamp 의 값을 previousValueTimestamp 로 복사한 후 presentValueTimestamp 값을 기록합니다.

This contains the date and time when the above previousValue was read. The value of presentValueTimestamp is

copied to the previousValueTimestamp, and the value of presentValueTimesamp is stored.

10.24.5. numberOfStates Attribute

MultiStateSensor 의 단계별 출력 값이 몇 단계로 나누어지는가를 나타내는 애트리뷰트입니다.

This attribute indicates the number of states of the MultiStateSensor output.

10.24.6. stateNames Attribute

각 단계별 명칭을 나타냅니다. 해당 애트리뷰트의 타입은 UM3 SEQUENCE OF UM3CharacterString 타입입니다. 첫 번째 엘리먼트의 인덱스는 0 부터 시작합니다. 해당 애트리뷰트은 현재의 상태값을 표현하기 위한 보조적인 수단으로 활용됩니다.

This contains the name of each state. The type of this attribute is UM3 SEQUENCE OF UM3CharacterString. The index of the first element is 0. This attribute is used as an auxiliary means of representing the current state.

10.24.7. updatePeriod Attribute

presentValue 를 업데이트하기 위한 주기를 나타냅니다. 타입은 UM3DateTime 이며 그 단위는 초 입니다.

This contains the period to update presentValue. The type is UM3DateTime and the unit is seconds.

10.24.8. stateChangedCount Attribute

측정시작 이후 혹은 서비스 시작 이후 상태가 천이된 회수를 나타냅니다.

This contains the total number of changes of states since the start of measurement or start of service.

10.24.9. stateCountStartedDateTime Attribute

상기 상태 변화를 기록하기 시작한 시점을 나타냅니다.

This contains the time when the above state change monitoring was started.

10.24.10. ACTION

PresentMultiStateSensorValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentMultiStateSensorValue class.

10.25. ControlBase Class

ControlBase 클래스는 AnalogControl, BinaryControl, MultistateControl 등의 클래스를 위한 최상위 베이스 클래스입니다. ControlBase 는 기본적으로 SensorBase 클래스와 그 애트리뷰트의 구성이 거의 유사합니다. 단, 데이터를 출력하는 물리적 자원을 SensorBase 클래스로 모델링한 반면, 데이터를 입력받고 해당 데이터이 값에 따라 특정 장치를 제어하거나 특성을 변화시키는 경우 이를 ControlBase 로부터 속성을 상속받는 클래스들로 모델링하게 됩니다.

The ControlBase class is the highest level base class for classes like AnalogControl, BinaryControl, and MultiStateControl. The configuration of attributes for ControlBase is very similar to that of SensorBase class. While SensorBase is modeling the physical resources that produce data, classes inherited from the ControlBase class are used to receive data and control and change the characteristics of specific devices based on the values of the data.

표 29-ControlBase 클래스의 애트리뷰트 구성 [Attributes of ControlBase]

Attribute	Type	M/O
manufacturerInfo	CompanyInfo	O
vendorInfo	CompanyInfo	O
maintenancePersonel	PersonInfo	O
operationalCondition	DeviceOperationalCondition	O
hasBattery	UM3Boolean	M
batteryInfo	InstalledBattery	O
hasBackupBattery	UM3Boolean	M
backupBatteryInfo	InstalledBattery	O
presentBatteryValueInfo	PresentBatteryValue	M
presentBackupBatteryValueInfo	PresentBatteryValue	O
installedEnvironment	InstalledEnvironment	O
powerConsumption	UM3Real	M
levelInAConfigurationTree	UM3UnsignedInteger16	M
upperGatewayAddress	UM3TcpIpAddress	O
operationalTimeInfo	DeviceMaintenanceScheduleInfo	M
presentSensorStatus	DeviceOperationStatus	M
serialNumber	UM3CharacterString	M

다음은 ControlBase 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The ControlBase class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
ControlBase UM3 OBJECT CLASS
        DERIVED FROM
```

UM3Base;
CHARACTERIZED BY
um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 AnalogControl, BinaryControl 등을 위한 베이스 클래스로 특징지워짐;
um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and they are characterized as base classes for AnalogControl and BinaryControl;
ATTRIBUTE NAME AS

manufacturerInfo	manufacturerInfo,
vendorInfo	vendorInfo,
maintenancePersonel	maintenancePersonel,
operationalCondition	operationalCondition,
hasBattery	hasBattery,
batteryInfo	batteryInfo,
hasBackupBattery	hasBackupBattery,
backupBatteryInfo	backupBatteryInfo,
presentBatteryValueInfo	presentBatteryValueInfo,
presentBackupBatteryValueInfo	presentBackupBatteryValueInfo,
installedEnvironment	installedEnvironment,
powerConsumption	powerConsumption,
levelInAConfigurationTree	levelInAConfigurationTree,
upperGatewayAddress	upperGatewayAddress,
operationalTimeInfo	operationalTimeInfo,
presentSensorStatus	presentSensorStatus,
serialNumber	serialNumber;

ACTION DEFINED AS
None;
BEHAVIOR
특정 제어장치의 구동을 위해 입력 데이터를 받아들이며, 상위 게이트웨이가 해당 제어장치를 관리;
This receives input date to drive a specific control device, and upper level gateway manages the corresponding control device.
;;

다음은 ControlBase 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The ControlBase class can be defined as follows by using ASN.1 regular expressions.

```
ControlBase ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
```

```
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentControlStatus           DeviceOperationStatus,
        &serialNumber                   UM3CharacterString
}
```

10.25.1. manufacturerInfo Attribute

제어장치 제조 회사의 정보를 갖고 있습니다.

This contains the information about the manufacturer of the control device.

10.25.2. vendorInfo Attribute

제어장치 공급 회사의 정보를 갖고 있습니다.

This contains the information about the vendor of the control device.

10.25.3. maintenancePersonel Attribute

제어 장치에 대한 유지보수 혹은 점검 담당자 정보를 갖고 있습니다.

This contains the information about the manager or personnel responsible for maintenance of the control device.

10.25.4. operationalCondition Attribute

제어장치의 운용환경에 대한 정보를 갖고 있습니다.

This contains the information about the operation environment of the control device.

10.25.5. hasBattery Attribute

제어장치를 위한 별도의 배터리 장치를 갖고 있는지를 나타냅니다. 그 값이 TRUE 일 경우 배터리가

장착되어 있음을 뜻하며, FALSE 일 경우 결선되어 있는 게이트웨이에서 전력을 공급함을 뜻 합니다.

This indicates whether there is a separate battery device for the control device. If this value is TRUE, then it means that a battery is installed. If it is FALSE, then it means that the power is supplied by the gateway device connected through a cable.

10.25.6. batteryInfo Attribute

배터리가 장착되어 있을 경우 해당 배터리의 정보를 기록합니다.

This attribute is for storing the information related to a battery when it is installed.

10.25.7. hasBackupBattery Attribute

백업용 보조 배터리가 장착되어 있는지의 여부를 나타냅니다.

This attribute indicates whether there is a backup battery in addition to the main battery.

10.25.8. backupBatteryInfo Attribute

백업용 보조 배터리의 사양 등 여러가지 정보를 기록합니다.

This attribute is for storing the information related to the aux battery for backup.

10.25.9. presentBatteryValueInfo Attribute

현재 배터리의 잔량 등 상태정보 값을 갖게 됩니다.

This contains the status information about the remaining capacity of the battery.

10.25.10. presentBackupBatteryValueInfo Attribute

백업용 보조 배터리의 현재 상태값 들을 갖게 됩니다.

This contains the status information about the remaining capacity of the aux battery for backup.

10.25.11. installedEnvironment Attribute

제어장치가 설치된 환경정보를 나타냅니다. 예를 들어 제어장치의 사이즈 등은 installedEnvironment.size 애트리뷰트의 값을 이용하여 UM3CharacterString 타입으로 정의할 수 있습니다.

This contains the information about the environment where the control device is installed. For example, the size of the control device can be defined in UM3CharacterString type by using the value of the installedEnvironment.size

attribute.

10.25.12. powerConsumption Attribute

제어장치의 전력소모와 관련된 정보를 기록합니다.

This contains the information related to the power consumption of the sensor.

10.25.13. levelInAConfigurationTree Attribute

configuration tree 에서의 레벨을 나타냅니다. 앞서 정의한 바와 같이 configuration tree 는 information tree 혹은 containment tree 와 동일하거나 유사한 개념으로 사용됩니다.

This indicates the level in the configuration tree. As defined earlier, the configuration is used with a similar or the same concept as the information tree of containment tree.

10.25.14. upperGatewayAddress Attribute

비록 제어장치가 결선으로 연결된 상위 레벨의 게이트웨이와 TCP/IP 기반의 통신을 하지 않는다 하더라도 AnalogControl 등과 같은 하위 오브젝트는 상위 게이트웨이의 TCP/IP 통신을 위한 주소를 갖고 있습니다.

해당 애트리뷰트는 선택요소 (optional) 로 정의되어 있습니다.

Even when the control device does not communicate with the upper level gateway based on TCP/IP through a cable, lower level objects like AnalogControl object has the address of the upper level gateway for TCP/IP communication.

The corresponding attribute is defined as an optional field.

10.25.15. operationalTimeInfo Attribute

SensorMaintenanceScheduleInfo 클래스 타입의 애트리뷰트이며 점검주기, 예정 점검일시 등을 나타냅니다.

This is an attribute of the SensorMaintenanceScheduleInfo class type, and it contains the inspection period and scheduled inspection date.

10.25.16. presentControlStatus Attribute

제어장치의 현재 상태를 나타냅니다. outOfService 의 경우 컨트롤러가 더 이상 제어 서비스를 제공하지 않는다는 뜻 입니다. 즉, 더 이상 장치를 제어할 수 없다는 뜻 입니다. inService 는 현재 정상적인

형태로 제어장치가 동작 중이며 제어 서비스를 제공하고 있다는 뜻 입니다. inMaintenance 는 현재 유지보수를 위한 점검 중이며 다시 서비스가 재개될 예정임을 뜻 합니다. inFault 는 현재 고장상태이며 다시 정상화 할 계획임을 의미합니다. 결선으로 연결된 컨트롤의 경우 communicationFailure 와 detached 는 동일한 의미로 사용될 수 있습니다. unknownCause 는 원인을 알 수 없는 비정상 상태임을 뜻 하며 해당 비정상 상태를 정상화 시킬 예정임을 그 뜻에 내포하고 있습니다.

This contains the current status of the control device. When this value is outOfService, it means that no more control service is provided. When it is inService, it means that the control device currently operates normally and provides the control service. inMaintenance means that the control device is currently under maintenance and service will resume later. inFault means that the sensor is out of order, and it will be corrected again. In case of a control device connected by cable, communicationFailure and detached can be used for the same meaning. unknownCause means that the control device is not working correctly due to some unknown reason, and it implies that the failure will be fixed.

10.25.17. serialNumber Attribute

시리얼 번호와 관련된 정보를 갖고 있는 애트리뷰트 입니다.

This attribute contains the information related to the serial number.

10.25.18. ACTION

ControlBase 클래스에서 별도로 다시 정의된 액션은 없습니다.

There is no action separately defined in the ControlBase class.

10.26. AnalogControl Class

애널로그 방식의 입력 신호를 받아들여 제어장치를 제어하는 경우 이를 AnalogControl 장치로 모델링 합니다. 즉, 일정한 전류 혹은 전압을 인가하는 경우 저항값의 변화를 통해 전압 혹은 전류를 변화시키고 이를 제어장치의 입력으로 사용하는 경우입니다.

AnalogControl 클래스는 다음과 같은 애트리뷰트로 구성됩니다.

When an analog input signal is received and used to control the control device, it is modeled as AnalogControl device. When constant current or voltage is applied, the voltage or current is changed by varying the resistance value, and it is used for the input for the control device.

The AnalogControl class has the following attributes.

표 30-AnalogControl 클래스의 애트리뷰트 구성 [Attributes of AnalogControl class]

Attribute	Type	M/O
presentValue	PresentAnalogControlValue	M

다음은 AnalogControl 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

AnalogControl class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
AnalogControl UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base, ControlBase;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 presentValue 애트리뷰
        트의 값으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized as the value of the presentValue attribute;
    ATTRIBUTE NAME AS
        presentValue                        presentValue;

    ACTION DEFINED AS
        None;
    BEHAVIOR
        상위 게이트웨이가 결선을 통해 제어장치를 제어 혹은 네트워크를 통해 제어장치를 제어;
        The control device is controlled through the cable or network by an upper level gateway.
;;
```

다음은 AnalogControl 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The AnalogControl class can be defined as follows by using ASN.1 regular expressions.

```
AnalogControl ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
```

```
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentControlStatus           DeviceOperationStatus,
        &serialNumber                   UM3CharacterString,
        &presentValue                   PresentAnalogControlValue
}
```

AnalogControl 클래스는 ControlBase 클래스와 UM3Base 클래스로부터 그 속성을 상속받습니다. 즉, 상기 ASN.1 정규표현식으로 정의된 각 애트리뷰트의 상세정의는 ControlBase 클래스와 UM3Base 클래스를 참조해야 합니다.

The AnalogControl class inherits the attributes from SensorBase class and UM3Base class. That means that the ControlBase class and UM3Base class should be referred for the detailed definition of attributes defined with the above ASN.1 regular expression.

10.26.1. presentValue Attribute

제어장치의 현재 값을 갖고 있는 애트리뷰트 입니다.

This attribute is used to indicate the current value of the control device.

10.26.2. ACTION

AnalogControl 클래스에서 별도로 다시 정의된 액션은 없습니다.

There is no action separately defined in the AnalogControl class.

10.27. PresentAnalogControlValue Class

애널로그 방식의 제어장치를 운용할 때 해당 제어장치로 입력된 현재의 값을 나타내기 위한 클래스 타입 입니다.

This class is used to represent the current value used as input for the corresponding control device when an analog type control device is used.

표 31-PresentAnalogControlValue 클래스의 애트리뷰트 구성 [Attributes of PresentAnalogControlValue class]

Attribute	Type	M/O
presentValue	UM3Real	M
presentValueTimestamp	UM3DateTime	M
previousValue	UM3Real	M
previousValueTimestamp	UM3DateTime	M
maxPresentValue	UM3Real	M
minPresentValue	UM3Real	M
resolution	UM3Real	M
units	UM3CharacterString	M
acceptableMaxPresentValue	UM3Real	M
acceptableMinPresentValue	UM3Real	M

다음은 PresentAnalogControlValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentAnalogControlValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

PresentAnalogControlValue UM3 OBJECT CLASS
DERIVED FROM
UM3Base;
CHARACTERIZED BY
um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 값, 현재의 값을 측정한 시각, 허용최대값, 허용최소값으로 특징지워짐;
um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and they are characterized by current value, time that current value is taken, upper limit value, and lower limit value;
ATTRIBUTE NAME AS

presentValue	presentValue,
presentValueTimestamp	presentValueTimestamp,
previousValue	previousValue,
previousValueTimestamp	previousValueTimestamp,
maxPresentValue	maxPresentValue,
minPresentValue	minPresentValue,
resolution	resolution,
Units	units,
acceptableMaxPresentValue	acceptableMaxPresentValue,
acceptableMinPresentValue	acceptableMinPresentValue;

ACTION DEFINED AS
None;
BEHAVIOR
애널로그 방식 제어장치의 현재 상태값 관련 정보를 저장하고 관리;
This stores and manages the information related to the current states of an analog type control device.

;;

다음은 PresentAnalogControlValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentAnalogControlValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentAnalogControlValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3Real,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3Real,
        &previousValueTimestamp         UM3DateTime,
        &maxPresentValue                UM3Real,
        &minPresentValue                UM3Real,
        &resolution                     UM3Real,
        &units                          UM3CharacterString,
        &acceptableMaxPresentValue      UM3Real,
        &acceptableMinPresentValue      UM3Real
}
```

10.27.1. presentValue Attribute

해당 애트리뷰트의 타입은 UM3Real 타입이며 가장 최근에 제어장치로 입력된 값 혹은 현재 제어장치의 상태 값을 나타냅니다.

This attribute is UM3Real type, and it contains the most recent value used as input for the control device or current state of the control device.

10.27.2. presentValueTimestamp Attribute

상기 presentValue 애트리뷰트의 측정일시를 나타냅니다.

This contains the date and time when the above presentValue was measured.

10.27.3. previousValue Attribute

해당 애트리뷰트의 타입은 UM3Real 타입이며 presentValue 애트리뷰트의 값을 측정하기 전의 presentValue 를 나타냅니다. 에이전트나 혹은 매니저는 presentValue 를 획득하거나 혹은 새롭게 입력하였을 경우 현재 presentValue 를 previousValue 애트리뷰트의 값으로 갱신하고 presentValue 에 최근에 읽

어들인 값을 기록합니다.

This attribute is UM3Real type, and it contains the presentValue before measuring the value of the presentValue attribute. When an agent or manager obtains presentValue, it updates the presentValue with previousValue and stores the last read value into presentValue.

10.27.4. previousValueTimestamp Attribute

상기 previousValue 를 읽어들인 일시를 나타내며 presentValueTimestamp 의 값을 previousValueTimestamp 로 복사한 후 presentValueTimestamp 값을 기록합니다.

This contains the date and time when the above previousValue was read. The value of presentValueTimestamp is copied to the previousValueTimestamp, and the value of presentValueTimesamp is stored.

10.27.5. maxPresentValue Attribute

서비스를 시작한 이후 지금까지 제어장치로 입력한 값들 중 최대값을 뜻합니다.

This contains the maximum value among the values used as input for the control device since the start of service until now.

10.27.6. minPresentValue Attribute

서비스를 시작한 이후 지금까지 제어장치로 입력한 값들 중 최소값을 뜻합니다.

This contains the minimum value among the values used as input for the control device since the start of service until now.

10.27.7. resolution Attribute

분해능을 나타냅니다. 즉, 제어장치로 입력하는 입력값의 정밀도 혹은 구분가능한 최소단위의 값을 의미합니다.

This contains the resolution, which means precision of values used as input for the control device or value of the minimum unit that can be resolved.

10.27.8. units Attribute

애널로그 값의 단위를 나타냅니다. 타입은 UM3CharacterString 타입입니다.

This contains the unit of analog value. The type is UM3CharacterString.

10.27.9. acceptableMaxPresentValue Attribute

presentValue 애트리뷰트의 값이 갖고 있을 수 있는 최대값을 의미합니다. presentValue 애트리뷰트의 값이 maxPresentValue 애트리뷰트의 값보다 더 클경우에는 제어장치에 고장을 일으킬 확률이 높거나 혹은 제어장치를 제어할 수 없는 경우를 나타냅니다. 따라서, 해당 애트리뷰트의 값보다 큰 값이 입력으로 요청될 경우 에이전트는 에러를 리턴해야 합니다.

해당 애트리뷰트는 임계치 (threshold value) 와는 다른 개념임에 주의해야 합니다.

This contains the maximum value that the value of the presentValue attribute is allowed to have. If the value of presentValue attribute is greater than the value of maxPresentValue attribute, then it is likely to cause failure of the control device or disable the control device. Therefore, the agent should return an error if the requested input value is greater than the value of this attribute.

Caution is required because this attribute is a different concept than the threshold value.

10.27.10. acceptableMinPresentValue Attribute

presentValue 애트리뷰트의 값이 갖고 있을 수 있는 최소값을 의미합니다. 해당 값 보다 작은 값을 받아들일 수 없는 제어장치임을 나타내며, 상기 acceptableMaxPresentValue 애트리뷰트와 마찬가지로 에이전트는 해당 애트리뷰트의 값보다 더 작은 값이 입력값으로 요청될 경우 이를 차단할 수 있어야 합니다.

This contains the minimum value that the value of the presentValue attribute is allowed to have. This means the control device cannot accept a value smaller than this value, and the agent should be able to block the request with the input value smaller this value as in the above acceptableMaxPresentValue attribute.

10.27.11. ACTION

PresentAnalogControlValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentAnalogControlValue class.

10.28. BinaryControl Class

디지털 방식의 입력 신호를 받아들여 제어장치를 제어하는 경우 이를 BinaryControl 장치로 모델링합니다. 즉, 0 혹은 1 로 모델링할 수 있는 값을 받아들여 장치를 제어하는 경우로서 on/off, open/close, high/low 등의 상태를 유지할 수 있는 물리적 자원 혹은 논리적 자원을 모델링 한 경우입니다.

BinaryControl 클래스는 다음과 같은 애트리뷰트로 구성됩니다.

This class is used for modeling the BinaryControl device when a digital signal is used for input to control the control device. It receives the value that can be modeled as 0 or 1, and it models the physical or logical resources that can maintain states like on/off, open/close, or high/low.

The BinaryControl class has the following attributes.

표 32-BinaryControl 클래스의 애트리뷰트 구성 [Attributes of BinaryControl class]

Attribute	Type	M/O
presentValue	PresentBinaryControlValue	M

다음은 BinaryControl 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

BinaryControl class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
BinaryControl UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base, ControlBase;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 presentValue 애트리뷰
            트의 값으로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized as the value of the presentValue attribute;
        ATTRIBUTE NAME AS
            presentValue                        presentValue;

        ACTION DEFINED AS
            None;
        BEHAVIOR
            상위 게이트웨이가 결선을 통해 제어장치를 제어 혹은 네트워크를 통해 제어장치를 제어;
            The control device is controlled by upper level gateway through a cable or network.
;;
```

다음은 AnalogControl 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The BinaryControl class can be defined as follows by using ASN.1 regular expressions.

```
BinaryControl ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
```

```
        &um3AttributeList                UM3ObjectList,
        &manufacturerInfo                CompanyInfo OPTIONAL,
        &vendorInfo                      CompanyInfo OPTIONAL,
        &maintenancePersonel             PersonInfo OPTIONAL,
        &operationalCondition            DeviceOperationalCondition OPTIONAL,
        &hasBattery                      UM3Boolean,
        &batteryInfo                     InstalledBattery OPTIONAL,
        &hasBackupBattery                UM3Boolean,
        &backupBatteryInfo               InstalledBattery OPTIONAL,
        &presentBatteryValueInfo         PresentBatteryValue,
        &presentBackupBatteryValueInfo   PresentBatteryValue OPTIONAL,
        &installedEnvironment            InstalledEnvironment OPTIONAL,
        &powerConsumption                UM3Real,
        &levelInAConfigurationTree       UM3UnsignedInteger16,
        &upperGatewayAddress             UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo             DeviceMaintenanceScheduleInfo,
        &presentControlStatus            DeviceOperationStatus,
        &serialNumber                    UM3CharacterString,
        &presentValue                    PresentBinaryControlValue
}
```

BinaryControl 클래스는 ControlBase 클래스와 UM3Base 클래스로부터 그 속성을 상속받습니다. 즉, 상기 ASN.1 정규표현식으로 정의된 각 애트리뷰트의 상세정의는 ControlBase 클래스와 UM3Base 클래스를 참조해야 합니다.

The BinaryControl class inherits the attributes from the SensorBase class and UM3Base class. That means that the ControlBase class and UM3Base class should be referred for the detailed definition of attributes defined with the above ASN.1 regular expression.

10.28.1. presentValue Attribute

제어장치의 현재 값 혹은 상태정보를 갖고 있는 애트리뷰트 입니다.

This attribute is used to indicate the current value or state information of the control device.

10.28.2. ACTION

BinaryControl 클래스에서 별도로 다시 정의된 액션은 없습니다.

There is no action separately defined in the AnalogControl class.

10.29. PresentBinaryControlValue Class

디지틀 방식의 제어장치를 운용할 때 해당 제어장치로 인가된 입력 값 혹은 현재의 상태 등을 나타내

기 위한 클래스입니다.

This class is used to represent the current value used as input for the corresponding control device when a digital type control device is used.

표 33-PresentBinaryControlValue 클래스의 애트리뷰트 구성 [Attributes of PresentBinaryControlValue class]

Attribute	Type	M/O
presentValue	UM3Boolean	M
presentValueTimestamp	UM3DateTime	M
previousValue	UM3Boolean	M
previousValueTimestamp	UM3DateTime	M
stateChangedCount	UM3UnsignedInteger16	M
stateCountStartedDateTime	UM3DateTime	M

다음은 PresentBinaryControlValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentBinaryControlValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PresentBinaryControlValue UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 값, 현재의 값
        을 측정한 시각, 현재값의 변화 횟수 등으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by current value, time that current value is taken, and the frequency of
        changes in the current value;
    ATTRIBUTE NAME AS
        presentValue                    presentValue,
        presentValueTimestamp           presentValueTimestamp,
        previousValue                   previousValue,
        previousValueTimestamp          previousValueTimestamp,
        stateChangedCount               stateChangedCount,
        stateCountStartedDateTime       stateCountStartedDateTime;
    ACTION DEFINED AS
        None;
    BEHAVIOR
        바이너리 방식 제어장치의 현재 상태값 관련 정보를 저장하고 관리;
        This stores and manages the information related to the current states of the binary type control
        device;
;;
```

다음은 PresentBinaryControlValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentBinaryControlValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentBinaryControlValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3Boolean,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3Boolean,
        &previousValueTimestamp         UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}
```

10.29.1. presentValue Attribute

해당 애트리뷰트의 타입은 UM3Boolean 타입이며 가장 최근에 제어장치로 입력된 상태 값 즉, 현재 제어장치의 상태값을 나타냅니다.

This attribute is UM3Boolean type, and it contains the most recent value used as input for the control device or current state of the control device.

10.29.2. presentValueTimestamp Attribute

상기 presentValue 애트리뷰트의 값을 인가한 일시를 나타냅니다.

This contains the date and time when the above presentValue was applied.

10.29.3. previousValue Attribute

해당 애트리뷰트의 타입은 UM3Boolean 타입이며 presentValue 애트리뷰트의 값을 인가하기 전의 presentValue 를 나타냅니다. 에이전트나 혹은 매니저는 presentValue 를 인가하였을 경우 현재 presentValue 를 previousValue 애트리뷰트의 값으로 갱신하고 presentValue 에는 가장 최근에 입력한 값을 기록합니다.

This attribute is UM3Boolean type, and it contains the presentValue before measuring the value of the presentValue attribute. When an agent or manager obtains presentValue, it updates the presentValue with previousValue and stores

the last read value into presentValue.

10.29.4. previousValueTimestamp Attribute

상기 previousValue 를 입력으로 인가한 일시를 나타내며 presentValueTimestamp 의 값을 previousValueTimestamp 로 복사한 후 presentValueTimestamp 값을 기록합니다.

This contains the date and time when the above previousValue was read. The value of presentValueTimestamp is copied to the previousValueTimestamp, and the value of presentValueTimesamp is stored.

10.29.5. stateChangedCount Attribute

제어를 시작한 이후 지금까지 바이너리 값이 변화한 횟수를 나타냅니다. 제어를 시작한 시점은 stateCountStartedDateTime 애트리뷰트에 기록됩니다.

This contains the frequency of changes in binary value from the start of measurement until now. The time when the measurement was started is stored in the stateCountStartedDateTime attribute.

10.29.6. stateCountStartedDateTime Attribute

상기 stateChangedCount 애트리뷰트의 값 즉, presentValue 애트리뷰트의 값이 변화한 횟수를 기록하기 시작한 시점을 나타냅니다. 거의 대부분 해당 제어장치의 상태를 리셋한 시점과 동일한 시점을 가리키게 됩니다.

This contains the time when the measurement of value of stateChangedCount attribute, frequency of changes in value of presentValue attributes, is started. In most cases, it is the same as the type when the corresponding control device is reset.

10.29.7. ACTION

PresentBinaryControlValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentBinaryControlValue class.

10.30. MultiStateControl Class

MultiStateControl 클래스는 다 단계의 값으로 그 출력 혹은 형상을 제어할 수 있는 물리적 자원을 모델링한 클래스입니다. 입력 값을 여러단계로 나누어 인가할 수 있으며 그에 따른 제어장치의 출력은 여러 단계의 입력값에 대응하는 여러 단계의 출력으로 나타나게 됩니다.

The MultiStateControl class is for modeling physical resources that control the output or configuration using the values with multiple states. The input value can be applied in many states, and the output of the control device has multiple states corresponding to the multiple states of input.

표 34-MultiStateControl 클래스의 애트리뷰트 구성 [Attributes of MultiStateControl class]

Attribute	Type	M/O
presentValue	PresentMultiStateControlValue	M

다음은 MultiStateControl 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The MultiStateControl class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
MultiStateControl UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base, ControlBase;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 presentValue 애트리뷰
            트의 값 즉, 단계별 상태 값으로 특징지워짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized as the value of the presentValue attribute i.e. value of each state;
        ATTRIBUTE NAME AS
            presentValue                        presentValue;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            다단계로 구분된 상태값에 따라 단계별로 상이한 형태로 상태값을 출력, 제어를 위한 입력값
            은 정의된 각 단계별 값으로 구성;
            The output states depend on the input that has multiple states, and the input value for control is
            configured with the value of each state.
;;
```

다음은 MultiStateControl 클래스의 ASN.1 정규표현식에 의한 정의입니다.

MultiStateControl class can be defined using ASN.1 regular expression as follows.

```
MultiStateControl ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
```

```
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentSensorStatus            DeviceOperationStatus,
        &serialNumber                   UM3CharacterString,
        &presentValue                   PresentMultiStateControlValue
}
```

MultiStateControl 클래스는 UM3Base 클래스와 ControlBase 클래스로부터 그 속성을 상속받는 것으로 정의되어 있습니다.

As in the AnalogSensor class, MultiStateControl class inherits the attributes from ControlBase class and UM3Base class.

10.30.1. presentValue Attribute

제어장치의 현재 상태값을 갖고 있는 애트리뷰트 입니다.

This attribute is used to indicate the current state of the control device.

10.30.2. ACTION

MultiStateControl 클래스에서 별도로 정의된 액션은 없습니다.

There is no action separately defined in the MultiStateControl class.

10.31. PresentMultiStateControlValue Class

PresentMultiStateControlValue 클래스는 다단계의 입력값으로 제어되는 MultiStateControl 의 현재 상태값

을 저장 관리하기 위한 클래스 타입입니다. 현재의 상태 값을 저장하기 위해서는 제어장치가 갖고 있는 상태 값의 구분단계에 관한정보, 현재의 값, 이전의 상태의 값 등에 관한정보를 함께 갖고 있어야 합니다.

This class is used to store and manage the current state of the MultiStateControl controlled by input value with multiple states. To store the current state, the information about the classification of stages for each state, current value, and previous state of the control device is also required.

표 35-PresentMultiStateControlValue 클래스의 애트리뷰트 구성
[Attributes of PresentMultiStateControlValue class]

Attribute	Type	M/O
presentValue	UM3UnsignedInteger16	M
presentValueTimestamp	UM3DateTime	M
previousValue	UM3UnsignedInteger16	M
previousValueTimestamp	UM3DateTime	M
numberOfStates	UM3UnsignedInteger16	M
stateNames	UM3 SEQUENCE OF UM3CharacterString	M
stateChangedCount	UM3UnsignedInteger16	M
stateCountStartedDateTime	UM3DateTime	M

다음은 PresentMultiStateControlValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

PresentMultiStateControlValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PresentMultiStateControlValue UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 상태값, 상태
        의 이름, 상태의 개수 등으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by current value, name of state, and number of states;
    ATTRIBUTE NAME AS
        presentValue                    presentValue,
        presentValueTimestamp           presentValueTimestamp,
        previousValue                   previousValue,
        previousValueTimestamp          previousValueTimestamp,
        numberOfStates                  numberOfStates,
        stateNames                      stateNames,
        updatePeriod                    updatePeriod,
```

stateChangedCount stateChangedCount,
stateCountStartedDateTime stateCountStartedDateTime;
ACTION DEFINED AS
None;
BEHAVIOR
단계별 상태 값을 갖는 센서의 측정 값 관련 정보를 저장하고 관리;
This stores and manages the information related to the measurement value of sensor that has state values by stage;
;;

다음은 PresentMultiStateControlValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentMultiStateControlValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentMultiStateControlValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3UnsignedInteger16,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3UnsignedInteger16,
        &previousValueTimestamp         UM3DateTime,
        &numberOfStates                 UM3UnsignedInteger16,
        &stateNames                     UM3 SEQUENCE OF UM3CharacterString,
        &updatePeriod                   UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}
```

10.31.1. presentValue Attribute

해당 애트리뷰트의 타입은 UM3UnsignedInteger16 타입이며 가장 최근에 센서가 측정한 값을 나타냅니다. presentValue 애트리뷰트의 값은 numberOfStates 애트리뷰트의 값보다 클 수 없으며 numberOfStates 애트리뷰트의 값이 n 이라면 그 값의 범위는 0 에서 $(n - 1)$ 로 정의됩니다.

This attribute is UM3UnsignedInteger16 type, and it contains the value most recently measured by the sensor. The value of the presentValue attribute cannot be greater than the value of numberOfState attribute. When the value of numberOfState attribute is n, the range of the value can be defined as $0 \sim (n - 1)$.

10.31.2. presentValueTimestamp Attribute

상기 presentValue 애트리뷰트의 측정일시를 나타냅니다.

This contains the date and time when the above presentValue was most recently measured.

10.31.3. previousValue Attribute

presentValue 애트리뷰트의 값을 측정하기 전의 presentValue 를 나타냅니다. 에이전트나 혹은 매니저는 presentValue 를 획득하였을 경우 현재 presentValue 를 previousValue 애트리뷰트의 값으로 갱신하고 presentValue 에 최근에 읽어들인 값을 기록합니다.

This contains the presentValue before measuring the value of the presentValue attribute. When an agent or manager obtains presentValue, it updates presentValue with previousValue and stores the last read presentValue.

10.31.4. previousValueTimestamp Attribute

상기 previousValue 를 읽어들인 일시를 나타내며 presentValueTimestamp 의 값을 previousValueTimestamp 로 복사한 후 presentValueTimestamp 값을 기록합니다.

This contains the date and time when the above previousValue was read. The value of presentValueTimestamp is copied to the previousValueTimestamp, and the value of presentValueTimesamp is stored.

10.31.5. numberOfStates Attribute

MultiStateSensor 의 단계별 출력 값이 몇 단계로 나누어지는가를 나타내는 애트리뷰트입니다.

This attribute indicates the number of states of the MultiStateControl output.

10.31.6. stateNames Attribute

각 단계별 명칭을 나타냅니다. 해당 애트리뷰트의 타입은 UM3 SEQUENCE OF UM3CharacterString 타입입니다. 첫 번째 엘리먼트의 인덱스는 0 부터 시작합니다. 해당 애트리뷰트은 현재의 상태값을 표현하기 위한 보조적인 수단으로 활용됩니다.

This contains the name of each state. The type of this attribute is UM3 SEQUENCE OF UM3CharacterString. The index of the first element is 0. This attribute is used as an auxiliary means of representing the current state.

10.31.7. updatePeriod Attribute

presentValue 를 업데이트하기 위한 주기를 나타냅니다. 타입은 UM3DateTime 이며 그 단위는 초 입니

다.

This contains the period to update presentValue. The type is UM3DateTime and the unit is seconds.

10.31.8. stateChangedCount Attribute

서비스를 시작한 이후 상태값이 변화한 횟수를 나타냅니다.

This contains total number of change of states since the start of service.

10.31.9. stateCountStartedDateTime Attribute

상기 stateChangedCount 애트리뷰트 값을 변화시키기 시작한 일시를 나타냅니다. 혹은 제어장치를 리셋한 후 새롭게 서비스를 시작한 시점을 나타냅니다.

This contains the time when the above state change monitoring was started, or it contains the time when the new service is started after resetting the control device.

10.31.10. ACTION

PresentMultiStateControlValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentMultiStateControlValue class.

10.32. DeviceOperationStatus Class

특정 장치의 현재 상태를 나타내기 위한 클래스 입니다. 특히 게이트웨이, 센서, 제어장치 등의 현재 상태가 서비스를 제공하지 않는 out of service, 고장 상태인 out of order, 서비스 점검 중인 in maintenance 등의 상태등의 정보를 갖고 있습니다.

This class is for showing the current state of a specific device. In particular, it contains the information of the gateway, sensor, and control device such as out of service, when the service currently cannot be provided, out of order, when devices are out of order, and in maintenance, when the system is under maintenance.

표 36-DeviceOperationStatus 클래스의 애트리뷰트 구성 [Attributes of DeviceOperationStatus class]

Attribute	Type	M/O
presentStatus	UM3Enumerated	M
statusTimestamp	UM3DateTime	M
resumeDateTime	UM3DateTime	M

표 37 은 표 36-DeviceOperationStatus 클래스의 애트리뷰트 구성 의 presentStatus 애트리뷰트의 상태 값의 종류를 나타냅니다. presentStatus 는 UM3Enumerated 타입으로 정의되어 있습니다.

Table 37 shows the type of presentStatus attribute value shown in Table 36, and the presentStatus is defined with UM3Enumerated type.

다음은 DeviceOperationStatus 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

DeviceOperationStatus class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
DeviceOperationStatus UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base UM3 OBJECT CLASS;
        CHARACTERIZED BY
            um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재의 상태값으로 구
            분되어짐;
            um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
            they are characterized by current state value;
        ATTRIBUTE NAME AS
            presentStatus                       presentStatus,
            statusTimestamp                     statusTimestamp,
            resumeDateTime                      resumeDateTime;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            장치의 현재 상태 값을 저장 관리;
            This stores and manages the current state of device.
;;
```

다음은 DeviceOperationStatus 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The DeviceOperationStatus class can be defined as follows by using ASN.1 regular expressions.

```
DeviceOperationStatus ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentStatus                  UM3Enumerated {
                                            outOfService            (0),
```

```
                                    inService               (1),
                                    inMaintenance           (2),
                                    inFault                 (3),
                                    communicationFailure    (4),
                                    unknownCause            (5),
                                    detached                (6)
                                },
        &statusTimestamp            UM3DateTime,
        &resumeDateTime             UM3DateTime
}
```

10.32.1. presentStatus Attribute

presentStatus 애트리뷰트는 특정 장치의 현재 상태를 나타냅니다. 표 37은 presentStatus 애트리뷰트가 가질 수 있는 상태값을 나타내고 있습니다. outOfService 는 서비스를 중단한 상태이며, 그 상태가 inService 상태로 다시 돌아간다는 보장이 없는 상태임을 나타냅니다. inService 는 현재 정상적인 서비스 제공 상태임을 나타냅니다. inMaintenance 는 현재 점검을 위해 서비스를 제공하지 않는 상태입니다. 특히 해당 장치가 게이트웨이, 센서 및 원격 제어장치 등이라면 현장에서 장치의 인터페이스를 통해서 해당 애트리뷰트의 값을 수동으로 설정하게 해야 합니다. 물론 원격의 서버에 설치되어 있는 매니저가 에이전트를 통해 해당 애트리뷰트의 값을 inMaintenance 등으로 변경하는 것도 가능해야 합니다.

The presentStatus attribute contains the current state of specific device. Table 37 shows the states that presentStatus attribute is allowed to have. outOfService means that the service is suspended, and there is no guarantee that the service will return to the inService state. inService means that the service is currently provided as normal. inMaintenance means that service is currently not provided for maintenance reasons. Especially if the corresponding device is a gateway, sensor, or remote control device, then the value of this attribute should be manually set through the interface of the device from the field. The system should support the mode to change the value of the corresponding attribute value to the one like inMaintenance through the agent by the manager installed in the remote server.

표 37-presentStatus 애트리뷰트의 상태 값 [States of presentStatus attributes]

identifier	Meaning	Value
outOfService	Service is suspended. Service may resume.	0
inService	Service is provided normally	1
inMaintenance	Under maintenance i.e. service it not provided	2
inFault	System fault	3
communicationFailure	Communication fault	4
unknownCause	Service is not provided due to unknown reason	5
detached	Cable is detached or communication line is not connect	6

inFault 는 outOfOrder 와 동일한 상태로 볼 수 있으며, 고장 발생 상태로 볼 수 있습니다. communicationFailure 는 해당 장치와의 통신이 불가능한 상태입니다. 이는 대부분 무선네트워크 구간의 에러에 의해 발생하는 상태값입니다. unknownCause 는 그 원인을 알 수 없는 서비스 중단 상태임을 나타냅니다. 대부분의 경우 unknownCause 는 서비스가 다시 원상복구되어야 한다는 의미를 내포하고 있는 것으로 간주되어야 합니다. detached 상태는 결선을 해제한 경우 즉, 연결을 끊은 경우로서 통신 케이블과 단절 시킨 경우를 나타내기도 합니다.

inFault can be considered as the same state as outOfOrder, and it indicates system trouble. communicationFailure means that communication with the corresponding device cannot be established. This usually occurs due to an error in the wireless network zone. UnknownCause means that service has been interrupted due to some unknown reason. In most cases, the unknownCause implies that the service should be restored. detached state means that a cable is disconnected or connection is disabled, and it also means that a communication cable is disconnected.

10.32.2. statusTimestamp Attribute

상기 presentStatus 애트리뷰트의 값이 설정된 시각을 나타냅니다.

This contains the time that the above presentStatus attribute value is set.

10.32.3. resumeDateTime Attribute

만약 상기 presentStatus 애트리뷰트 값이 inMaintenance 등 서비스의 재기동을 내포하고 있는 상태들 중의 하나라면, 해당 장치의 재기동 예상 시각을 그 값으로 갖고 있게 됩니다.

If the value of the above presentStatus attribute is one of the states that imply a service restart such as inMaintenance, then this attribute contains the scheduled resume time of the corresponding device as its value.

10.32.4. ACTION

DeviceOperationStatus 클래스에는 추가로 정의된 액션이 없습니다.

There is no action separately defined in the DeviceOperationStatus class.

10.33. DeviceMaintenanceScheduleInfo Class

서비스의 정상적인 운영을 위한 각종 장치들에 대한 오프라인 혹은 온라인 점검 주기, 시각 등에 관한 정보를 나타내는 클래스입니다.

This class is used to store information required for normal operation of the service such as the offline or online

inspection period and scheduled inspection time.

표 38-DeviceMaintenanceScheduleInfo 클래스의 애트리뷰트 구성
[Attributes of DeviceMaintenanceScheduleInfo class]

Attribute	Type	M/O
inServiceDateTime	UM3DateTime	M
latestMaintenanceDateTime	UM3DateTime	O
nextMaintenanceDateTime	UM3DateTime	O
durablePeriod	UM3DateTime	M
maintenancePeriod	UM3DateTime	O
numberOfRebooting	UM3UnsignedInteger16	O
shutdownScheduleInfo	UM3DateTime	O

다음은 DeviceMaintenanceScheduleInfo 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

DeviceMaintenanceScheduleInfo class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows

```
DeviceMaintenanceScheduleInfo UM3 OBJECT CLASS
    DERIVED FROM
        None;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 점검주기, 점검일자,
        shutdown 등의 일정으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by inspection period, inspection date, and shutdown schedule;
    ATTRIBUTE NAME AS
        inServiceDateTime                inServiceDateTime,
        latestMaintenanceDateTime        latestMaintenanceDateTime,
        nextMaintenanceDateTime          nextMaintenanceDateTime,
        durablePeriod                    durablePeriod,
        maintenancePeriod                maintenancePeriod,
        numberOfRebooting                numberOfRebooting,
        shutdownScheduleInfo             shutdownScheduleInfo;
    ACTION DEFINED AS
        None;
    BEHAVIOR
        유지보수 등을 위한 주요 일정정보를 저장 관리;
        This stores and manages the major schedule information for maintenance;
;;
```

다음은 DeviceMaintenanceScheduleInfo 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The DeviceMaintenanceScheduleInfo class can be defined as follows by using ASN.1 regular expressions.

```
DeviceMaintenanceScheduleInfo ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &inServiceDateTime              UM3DateTime,
        &latestMaintenanceDateTime      UM3DateTime OPTIONAL,
        &nextMaintenanceDateTime        UM3DateTime OPTIONAL,
        &durablePeriod                  UM3DateTime,
        &maintenancePeriod              UM3DateTime OPTIONAL,
        &numberOfRebooting              UM3UnsignedInteger16 OPTIONAL,
        &shutdownScheduleInfo           UM3DateTime OPTIONAL
}
```

10.33.1. inServiceDateTime Attribute

해당 애트리뷰트를 갖고 있는 클래스로 모델링한 특정 장치의 서비스 시작 일시를 나타냅니다.

This contains the start date of the specific device modeled by the class with the corresponding attributes.

10.33.2. latestMaintenanceDateTime Attribute

최근 마지막으로 유지보수활동 혹은 점검을 실시한 일시를 나타냅니다.

This contains the date of the last maintenance activity or inspection.

10.33.3. nextMaintenanceDateTime Attribute

다음 유지보수활동에 따른 점검 예정 일시를 나타냅니다.

This contains the schedule for the next maintenance activity.

10.33.4. durablePeriod Attribute

해당 애트리뷰트는 inServiceDateTime 애트리뷰트의 값으로 기록된 일시로부터 사용가능한 기간 혹은

감가상각비가 0 이 되는 기간을 나타냅니다.

This contains the period from the date and time in the inServiceDateTime for the useable period or until the depreciated value becomes 0.

10.33.5. maintenancePeriod Attribute

점검주기를 나타내며 UM3DateTime 타입으로 기록합니다. 해당 애트리뷰트는 초 단위로 저장되며 적절한 단위로 변경 후 사용해야 합니다.

This contains the maintenance schedule, stored in UM3DateTime type. It is stored in the unit of seconds and it should be converted to the proper format before use.

10.33.6. numberOfRebooting Attribute

inServiceDateTime 애트리뷰트에 기록된 서비스 시작일로부터 지금까지 전원을 다시 켠 회수를 나타냅니다.

This indicates the frequency of rebooting from the service start date recorded in the inServiceDateTime attribute until now.

10.33.7. shutdownScheduleInfo Attribute

다음 rebooting 혹은 shutdown 관련 일정을 나타내는 애트리뷰트 입니다.

This contains the schedule for the next reboot or shutdown.

10.33.8. ACTION

DeviceMaintenanceScheduleInfo 클래스에서 새롭게 정의된 액션은 없습니다.

There is no action separately defined in the DeviceMaintenanceScheduleInfo class.

10.34. PresentGatewayValue Class

게이트웨이 장치가 운용 중인 상태일 경우 해당 게이트웨이의 현재 상태 정보를 나타내기 위한 클래스입니다. 해당 클래스 오브젝트의 값들은 원격장치의 상태값을 확인하고 처리하기 위한 서비스관리 정보모델을 구성하는 주요 오브젝트들 중의 하나입니다.

This class is used to store the current state information of the corresponding gateway when the gateway device is in operation. The values of the corresponding objects are major objects configured in the service management information model for checking and processing the status value of remote devices.

표 39. PresentGatewayValue 클래스의 애트리뷰트 구성 [Attributes of PresentGatewayValue class]

Attribute	Type	M/O
kbytesMemoryInUse	UM3Real	M
kbytesMemoryInUseTimestamp	UM3DateTime	M
mbytesHarddiskInUse	UM3Real	O
mbytesHarddiskInUseTimestamp	UM3DateTime	O
percentCpuTime	UM3Real	M
percentCpuTimeTimestamp	UM3DateTime	M
bytesSentPerSecond	UM3UnsignedInteger32	M
bytesReceivedPerSecond	UM3UnsignedInteger32	M
bytesPerSecondTimestamp	UM3DateTime	M

다음은 PresentGatewayValue 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The PresentGatewayValue class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
PresentGatewayValue UM3 OBJECT CLASS
    DERIVED FROM
        None;
    CHARACTERIZED BY
        um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값으로 구분되며 현재 CPU 상태, 메모
        리 사용상태, 네트워크 인터페이스 관련 상태값 등으로 특징지워짐;
        um3ClassIdentifier and um3ObjectName attribute values are used to identify the objects, and
        they are characterized by current CPU state, memory usage, and network interface related
        states;
    ATTRIBUTE NAME AS
        kbytesMemoryInUse                   kbytesMemoryInUse,
        kbytesMemoryInUseTimestamp          kbytesMemoryInUseTimestamp,
        mbytesHarddiskInUse                 mbytesHarddiskInUse,
        mbytesHarddiskInUseTimestamp        mbytesHarddiskInUseTimestamp,
        percentCpuTime                      percentCpuTime,
        percentCpuTimeTimestamp             percentCpuTimeTimestamp,
        bytesSentPerSecond                  bytesSentPerSecond,
        bytesReceivedPerSecond              bytesReceivedPerSecond,
        bytesPerSecondTimestamp             bytesPerSecondTimestamp;
    ACTION DEFINED AS
        None;
    BEHAVIOR
        게이트웨이 장치의 현재 상태 정보 값을 저장 관리;
        This stores and manages the current state of the gateway device.
;;
```

다음은 PresentGatewayValue 클래스의 ASN.1 정규표현식에 의한 정의입니다.

The PresentGatewayValue class can be defined as follows by using ASN.1 regular expressions.

```
PresentGatewayValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &kbytesMemoryInUse              UM3Real,
        &kbytesMemoryInUseTimestamp     UM3DateTime,
        &mbytesHarddiskInUse            UM3Real OPTIONAL,
        &mbytesHarddiskInUseTimestamp   UM3DateTime OPTIONAL,
        &percentCpuTime                 UM3Real,
        &percentCpuTimeTimestamp        UM3DateTime,
        &bytesSentPerSecond             UM3UnsignedInteger32,
        &bytesReceivedPerSecond         UM3UnsignedInteger32,
        &bytesPerSecondTimestamp        UM3DateTime
}
```

10.34.1. kbytesMemoryInUse Attribute

현재 게이트웨이가 사용 중인 메모리의 크기를 나타냅니다. 애트리뷰트의 명칭 혹은 이름에 명기된 것과 같이 그 단위는 Kbyte 입니다. 해당 애트리뷰트의 타입은 UM3Real 타입입니다.

This contains the size of memory currently used by the gateway device. The unit is Kbytes as specified in the name of the attribute. The type of this attribute is UM3Real.

10.34.2. kbytesMemoryInUseTimestamp Attribute

상기 kbytesMemoryInUse 애트리뷰트의 측정시각을 나타냅니다.

This contains the measurement time of the above kbytesMemoryInUse attribute.

10.34.3. mbytesHarddiskInUse Attribute

하드디스크가 장착되어 있는 경우, 해당 하드디스크의 현재 사용량을 나타내며 그 단위는 MByte 입니다. 만약 게이트웨이에 하드디스크가 장착되어 있지 않다면 애트리뷰트에 저장된 값은 신뢰성이 없는 gargabe 데이터가 저장되어 있게 됩니다. 따라서, 반드시 게이트웨이에 하드디스크가 장착되어 있는지의 여부를 먼저 확인하고 해당 애트리뷰트의 값을 참조해야 합니다.

This contains the current use of the corresponding hard disk when a hard disk is installed. The unit is MBytes. If a hard disk is not installed in the gateway device, then the data stored in this attribute will be unreliable garbage. Check whether a hard disk is installed in the gateway before referring to the value of this attribute.

10.34.4. mbytesHarddiskInUseTimestamp Attribute

상기 mbytesHarddiskInUse 애트리뷰트의 값을 측정한 시각을 나타냅니다.

This contains the time when the value of the above mbytesHarddiskInUse is measured.

10.34.5. percentCpuTime Attribute

CPU 의 점유율을 나타내며 그 단위는 % 입니다.

This contains the occupy rate of CPU. The unit is %.

10.34.6. percentCpuTimeTimestamp Attribute

상기 percentCpuTime 애트리뷰트를 측정한 시각을 나타냅니다.

This contains the time when the above percentCpuTime attribute is measured.

10.34.7. bytesSentPerSecond Attribute

네트워크 인터페이스를 이용해 초당 송신한 바이트의 수를 나타내며 그 단위는 byte per second 입니다.

This contains the number of bytes transmitted per second through the network interface, and the unit is bytes per second.

10.34.8. bytesReceivedPerSecond Attribute

네트워크 인터페이스를 통해 초당 수신한 바이트의 수를 나타내며 그 단위는 byte per second 입니다.

This contains the number of bytes received per second through the network interface, and the unit is bytes per second.

10.34.9. bytesPerSecondTimestamp Attribute

상기 bytesSentPerSecond 와 bytesReceivedPerSecond 애트리뷰트의 값을 측정한 시각을 나타냅니다.

This contains the time when the above bytesSentPerSecond and bytesReceivedPerSecond are measured.

10.34.10. ACTION

PresentGatewayValue 클래스에서 추가로 정의된 액션은 없습니다.

There is no action separately defined in the PresentGatewayValue class.

10.35. UM3OperationErrorCode Type

매니저가 에이전트로 전송한 오퍼레이션 APDU 에 따라 에이전트가 필요한 수신절차를 진행할 때, 그 수신절차의 수행 결과가 실패하였을 경우, 그 원인을 기록하여 매니저에게 전송하기 위해 사용되는 타입입니다. UM3OperationErrorCode 타입은 UM3Enumerated 타입과 동일한 타입입니다. 즉, 애트리뷰트를 갖는 클래스 타입이 아닌 UM3 기본 타입들 중 UM3Enumerated 타입으로 정의되는 기본 타입의 클래스입니다.

When the agent performs procedures required to receive data according to operation APDU sent to agent by manager, if the receive procedure fails, then this type is used to send the error code to the manager. UM3OperationErrorCode is the same type as UM3Enumerated. In other words, it is a primitive type class defined with UM3Enumerated type among UM3 primitive type instead of class type with attributes.

표 40-UM3OperationErrorCode 타입의 코드별 의미
[Description of UM3OperationErrorCode Type by Error Codes]

Error Code	Description	Value
accessDenied	UM3-GET, UM3-SET 등의 서비스 오퍼레이션이 적용될 오브젝트에 대한 접근이 거부된 경우 Access to the object to be applied to the service operation such as UM3-GET or UM3-SET is denied.	
notFound	파라메터로 주어진 UM3ClassIdentifier 와 UM3ObjectName 파라메터의 오브젝트 혹은 애트리뷰트 오브젝트를 찾을 수 없는 경우 Object or attribute object with the given parameters of UM3ClassIdentifier and UM3ObjectName cannot be found.	
noSuchClass	특정 클래스가 존재하지 않을 경우 Specific class does not exist.	
noSuchObject	특정 오브젝트가 존재하지 않을 경우 Specific object does not exist.	
noSuchAttribute	특정 애트리뷰트가 존재하지 않을 경우 Specific attribute does not exist.	
invalidAttributeClassType	애트리뷰트의 클래스 타입이 파라메터로 주어진 타입과 다를 경우	

	Class type of the attribute is different from the type of the given parameter.
processingFailure	오퍼레이션의 수행 중 오류가 발생한 경우 There is an error while processing an operation.
unrecognizedOperation	정의되지 않은 오퍼레이션이 요청되었을 경우 Undefined operation is requested.
unknownSignature	정의되지 않은 오퍼레이션 signature 일 경우 Operation signature is undefined.
eventDrivenNotSupported	이벤트 검출 방식을 지원하지 않는 경우 Event driven method is not supported.
noSuchSession	주어진 UM3 session identifier 를 갖는 프로세스가 없는 경우 There is no process corresponding to the given UM3 session identifier.

표 40-UM3OperationErrorCode 타입의 코드별 의미는 UM3OperationErrorCode 타입의 코드 구성을 나타냅니다.

Table 40 shows the codes for UM3OperationErrorCode.

accessDenied 의 경우 특정 오퍼레이션이 적용되어야 할 오브젝트에 대한 접근 자체가 불가능한 경우를 나타냅니다. notFound identifier 는 주어진 오브젝트가 존재하지 않는 경우를 나타내며, noSuchClass 는 주어진 클래스 아이디 값으로 정의된 클래스가 존재하지 않는 경우를 나타냅니다. noSuchObject 는 주어진 오브젝트가 존재하지 않을 경우 즉, UM3ObjectName 이 존재하지 않는 경우를 나타냅니다. noSuchAttribute 는 파라메터로 주어진 애트리뷰트가 존재하지 않는 경우를 나타내며, invalidAttributeClassType 은 파라메터로 주어진 클래스 타입이 애트리뷰트의 클래스 타입과 상이할 경우를 나타냅니다. processingFailure identifier 는 오퍼레이션의 수행 도중 오류가 발생한 경우이며, unrecognizedOperation identifier 는 주어진 오퍼레이션 클래스 아이디가 본 권고안에서 정의되어있지 않은 경우를 나타냅니다. unknownSignature identifier 는 오퍼레이션의 파라메터로 주어진 타입과 값들이 본 권고안에서 정의되지 않은 형태임을 나타냅니다. eventDrivenNotSupported identifier 는 에이전트가 polling 방식 즉, UM3-GET 서비스는 지원하는 반면 UM3-EVENT-REPORT 서비스를 지원하지 않음을 뜻합니다. noSuchSession identifier 는 오퍼레이션의 파라메터로 주어진 UM3 session identifier 를 갖는 프

로세스 가 존재하지 않을 경우를 나타냅니다.

상기 UM3OperationErrorCode 타입의 ASN.1 정규표현식으로의 표현은 다음과 같습니다.

accessDenied means that the access to the object to be applied to a specific operation has been denied. notFound identifier indicates that the specific object cannot be found. noSuchClass means that the class defined with the corresponding class ID does not exist. noSuchObject means that the given object does not exist. In other words, the object corresponding to the UM3ObjectName does not exist. noSuchAttribute means that the attribute given as a parameter does not exist. invalidAttributeClassType means that the class type given as a parameter is different from the class type of the attribute. processingFailure identifier indicates an error during operation, and the unrecognizedOperation identifier indicates that the given operation class ID is not defined in this recommendation. unknownSignature identifier indicates that the type given as parameter of operation is not defined in this recommendation. eventDrivenNotSupported identifier means that the agent supports the polling method i.e. UM3-GET service, but it does not support the UM3-EVENT-REPORT service. noSuchSession identifier indicates that the process with UM3 session identifier that is given as a parameter of operation does not exist.

The UM3OperationErrorCode type can be defined as follows by using ASN.1 regular expressions.

```
UM3OperationErrorCode ::= UM3Enumerated {
        notFound                    (0),
        accessDenied                (1),
        noSuchClass                 (2),
        noSuchObject                (3),
        noSuchAttribute             (4),
        invalidAttributeClassType   (5),
        processingFailure           (6),
        unrecognizedOperation       (7),
        unknownSignature            (8),
        eventDrivenNotSupported     (9),
        noSuchSession               (10)
}
```

10.36. UM3ObjectList Class

UM3ObjectList 클래스 타입은 UM3 SEQUENCE OF 복합 형식의 타입으로 정의합니다. 또한 UM3ObjectList 타입은 일련의 오브젝트들의 연속적인 나열로 표현되는 타입입니다. UM3ObjectList 타입은 UM3 오퍼레이션들 중 GetObjectGroupValue, GetObjectAttributeGroupValue, SetObjectGroupValue, SetObjectAttributeGroupValue 등의 오퍼레이션과 같이 파라메터로 여러 개의 오브젝트 정보를 한 번에 넘겨주어야 할 경우 사용하는 타입입니다. 또한 이와 같은 요청관련 오퍼레이션에 대한 성공 혹은 실

패의 결과를 리턴할 때, OperaetionResponse 오퍼레이션이 여러 개의 오브젝트를 한 번에 전달하기 위해 사용되기도 합니다.

The UM3ObjectList class type is defined as a complex UM3 SEQUENCE OF type, and it is a type that lists a series of objects. The UM3ObjectList type is used when passing the information of multiple objects as a parameter for some operations among UM3 operations such as GetObjectGroupValue, GetObjectAttributeGroupValue, SetObjectGroupValue, and SetObjectAttributeGroupValue. It is also for the OperationResponse operation to pass multiple objects when returning the results of success or failure for the operation related to request.

UM3ObjectList 타입에 대한 ASN.1 정규표현식으로의 정의는 다음과 같습니다.

The UM3ObjectList type can be defined as follows by using ASN.1 regular expression

UM3ObjectList ::= UM3 SEQUENCE OF ANY DEFINED BY UM3 ASN.1 module

위의 ASN.1 정규표현식에 있어서 ANY DEFINED BY UM3 ASN.1 module 은 본 권고안이 정의하는 모든 타입들을 뜻합니다.

UM3ObjectList 타입은 UM3 SEQUENCE OF 타입과 동일한 타입이지만 클래스 아이디는 UM3 SEQUENCE OF 타입과는 별도로 정의되어있는 점에 유의해야 합니다.

ANY DEFINED BY UM3 ASN.1 module in the above ASN.1 regular expression means all types defined in this recommendation.

Caution is required because the UM3ObjectList type is the same as UM3 SEQUENCE OF type, but the class ID is defined separately from the UM3 SEQUENCE OF type.

10.37. UM3ObjectIndicator Type

UM3ObjectIndicator 타입은 특정 오브젝트를 가리키기 위해 해당 오브젝트의 클래스아이디, 오브젝트 이름에 관한 정보를 갖고 있습니다. 해당 오브젝트는 UM3ObjectList 타입의 엘리먼트로 활용될 수 있습니다.

UM3ObjectIndicator 타입의 ASN.1 정규표현식으로의 표현은 아래와 같습니다.

The UM3ObjectIndicator type contains the information about the class ID and name of the corresponding object to point at a specific object. The corresponding object can be used as an element of the UM3ObjectList type.

The UM3ObjectIndicator type can be defined as follows by using ASN.1 regular expression

```
UM3ObjectIndicator ::= UM3 SEQUENCE {
        um3ClassIdentifierIndicator     UM3ClassIdentifier,
        um3ObjectNameIndicator          UM3ObjectName
}
```

UM3ObjectIndicator 타입은 UM3 복합형식 타입이며 클래스아이디는 UM3 SEQUENCE 타입과는 별도로 지정되어 있습니다.

UM3ObjectIndicator is a UM3 complex type, and the class ID is defined separately from UM3 SEQUENCE type.

10.38. UM3ObjectValueCondition Class Type

UM3ObjectValueCondition 클래스는 조건 정보를 저장하기 위한 애트리뷰트들로 구성되며, 오퍼레이션의 파라메터로 전송되거나 혹은, 클래스의 애트리뷰트로 저장되어 특정 오브젝트의 액션 수행에 활용될 수 있습니다.

본 권고안이 정의하는 UM3 프로토콜의 조건은 일반적인 연산에서 사용되는 것과 유사한 형태로 정의합니다.

The UM3ObjectValueCondition class has the attributes for storing condition information, and it can be used for passing parameters in operation or launching an action of a specific object as an attribute of a class.

The conditions of the UM3 protocol defined in this recommendation are defined in a similar way as those used in general computational statements.

1) 기준이 되는 값이 한 개인 경우
   가) 비교대상이 되는 값이 기준이 되는 값보다 큰 경우 (greater than)
   나) 비교대상이 되는 값이 기준이 되는 값보다 작은 경우 (less than)
   다) 비교대상이 되는 값과 기준이 되는 값이 같을 경우 (equal)
2) 기준이 되는 값이 두 개인 경우
   가) 비교대상이 되는 값이 기준이 되는 값들 중 작은 값 보다 작을 경우 (less than lower end)
   나) 비교대상이 되는 값이 기준이 되는 두 개의 값 사이에 있을 경우 (between lower end and upper end; greater than lower end and less than upper end)
   다) 비교대상이 되는 값이 기준이 되는 값들 중 큰 값 보다 클 경우 (greater than upper end)

1) When there is one reference…

A) If the comparison target is greater than the reference (greater than)
B) If the comparison target is smaller than the reference (less than)
C) If the comparison target is equal to the reference (equal)

2) When there are two references
D) If the comparison target is smaller than smaller reference (less than lower end)
E) If the comparison target is between the smaller and bigger reference (between lower end and upper end; greater than lower end and less than upper end)
F) If the comparison target is greater than the bigger reference (greater than upper end)

이상과 같은 값은 애널로그 값 (analog value) 에 대한 조건 연산에 적용할 수 있습니다. 바이너리 값 (binary value) 에 대한 조건 연산은 아래와 같이 두 가지의 경우만 존재할 수 있는 것으로 정의합니다.

These values can be applied to the conditional computation for analog values. Conditional computation for binary values can only be defined according to the following two cases.

1) 기준이 되는 한 개의 값에 대하여,
가) 비교대상이 되는 값이 기준이 되는 값과 같은 경우 (equal)
나) 비교대상이 되는 값이 기준이 되는 값과 같지 않은 경우 (not equal)

1) When there is one reference,
A) If the comparison target is equal to the reference (equal)
B) If the comparison target is not equal to the reference (not equal)

또한 비교대상이 되는 값에 관계없이 해당 애트리뷰트의 값이 변화한다면 이를 조건에 대한 만족으로 정의하는 것도 가능합니다.

If the value of the target attributes change independently from the values to compare, then the following condition can be defined by using the change as condition.

1) 애트리뷰트의 값이 변화하는 경우

1) If the value of attribute changes

UM3ObjectValueCondition 클래스는 앞서 정의한 upper end 값 및 lower end 값, 그리고 해당 값들과의 비교를 통한 조건연산자를 그 애트리뷰트로 갖고 있습니다. 또한 경우에 따라 비교값과 그 조건에 따른 액션을 실행시킬 수 있는 구조로 정의되어 있습니다.

표 41-UM3ObjectValueCondition 클래스의 애트리뷰트 구성 은 UM3ObjectValueCondition 클래스의 애트

리뷰트의 구성을 나타내고 있습니다.

The UM3ObjectValueCondition class has the upper end, lower end, as defined earlier, and conditional operator for comparison as its attributes. Its structure allows the execution of an action based on the result of the evaluation of the conditional statement.

Table 41 shows the attributes of the UM3ObjectValueCondtion class.

표 41-UM3ObjectValueCondition 클래스의 애트리뷰트 구성 [Attributes of UM3ObjectValueCondition class]

Attribute	Class Type	M/O
upperEnd	UM3Integer16, UM3Integer32, UM3Integer64, UM3UnsignedInteger16, UM3UnsignedInteger32, UM3UnsignedInteger64, UM3Real, UM3CharacterString, UM3Boolean, UM3DateTime	M
lowerEnd	UM3Integer16, UM3Integer32, UM3Integer64, UM3UnsignedInteger16, UM3UnsignedInteger32, UM3UnsignedInteger64, UM3Real, UM3CharacterString, UM3Boolean, UM3DateTime	M
condition	UM3Enumerated	M
targetAttributeClassIdentifier	UM3ClassIdentifier	M
targetAttributeObjectName	UM3ObjectName	M

표 41-UM3ObjectValueCondition 클래스의 애트리뷰트 구성의 upperEnd 와 lowerEnd 값은 targetClassIdentifier 와 targetObjectName 으로 이루어지는 특정 오브젝트 혹은 오브젝트 애트리뷰트의 타입에 따라 그 타입이 달라집니다. 단, upperEnd 와 lowerEnd 애트리뷰트의 타입은 동일해야 하며 서로 다른 타입을 가질 수 없습니다.

The type of upperEnd and lowerEnd in Table 41 may change depending on the type of a specific object or type of object attribute that is identified by targetClassIdentifier and targetObjectName. The types of upperEnd and lowerEnd attributes should be same, and they are not allowed to have different ones.

다음은 UM3ObjectValueCondition 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The UM3ObjectValueCondition class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
UM3ObjectValueCondition OBJECT CLASS
    DERIVED FROM
        UM3Base OBJECT CLASS;
    CHARACTERIZED BY
        upperEnd, lowerEnd 및 condition 애트리뷰트로 정의되는 조건;
        Conditions defined by upperEnd, lowerEnd, and condition attributes;
```

ATTRIBUTE NAME AS
upperEnd upperEnd
lowerEnd lowerEnd
condition condition
targetAttributeClassIdentifier targetAttributeClassIdentifier
targetAttributeObjectName targetAttributeObjectName;
ACTION DEFINED AS
NONE;
BEHAVIOR
연산조건 관련 정보를 저장하며 이를 오퍼레이션을 통해 전달하기 위한 클래스;
This class is for storing information related to conditional statements and passing it through operation.
;;

상기 UM3 CLASS DEFINITION SYNTAX 에 정의된 것과 같이, UM3ObjectValueCondition 은 upperEnd, lowerEnd 및 condition 애트리뷰트로 그 특성이 결정됩니다.

As defined in the above UM3 CLASS DEFINITION SYNTAX, the UM3ObjectValueCondition class is characterized by upperEnd, lowerEnd, and condition attributes.

다음은 상기 UM3ObjectValueCondition 클래스의 ASN.1 정규표현식에 의한 표현입니다.

The UM3ObjectValueCondition class can be defined as follows by using ASN.1 regular expressions.

```
UM3ObjectValueCondition ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &upperEnd                       &EndLimitType,
        &lowerEnd                       &EndLimitType,
        &condition                      UM3Enumerated {
                                                LL          (0),
                                                GLLU        (1),
                                                GU          (2),
                                                LEL         (3),
                                                GELaLEU     (4),
                                                GEU         (5),
                                                GELaLU      (6),
                                                GLaLEU      (7),
                                                EU          (8),
                                                EL          (9),
                                                LLoGU       (10),
```

```
                                        LELoGU      (11),
                                        LELoGEU     (12),
                                        LLoGEU      (13),
                                        COV         (14)
                                    },
        &targetAttributeClassIdentifier   UM3ClassIdentifier,
        &targetAttributeObjectName        UM3ObjectName
}


EndLimitType ::=
        UM3Integer16 |
        UM3Integer32 |
        UM3Integer64 |
        UM3UnsignedInteger16 |
        UM3UnsignedInteger32 |
        UM3UnsignedInteger64 |
        UM3Real |
        UM3CharacterString |
        UM3Boolean |
        UM3DateTime
```

상기 ASN.1 정규표현식에 있어서 &upperEnd &EndLimitType 혹은 &lowerEnd &EndLimitType 은 앞서 정의한 바와 같이 upperEnd 와 lowerEnd 애트리뷰트가 취할 수 있는 타입이 EndLimitType 타입의 정의에 나타낸 바와 같이 변화할 수 있음을 나타냅니다.

It is shown in the above ASN.1 regular expression that the type of upperEnd and lowerEnd attribute specified as &upperEnd &EndLimitType and &lowerEnd &EndLimitType can be changed by the definition of the EndLimitType type.

10.38.1. um3ClassIdentifier Attribute

UM3 오퍼레이션의 파라메터로 넘겨지는 해당 오브젝트의 클래스아이디를 나타냅니다. 즉, UM3ObjectValueCondition 클래스의 클래스 아이디가 저장됩니다.

This contains the class ID of the object that is passed as a parameter in UM3 operation. In other words, the class ID of the UM3ObjectValueCondition class is stored.

10.38.2. um3ObjectName Attribute

UM3ObjectValueCondition 오브젝트의 오브젝트 이름이 기록됩니다.

Name of UM3ObjectValueCondition object is stored.

10.38.3. um3ObjectNameAlias Attribute

해당 오브젝트의 별칭이 기록됩니다. 해당 애트리뷰트는 선택요소로 지정되어 있습니다.

Alias of the corresponding object is stored. This attribute is classified as an optional element.

10.38.4. upperEnd Attribute

기준이 되는 두 개의 값들 중 상위 값을 나타냅니다. 표 41-UM3ObjectValueCondition 클래스의 애트리뷰트 구성에 정의된 것과 같이 다음과 같은 값들로 그 타입이 지정될 수 있습니다.

This is the higher value among the two reference values. Type can be assigned with one of the following as defined in Table 41.

- UM3Integer16, UM3Integer32, UM3Integer64
- UM3UnsignedInteger16, UM3UnsignedInteger32, UM3UnsignedInteger64
- UM3Real, UM3Boolean, UM3CharacterString

단, 상기 타입들은 lowerEnd 애트리뷰트와 항상 동일한 타입으로 지정되어야 합니다.

In this case, the type should always be same as the lowerEnd attribute.

10.38.5. lowerEnd Attribute

upperEnd 애트리뷰트와 반대되는 두 개의 기준 값들 중 하위의 기준 값을 나타냅니다. upperEnd 애트리뷰트와 동일한 타입으로 정의되어야 하며, 그 타입은 upperEnd 애트리뷰트와 동일합니다.

This is the lower value of the two references, opposite from upperEnd. The type should be same as that of the upperEnd attribute.

10.38.6. condition Attribute

condition 애트리뷰트는 상기 lowerEnd 와 upperEnd 애트리뷰트에 대한 비교대상 값과의 비교 조건을 UM3Enumerated 타입으로 나타냅니다.

The condition attribute contains the comparison condition between the comparison target value and lowerEnd and upperEnd attributes, and it is UM3Enumerated type.

표 42-조건문의 종류 [Types of conditional statements]

Identifier	Value	Meaning and Description	
LL	0	lessThanLower	X < lowerEnd
GLLU	1	greaterThanLowerAndLessThanUpper	lowerEnd < X < upperEnd
GU	2	greaterThanUpper	upperEnd < X
LEL	3	lessThanOrEqualToLower	X <= lowerEnd
GELaLEU	4	greaterThanOrEqualToLowerAndLessThanOrEqualTo-Upper	lowerEnd <= X <= upperEnd
GEU	5	greaterThanOrEqualToUpper	upperEnd <= X
GELaLU	6	greaterThanOrEqualToLowerAndLessThanUpper	lowerEnd <= X < upperEnd
GLaLEU	7	greaterThanLowerAndLessThanOrEqualToUpper	lowerEnd < X <= upperEnd
EU	8	equalToUpper	X == upperEnd
EL	9	equalToLower	X == lowerEnd
LLoGU	10	lessThanLowerOrGreaterThanUpper	X < lowerEnd OR upperEnd < X
LELoGU	11	lessThanOrEqualToLowerOrGreaterThanUpper	X <= lowerEnd OR upperEnd < X
LELoGEU	12	lessThanOrEqualToLowerOrGreaterThanOrEqualTo-Upper	X <= lowerEnd OR upperEnd <= X
LLoGEU	13	lessThanLowerOrGreaterThanOrEqualToUpper	X < lowerEnd OR upperEnd <= X
COV	14	change of attribute value	change

표 42-조건문의 종류 은 condition 애트리뷰트가 갖는 조건의 종류를 나타냅니다. 각 identifier 는 모든 가능한 조건들을 나열하고 있으며, 각각 LL, GEU 등의 대문자로 표시된 조건과 더불어, 경우에 따라 a 로 표현된 AND 논리 연산자와 o 로 표현된 OR 논리연산자를 나타냅니다. 특히 COV 는 change of value 의 약자로서 비교과정 없이 대상이 되는 애트리뷰트의 값이 변화할 경우 조건이 만족되는 경우로 판단하는 경우입니다.

Table 42 contains the types of conditions that the condition attribute can have. Each identifier lists all possible conditions, containing conditions indicated by upper case letters like LL and GEU, along with logical AND operators shown as a and logical OR operators shown as o for some cases. COV stands for change of value, and it is used to determine whether the condition is satisfied by examining the change of value of attribute without the comparison process.

10.38.7. targetAttributeClassIdentifier Attribute

targetClassIdentifier 애트리뷰트는 상기 조건 들이 적용되어야 하는 애트리뷰트 오브젝트의 타입을 지정하기 위한 애트리뷰트 입니다.

The targetClassIdentifier attribute for assigning the type of attribute object to which the above conditions are applied.

10.38.8. targetAttributeObjectName Attribute

targetObjectName 은 상기 조건들이 적용되어야 하는 애트리뷰트 오브젝트의 이름을 지정하기 위한 애트리뷰트 입니다.

The targetObjectName attribute for assigning name of attribute object to which the above conditions are applied.

10.39. UM3ObjectListToBeCreated Type

UM3ObjectListToBeCreated 타입은 CreateObjectGroup 오퍼레이션을 위한 파라메터로 사용되는 데이터 타입입니다. UM3ObjectListToBeCreated 타입은 생성하고자 하는 오브젝트의 DN 정보를 갖고 있는 UM3ObjectName 타입과 본 권고안이 정의하는 오브젝트 클래스 타입 등 두 가지 타입의 데이터가 연속적으로 나열된 타입의 형식으로 구성 되며 그림 9 에 나타낸 것과 같은 구조를 갖습니다.

UM3ObjectListToBeCreated is a data type used a parameter for CreateObjectGroup operation.
UM3ObjectListToBeCreated has two data types listed in order, including the UM3ObjectName type that contains the DN information of the object to create, as well as the object class type defined in this recommendation. Its structure is shown in Fig. 9.

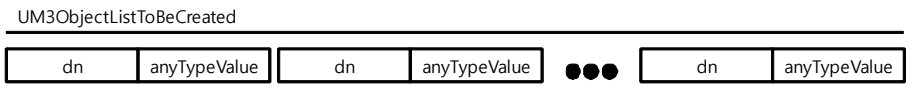


그림 9-UM3ObjectListToBeCreated 타입의 구성 [Structure of UM3ObjectListToBeCreated type]

UM3ObjectListToBeCreated 타입의 ASN.1 정규표현식으로의 표현은 아래와 같습니다.

The UM3ObjectListToBeCreated type can be defined as follows by using ASN.1 regular expressions.

```
UM3ObjectListToBeCreated ::= UM3 SEQUENCE OF UM3 SEQUENCE {
        dn                          UM3ObjectName,
        anyTypeValue                ANY DEFINED BY UM3 ASN.1 module
}
```

UM3ObjectListToBeCreated 타입은 앞서 정의한 바와 같이 CreateObjectGroup 오퍼레이션 만을 위해 사용되며, 그 이외의 용도로는 사용되지 않습니다.

The UM3ObjectListToBeCreated type is only used for CreateObjectGroup operation as defined earlier, and it is not used for any other purpose.

10.40. UM3ActionBroker Class Type

UM3ActionBroker 클래스 타입은 RequestChangeOfAttributeValueReport, RequestChangeOfAttributeGroupValueReport, RequestConditionDetectedReport 오퍼레이션을 수행하기 위한 클래스입니다. 즉, 특정 오브젝트의 액션을 기동시키기 위한 active class 로 정의할 수 있습니다. 본 권고안이 정의하는 다른 클래스 타입들은 정보를 전달하거나 저장하기 위한 passive class 로 분류할 수 있습니다.

The UM3ActionBroker class type is for executing RequestChangeOfAttributeValueReport, RequestChangeOfAttributeGroupValueReport, or RequestConditionDetectedReport operation. In other words, it can be defined as an active class to launch an action of a specific object. Other class types defined in the recommendation can be classified as passive classes for delivering or storing information.

표 43-UM3ActionBroker 클래스의 애트리뷰트 구성 [Attributes of UM3ActionBroker class]

Attribute	Class Type	M/O
targetObjectClassIdentifier	UM3ClassIdentifier	M
targetObjectObjectName	UM3ObjectName	M
targetAttributeClassIdentifier	UM3ClassIdentifier	M
targetAttributeObjectName	UM3ObjectName	M
targetPeriod	UM3DateTime	M
isOnlyOnceAction	UM3Boolean	M
reservedActionTimestamp	UM3DateTime	M
receipientAddress	UM3 SEQUENCE OF UM3CharacterString	M
targetCondition	UM3ObjectValueCondition	M

앞서 정의한 바와 같이 UM3ActionBroker 클래스는 오퍼레이션의 요청을 받아 configuration tree 혹은 information tree 를 구성하고 있는 특정 오브젝트의 Notify 액션을 기동시키는 역할을 수행합니다. 이러한 절차 및 기능을 구현하는 방법은 여러가지 방법이 있을 수 있으며, 해당 구현 방법은 각각의 상황에 적절한 형태로 구현해야 합니다.

As defined earlier, the UM3ActionBroker class receives the request from another operation and launches a Notify action of a specific object in the configuration tree or information tree. There are many ways of implementing this

procedure and function, and the specific implementation should be appropriate for the given environment.

다음은 UM3ActionBroker 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The UM3ActionBroker class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
UM3ActionBroker UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base OBJECT CLASS;
        CHARACTERIZED BY
            Target 애트리뷰트, 조건문 등으로 특징 지워짐;
            Target attribute and conditional statements;

        ATTRIBUTE NAME AS
            targetObjectClassIdentifier             targetObjectClassIdentifier,
            targetObjectObjectName                  targetObjectObjectName,
            targetAttributeClassIdentifier          targetAttributeClassIdentifier,
            targetAttributeObjectName               targetAttributeObjectName,
            targetPeriod                            targetPeriod,
            isOnlyOnceAction                        isOnlyOnceAction,
            reservedActionTimestamp                 reservedActionTimestamp,
            receipientAddress                       receipientAddress,
            targetCondition                         targetCondition;
        ACTION DEFINED AS
            Notify
                UM3Base 클래스의 Notify 액션의 override 액션, 해당 액션은 target 애트리뷰트를 갖고
                있는 오브젝트의 Notify 액션을 호출;
                It is an override action of the Notify action of UM3Base class, and the corresponding action
                calls Notify action of the object with the target attribute;


        BEHAVIOR
            오퍼레이션의 요청에 따른 해당 오브젝트의 액션을 호출하고 그 과정을 관리;
            This calls the action of the corresponding object, based on the request of the operation, and
            manages the process;
;;
```

다음은 상기 UM3ActionBroker 클래스의 ASN.1 정규표현식에 의한 표현입니다.

The UM3ActionBroker class can be defined as follows by using ASN.1 regular expressions.

```
UM3ActionBroker ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &targetObjectClassIdentifier    UM3ClassIdentifier,
        &targetObjectObjectName         UM3ObjectName,
        &targetAttributeClassIdentifier UM3ClassIdentifier,
        &targetAttributeObjectName      UM3ObjectName,
        &targetPeriod                   UM3DateTime,
        &isOnlyOnceAction               UM3Boolean,
        &reservedActionTimestamp        UM3DateTime,
        &receipientAddress              UM3 SEQUENCE OF UM3CharacterString,
        &targetCondition                UM3ObjectValueCondition
}
```

10.40.1. targetObjectClassIdentifier Attribute

이벤트 감시의 대상이 되는 애트리뷰트가 속한 오브젝트의 클래스 아이디를 나타냅니다.

This contains the class ID of the object that includes the attributes currently monitored for an event.

10.40.2. targetObjectObjectName Attribute

이벤트 감시의 대상이 되는 애트리뷰트가 속한 오브젝트의 오브젝트 이름을 나타냅니다.

This contains the name of the object that includes the target attribute for event monitoring.

10.40.3. targetAttributeClassIdentifier Attribute

이벤트 감시의 대상이 되는 애트리뷰트의 클래스 아이디를 나타냅니다.

This contains the class ID of the target attribute for event monitoring.

10.40.4. targetAttributeObjectName Attribute

이벤트 감시의 대상이 되는 애트리뷰트의 오브젝트 이름을 나타냅니다.

This contains the object name of the target attribute for event monitoring.

10.40.5. targetPeriod Attribute

이벤트를 감시하기 위한 측정 주기를 나타냅니다.

This contains the period of measurement for event monitoring.

10.40.6. isOnlyOnceAction Attribute

만약 특정 이벤트를 한 번만 측정하려한다면 해당 애트리뷰트의 값은 TRUE 가 됩니다. 이 때 targetPeriod 애트리뷰트의 값은 무의미한 값으로 무시됩니다.

The value of this attribute should be TRUE if a specific event is to be measured only once. In this case, the value of targetPeriod attribute has no meaning and is ignored.

10.40.7. reservedActionTimestamp Attribute

상기 isOnlyOnceAction 애트리뷰트의 값이 TRUE 일 경우, 한 번의 Notify 액션의 수행 후 바로 액션의 수행을 종료합니다. 이 때 reservedActionTimestamp 에 명기된 일시까지 해당 이벤트가 발생하지 않을 경우 자동으로 액션의 수행을 종료합니다.

When the value of the above isOnlyOnceAction attribute is TRUE, the Notify action is executed once and then terminated. In this case, the execution of the action is automatically terminated if there is no event until the time specified in the reservedActionTimestamp.

10.40.8. receipientAddress Attribute

UM3 SEQUENCE OF 타입의 애트리뷰트이며 Notify 액션의 수행 결과를 매니저로 송신할 때 해당 매니저의 주소를 제외한 다른 수신자의 주소를 기록합니다. 그러나, 해당 애트리뷰트의 활용은 해킹 등 보안관련 사항이 안정적일 경우만 사용해야 하며 그렇지 않을 경우 시스템 보안에 심각한 문제를 일을킬 수 있습니다.

This is a UM3 SEQUENCE OF type. When the result of the Notify action is sent to the manager, it stores the address of other recipients, except the address of the corresponding manager.

10.40.9. targetCondition Attribute

상기 이벤트 감시의 대상이 되는 애트리뷰트에 적용할 조건문을 나타냅니다. 조건에는 임계치의 적용, 단순 변화의 감지 등이 가능합니다.

This contains the conditional statement to be applied to the target attribute for event monitoring. Various conditions like threshold and simple change detection can be applied.

10.40.10. Action

해당 클래스에 UM3Base 로부터 상속받은 Notify 액션 이외에 추가로 정의된 액션은 없습니다.

There is not any additional action defined in this class besides the Notify action inherited from UM3Base.

10.41. UM3EventReport Class Type

UM3EventReport 클래스 타입은 ChangeOfAttributeValueReport, ConditionDetectedReport 오퍼레이션의 파라메터로 활용되는 클래스 타입입니다. 즉, 이벤트 발생에 따라 해당 이벤트의 정보를 기록하여 이벤트 보고를 요청한 매니저에게 송신하기 위한 passive 클래스 타입입니다.

The UM3EventReport class type is used as a parameter for ChangeOfAttributeValueReport and ConditionDetectedReport operations. This is a passive class type for storing event information when an event is generated and sending it to the manager that requested the event report.

표 44-UM3EventReport 클래스의 애트리뷰트 구성 [Attributes of UM3EventReport class]

Attribute	Class 타입	M/O
targetObjectClassIdentifier	UM3ClassIdentifier	M
targetObjectObjectName	UM3ObjectName	M
targetAttributeClassIdentifier	UM3ClassIdentifier	M
targetAttributeObjectName	UM3ObjectName	M
targetAttributeValue	ANY DEFINED BY UM3 ASN.1 module	M
eventTimestamp	UM3DateTime	M

다음은 UM3EventReport 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The UM3EventReport class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
UM3EventReport UM3 OBJECT CLASS
    DERIVED FROM
        UM3Base OBJECT CLASS;
    CHARACTERIZED BY
        Target 애트리뷰트, 결과값 등으로 특징 지워짐;
        Target attributes and results;

    ATTRIBUTE NAME AS
        targetObjectClassIdentifier          targetObjectClassIdentifier,
        targetObjectObjectName               targetObjectObjectName,
```

```
        targetAttributeClassIdentifier          targetAttributeClassIdentifier,
        targetAttributeObjectName               targetAttributeObjectName,
        targetAttributeValue                    targetAttributeValue,
        eventTimestamp                          eventTimestamp;
    ACTION DEFINED AS
        None;
    BEHAVIOR
        발생한 이벤트에 관한 정보를 기록 저장;
        This stores the information of the generated event;
;;
```

다음은 상기 UM3EventReport 클래스의 ASN.1 정규표현식에 의한 표현입니다.

The UM3EventReport class can be defined as follows by using ASN.1 regular expressions.

```
UM3EventReport ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &targetObjectClassIdentifier    UM3ClassIdentifier,
        &targetObjectObjectName         UM3ObjectName,
        &targetAttributeClassIdentifier UM3ClassIdentifier,
        &targetAttributeObjectName      UM3ObjectName,
        &targetAttributeValue           ANY DEFINED BY UM3 ASN.1 module,
        &eventTimestamp                 UM3DateTime
}
```

10.41.1. targetObjectClassIdentifier Attribute

이벤트가 발생한 애트리뷰트가 속한 오브젝트의 클래스 아이디를 나타냅니다.

This contains the class ID of the object that has the attribute associated with the generated event.

10.41.2. targetObjectObjectName Attribute

이벤트가 발생한 애트리뷰트가 속한 오브젝트의 오브젝트 이름을 나타냅니다.

This contains the object name of the object that has the attribute associated with the generated event.

10.41.3. targetAttributeClassIdentifier Attribute

이벤트가 발생한 애트리뷰트의 클래스 아이디를 나타냅니다.

This contains the class ID of the attribute associated with the generated event.

10.41.4. targetAttributeObjectName Attribute

이벤트가 발생한 애트리뷰트의 오브젝트 이름을 나타냅니다.

This contains the object name of the attribute associated with the generated event.

10.41.5. targetAttributeValue Attribute

이벤트가 발생한 애트리뷰트의 변화된 결과값을 갖고 있습니다.

This contains the changed results of attribute associated with the generated event.

10.41.6. eventTimestamp Attribute

이벤트가 발생한 일시를 나타냅니다.

This contains the date and time of the event.

10.41.7. Action

해당 클래스에서 별도로 추가정의된 액션은 없습니다. 또한 UM3Base 클래스의 Notify 액션은 상속받게 되어 있으나 해당 액션이 호출되는 경우는 없는 것으로 정의됩니다. 이는 해당 클래스 오브젝트가 항상 오퍼레이션의 파라메터로만 활용되기 때문입니다.

There is no action defined in this class. The Notify action of the UM3Base class is inherited, but the corresponding action will never be called by definition. This is because the corresponding class object is always used as the parameter of operations.

10.42. UM3ProtocolSupportDescription Class Type

UM3ProtocolSupportDescription 클래스 타입은 UM3 프로토콜을 지원하는 장치 혹은 해당 장치에서 운용되는 매니저 및 에이전트 소프트웨어가 지원하는 UM3 Protocol의 버전번호와 레벨에 관한 정보를 나타내기 위해 사용되는 클래스 타입입니다.

The UM3ProtocolSupportDescription class type is used to store the information of the version number and level of the UM3 protocol that is supported by the manager and agent software running on the corresponding device or

another device that supports UM3 protocol.

표 45-UM3ProtocolSupportDescription 클래스의 애트리뷰트 구성
[Attributes of UM3ProtocolSupportDescription class]

Attribute	Class Type	M/O
version	UM3CharacterString	M
level	UM3CharacterString	M

다음은 UM3ProtocolSupportDescription 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The UM3ProtocolSupportDescription class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
UM3ProtocolSupportDescription UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base OBJECT CLASS;
        CHARACTERIZED BY
            Target 애트리뷰트, 결과값 등으로 특징 지워짐;
            Target attributes and results;

        ATTRIBUTE NAME AS
            version                                 version,
            level                                   level;
        ACTION DEFINED AS
            None;
        BEHAVIOR
            UM3 프로토콜 버전번호와 지원레벨에 관한 정보를 저장;
            This stores the information about the version number and the support level of UM3 protocol;
;;
```

다음은 상기 UM3ProtocolSupportDescription 클래스의 ASN.1 정규표현식에 의한 표현입니다.

The UM3ProtocolSupportDescription class can be defined as follows by using ASN.1 regular expressions.

```
UM3ProtocolSupportDescription ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
```

```
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &version                    UM3CharacterString,
        &level                      UM3CharacterString
}
```

10.42.1. version Attribute

현재 장치 혹은 장치에 설치되어 구동되는 소프트웨어가 지원하는 UM3 프로토콜의 버전번호를 나타냅니다. UM3 프로토콜의 버전번호는 '2011.02' 등과 같은 UM3 Protocol 권고안의 발행번호로 표시합니다.

This contains the version number of the UM3 protocol that is supported by the current device or the software that is installed and running on the device.

10.42.2. level Attribute

현재의 장치 혹은 장치에 설치되어 구동되는 소프트웨어가 지원하는 UM3 프로토콜의 level 을 나타냅니다. 레벨에 관한 정보는 표 2 를 참조하시기 바랍니다.

This contains the level of the UM3 protocol that is supported by the current device or the software that is installed and running on the device. Refer to Table 2 for information about level.

10.43. UM3OperationParameterContainer Class Type

UM3OperationParameterContainer 클래스 타입은 특정 UM3 오퍼레이션의 수행에 따른 결과 값의 리턴 시에 해당 결과가 오브젝트 혹은 애트리뷰트로 리턴될 때 이를 전달하기 위한 용도로 사용됩니다. 즉, 특정 오브젝트에 대해 해당 오브젝트의 um3ObjectName 등과 같은 애트리뷰트의 값을 완전하게 보존한 상태로 전달하기 위한 용도로 활용됩니다. 이 때 UM3OperationParameterContainer 클래스는 본 권고안이 정의하는 모든 타입의 오브젝트를 그 애트리뷰트로 포함할 수 있어야 합니다.

The UM3OperationParameterContainer class type is used when the result of a specific operation is returned as an object or attribute. In other words, it is used for the purpose of passing the values of attributes of the corresponding object such as um3ObjectName while preserving the contents. In this case, the UM3OperationParameterContainer class should be able to include all types of objects defined in this recommendation as its attribute.

표 46-UM3OperationParameterContainer 클래스의 애트리뷰트 구성
[Attributes of UM3OperationParameterContainer class]

Attribute	Class Type	M/O
value	ANY DEFINED BY UM3 ASN.1 module	M

다음은 UM3OperationParameterContainer 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The UM3OperationParameterContainer class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
UM3OperationParameterContainer UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base OBJECT CLASS;
        CHARACTERIZED BY
            Target 애트리뷰트, 결과값 등으로 특징 지워짐;
            Target attributes and results;

        ATTRIBUTE NAME AS
            value                                   value
        ACTION DEFINED AS
            None;
        BEHAVIOR
            특정 UM3 오퍼레이션의 수행에 따른 결과값을 리턴할 경우 사용됨, anyTypeValue 애트리뷰
            트의 값으로 그 특징이 구분됨;
            This is used to return the results of a specific UM3 operation, and it is characterized by the value
            of anyTypeValue attribute;
;;
```

다음은 상기 UM3ProtocolSupportDescription 클래스의 ASN.1 정규표현식에 의한 표현입니다.

The UM3ProtocolSupportDescription class can be defined as follows by using ASN.1 regular expressions.

```
UM3OperationParameterContainer ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
```

```
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &value                      ANY DEFINED BY UM3 ASN.1 module
}
```

10.43.1. value Attribute

특정 UM3 오퍼레이션의 수행에 따른 결과를 리턴하기 위해 사용되며, 해당 애트리뷰트의 값은 본 권고안이 정의한 모든 UM3 타입의 오브젝트로 정의됩니다.

This is used to return the result of a specific UM3 operation, and the value of the corresponding attribute is defined by using any UM3 type objects defined in this recommendation.

10.44. UM3UnstructuredObject Class Type

UM3UnstructuredObject 클래스 타입은 UM3 프로토콜이 갖는 확장성 (scalability) 과 유연성 (flexibility) 을 나타내는 핵심적인 요소입니다. 특히 엄격한 표준의 정의와 이를 통해 상호운용성을 강조해오던 국제표준과는 달리 하나의 데이터 포인트, 혹은 센서 디바이스, 조차도 시간에 따라 다른 형식과 구성의 데이터를 생산하는 경우까지도 처리할 수 있는 방법을 제시합니다.

The UM3UnstructuredObject class type is a core element that represents the scalability and flexibility of the UM3 protocol. Unlike the definition of the strict standard and international standards that emphasize interoperability through strictly defined standards, it suggests methods, even for generating data with various types, and configurations depending on one data point or sensor device. ??

UM3UnstructuredObject 클래스 타입은 UM3 primitive 타입의 애트리뷰트와 본 권고안에 정의된 UM3 complex type, 그리고 또 다른 UM3UnstructuredObject 타입으로 구성된 애트리뷰트를 갖는 오브젝트 클래스로 정의됩니다. 이는 um3ClassIdentifier, um3ObjectName, um3ObjectNameAlias, um3ObjectDescription, um3AttributeList 등의 필수 애트리뷰트를 제외한 나머지 애트리뷰트의 구성에 아무런 제약이 없음을 뜻합니다. 또한 UM3 프로토콜이 정의하는 SER과 XML 기반 인코딩 방법이 제약없이 적용되며, 애트리뷰트의 이름과 클래스 타입, 애트리뷰트의 값 등을 함께 전송할 수 있으므로 오브젝트 클래스의 schema 를 함께 전송할 수 있어 비정형데이터의 송수신을 완벽하게 지원하게 됩니다.

The UM3UnstructuredObject class type is defined as an object class with attributes of UM3 primitive type, UM3 complex type as defined in this recommendation, and attributes of another UM3UnstructuredObject type. This means that there is no restriction in the configuration of all attributes except the essential attributes such as um3ClassIdentifier, um3ObjectName, um3ObjectNameAlias, um3ObjectDescription, and um3AttributeList. There is no restriction in the application of the SER and XML based encoding method defined in UM3 protocol, and the attribute name, class type, and value of attribute can be passed so that the schema of the object class can be passed.

Therefore, it provides full support of transmission and reception of atypical data.

표 47-UM3UnstructuredObject 클래스의 애트리뷰트 구성 [Attributes of UM3UnstructuredObject class]

Attribute	Class Type	M/O
UM3OpenType	Any UM3 complex type that is composed of UM3 Primitive types, other UM3 complex types and other UM3UnstructuredObject types	M

다음은 UM3UnstructuredObject 클래스를 UM3 CLASS DEFINITION SYNTAX 로 표현한 결과입니다.

The UM3UnstructuredObject class can be defined by using UM3 CLASS DEFINITION SYNTAX as follows.

```
UM3UnstructuredObject UM3 OBJECT CLASS
        DERIVED FROM
            UM3Base OBJECT CLASS;
        CHARACTERIZED BY
            Target 애트리뷰트, 결과값 등으로 특징 지워짐;
            Target attributes and results;

        ATTRIBUTE NAME AS
            Unspecified with UM3OpenType            Unspecified with UM3OpenType
        ACTION DEFINED AS
            None;
        BEHAVIOR
            시간에 따라 변화하는 데이터 구조, 혹은 정형화 되어 있지 않은 애트리뷰트의 구성을 나타
            냄;
            This represents the configuration of attributes for time variant data structure or atypical data;
;;
```

다음은 상기 UM3UnstructuredObject 클래스의 ASN.1 정규표현식에 의한 표현입니다.

The UM3UnstructuredObject class can be defined as follows by using ASN.1 regular expressions.

```
UM3UnstructuredObject ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
```

```
        &um3AttributeList               UM3ObjectList,
        &UM3OpenType
}
```

10.44.1. UM3OpenType Type Attribute

UM3UnstructuredObject 클래스는 &UM3OpenType 으로 지정되어 있는 애트리뷰트의 타입, 값, 애트리뷰트의 이름 등을 제한없이 정의할 수 있습니다. 이는 ITU-T ASN.1 이 정의하는 ANY DEFINED BY, open type 과는 다른 형식과 의미를 갖고 있습니다. UM3 Open type 은 UM3 primitive type 으로 정의된 애트리뷰트, UM3 primitive type 애트리뷰트로 구성된 UM3 complex type 오브젝트 클래스, 또 다른 형식과 구성을 갖고 있는 UM3UnstructuredObject 클래스 오브젝트 등으로 구성되며, 인코딩은 UM3 SER, UM3 XML encoding rule 등을 그대로 적용하게 됩니다.

The UM3UnstructuredObject class is used to define the type, value, and name of attribute defined as &UM3OpenType without restriction. It has a different format and meaning from the ANY DEFINED BY and open type defined in ITU-T ASN.1. UM3 Open type is configured as attributes defined with UM3 primitive type, UM3 complex type object class configured with UM3 primitive type attributes, or UM3UnstructuredObject class object with another format and configuration. UM3 SER and UM3 XML encoding rules are applied without modification.

11. UM3 Application Service Information Model

본 권고안이 정의하는 서비스관리 정보모델의 클래스들 만으로도 거의 대부분의 서비스 시스템을 위한 모델링이 가능합니다. 이 때 모델링은 물리적 자원과 논리적 자원에 대한 모델링을 뜻 합니다. 그러나 UM3 프로토콜이 지원하는 서비스의 범위가 넓어지고 이에 따라 다양한 정보모델이 추가적으로 필요할 경우 각 서비스 영역별로 새로운 정보모델을 정의하고 이를 활용할 수 있습니다. 이는 엄격하게 상호운용성을 정의하는 국제표준과는 달리 UM3 프로토콜은 모든 형식의 데이터 모델을 자유롭게 정의하여 UM3 SER 혹은 XML, JSON 등의 인코딩 방식을 이용해 데이터를 주고 받을 수 있음을 뜻합니다.

Almost all service systems can be modeled with the classes of the service management information model defined in this recommendation. In this case, modeling means the modeling of physical resources and logical resources. A new information model, however, can be defined for each service area, and it can be utilized as the scope of the service provided by UM3 protocol is widened and additional new information model is required. Unlike the international standards that demand strict interoperability, all types of data models can be defined without restriction in UM3 protocol, and data can be exchanged using various encoding methods including UM3 SER, XML, or JSON.

UM3 응용서비스 모델은 상기 서비스관리 정보모델이 정의하는 서비스 구성자원, 서비스 관리를 위한 여러가지 논리적인 정보 등을 제외한 특정 서비스에 특화되어 본 권고안이 정의하는 서비스관리 정보모델의 클래스로 모델링이 불가능할 경우에만 사용해야 합니다.

The UM3 application service model is specialized for specific services excluding the service configuration resource defined in the above service management information model and various logical information for service management. It should only be used when the modeling is not possible with the classes of service management information model as defined in this recommendation.

예를 들어 주차관리 서비스의 경우 차량 한 대가 주차할 수 있는 단위 주차구역에 설치되는 센서는 AnalogSensor 혹은 BinarySensor 로 모델링할 수 있습니다. 또한 이러한 주차장 사용 현황 데이터를 수집하기 위해 센서로부터 차량의 주차구역 주차 여부에 관한 데이터를 수집하는 게이트웨이 또한 Gateway 클래스를 이용하여 모델링이 가능합니다. 또한 차량의 주차장 진출입을 통제할 수 있는 차단기, 신호등 또한 BinaryControl 등의 클래스로 모두 모델링이 가능합니다.

Using a parking management service as an example, the sensor installed in the parking area where only one vehicle can be parked can be modeled with AnalogSensor or Binary sensor. The gateway device for collecting data about the status of parking in the parking area from the sensor used for collecting the parking status data can be modeled with Gateway class. The gate and lights for controlling access to the parking lot can be modeled with classes like BinaryControl.

또한 주차요금, 차량번호 등과 같은 데이터의 경우 UM3CharacterString 타입으로 모델링이 가능합니다. 즉, 본 권고안이 정의하는 서비스관리정보모델의 오브젝트들을 이용할 경우 현존하는 모든 형태의 센서 혹은 컨트롤러들에 대한 모델링이 가능합니다.

Data like parking fee and vehicle number can be modeled with UM3CharacterString type. In other words, all types of sensors or controllers in the field can be modeled with the objects of the service management information model defined in this recommendation.

만약 위와 같은 기존 정보모델을 활용하는 것이 불가능하여 새로운 정보모델에 대한 모델링 작업이 필요할 경우에는 새로운 데이터타입 즉, 새로운 UM3ClassIdentifier 를 갖는 클래스를 정의하는 것 보다 본 권고안이 정의하고 있는 UM3UnstructuredObject 클래스를 활용하여야 합니다. 이는 UM3 프로토콜이 갖는 최대의 장점중의 하나인 무한 확장성의 특징을 최대로 활용하기 위함이며 특히 UM3ClassIdentifier 를 활용할 경우 서비스 확장을 위한 새로운 타입을 정의하는 과정에서 UM3 프로토콜을 이용한 상호운용성 확보에 심각한 문제가 발생할 수도 있음을 뜻합니다.

If a new information model is required because the existing information models cannot be used, then the UM3UnstructuredObject class defined in this recommendation should be used instead of defining a new class with UM3ClassIdentifier. The purpose is to maximize the utilization of unlimited scalability, one of the biggest advantages of UM3 protocol. It also means that serious trouble could result in securing interoperability by using UM3 protocol during the definition of new types for service extension using UM3ClassIdentifier.

12. Containment tree 와 UM3 RDN 및 UM3 DN

본 권고안이 정의하는 정보모델의 오브젝트들은 포함관계를 갖고 있습니다. 즉, 하나의 상위 오브젝트가 다른 오브젝트들을 포함하며, 그 상위의 오브젝트는 다른 오브젝트의 하위 오브젝트로 정의될 수 있습니다. 이러한 관계를 포함관계 (containment relationship) 로 정의하고 해당 오브젝트들의 형상을 containment tree 로 정의합니다.

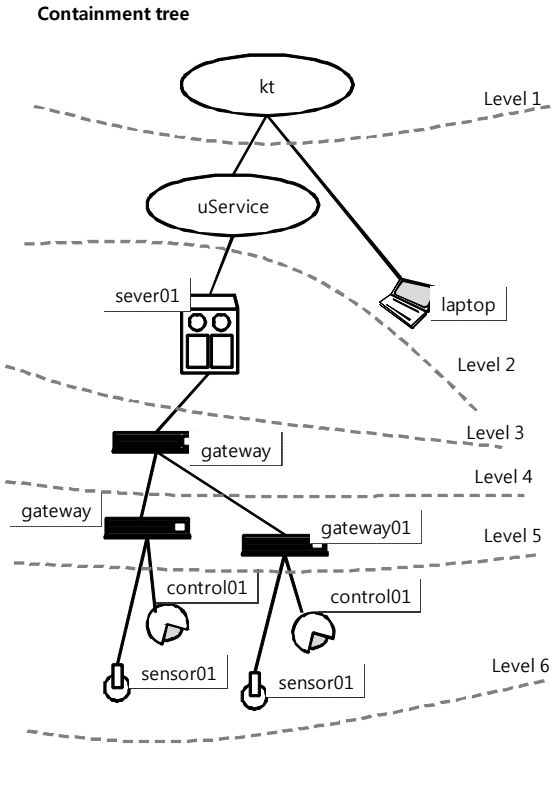


그림 10-Containment tree 와 UM3 RDN 및 UM3 DN [Containment tree, UM3 RDN and UM3 DN]

이상과 같은 containment tree 에서 특정 오브젝트를 찾을 때는 UM3 RDN (Relative distinguished name) 과 UM3 DN (Distinguished name) 을 활용합니다.

UM3 RDN 은 특정 오브젝트의 특정 레벨에 대한 상대적인 구분을 위한 이름을 뜻합니다. 그림 10-Containment tree 와 UM3 RDN 및 UM3 DN [에서 level 5 에 속해 있는 gateway 오브젝트의 level 5 에 대한 UM3 RDN 은 “gateway” 입니다. 그러나 level 3 에 대한 UM3 RDN 은 “server01.gateway.gateway” 입니다. 레벨 6의 좌측 sensor01 과 우측 sensor01 은 level 6 에 대해 동일한 UM3 RDN 을 갖고 있습니다. 그러나, level 5 에 대한 UM3 RDN 은 각각 “gateway.sensor01” 과 “gateway01.sensor01” 로 구분이 가능합니다.

DN 은 최상위 레벨에 대한 해당 오브젝트의 이름을 뜻합니다. 그림 10-Containment tree 와 UM3 RDN 및 UM3 DN [의 level 6 에 속해 있는 좌측의 control01 과 우측의 control01 은 각각 “kt.uService.server01.gateway.gateway.control01” 과 “kt.uService.server01.gateway.gateway01.control01” 이라는 UM3 DN 을 갖고 있습니다. 따라서, 특정 오브젝트의 UM3 DN 은 전체 containment tree 에서 해당 오브젝트를 구분할 수 있는 global name 과 동일한 의미입니다.

본 권고안에서는 UM3 RDN 과 UM3 DN 의 이름을 부여하는 방법을 다음과 같이 정의합니다.

UM3 DN 은 최상위 레벨의 오브젝트로부터 차례로 해당 레벨 별 오브젝트의 UM3 RDN 을 ‘.’ 뒤에 추가하는 방식으로 정의합니다.

UM3 RDN 은 기준이 되는 레벨의 오브젝트로부터 차례로 해당 레벨 별 오브젝트의 UM3 RDN 을 ‘.’ 뒤에 추가하는 방식으로 정의합니다. 일반적으로 사용하는 UM3 RDN 은 해당 오브젝트가 속한 레벨을 기준으로 오브젝트의 이름을 부여하는 방식으로 정의합니다. 즉, 오브젝트의 이름만을 명기하며 ‘.’ 없이 사용하는 것으로 정의합니다.

13. UM3 communication service model

13.1. Definition of UM3 Communication Service Model Operation

본 권고안이 정의하는 UM3 오퍼레이션은 UM3 서비스를 사용하는 사용자의 요청을 받아들여 원격에 존재하는 매니저 혹은 에이전트에게 인코딩된 오퍼레이션 패킷을 전송하고, 해당 매니저 혹은 에이전에이부터 그 응답을 받아 이를 사용자에게 전달하는 역할을 수행합니다.

UM3 operation defined in this recommendation sends the encoded operation packet to the manager or agent in the remote side upon the request of a user using UM3 service. It receives the response from the corresponding manager or agent and delivers the information to the user.

UM3 통신서비스모델은 메니저와 에이전트간의 통신을 위한 오퍼레이션과 해당 오퍼레이션의 파라메터 (parameter) 및 리턴 값 (return value) 을 정의합니다.

The UM3 communication service model defines the operations for communication between manager and agent, as well as parameters and return values of the corresponding operation.

본 권고안은 UM3 통신서비스모델을 정의함에 있어서 오퍼레이션을 클래스에 속한 멤버 오퍼레이션 (member operation) 이 아닌 독립된 개체로 간주하고 정의합니다. 이는 일반적인 C 프로그래밍언어의 함수 (function) 와는 다른 의미이며, 상기 독립된 개체로서의 오퍼레이션의 정의는 일반적인 프로그래밍 언어들 중 C# 혹은 Java 의 인터페이스 (Interface) 혹은 추상클래스 (abstract class) 로 볼 수도 있습니다.

In this recommendation, the operations in the UM3 communication service model are defined as independent objects instead of member operations that belong to a class. It has a different meaning from the function in the C programming language, and the definition of operation as an independent object can be considered as an Interface or abstract class in general programming leagues like C# or Java.

13.2. Services of UM3 Communication Service Model

UM3 프로토콜의 오퍼레이션은 다음과 같은 6가지의 서비스를 지원해야 합니다

The operations of the UM3 protocol should support the following six services.

1) UM3-GET
2) UM3-SET
3) UM3-CREATE
4) UM3-DELETE

5) UM3-CANCEL-GET
6) UM3-EVENT-REPORT

상기 UM3 프로토콜 서비스 분류의 기본 개념은 매니저와 에이전트를 기반으로, 정보모델과 현장의 센서 혹은 제어장치의 상태를 동기화 시키기 위해 필요한 기능들을 그 기능별로 분류한 결과입니다

The basic concept of the classification of the above UM3 protocol services is that they are classified by functions that are required to synchronize the information model and sensor or controller in the field by using a manager or agent.

13.2.1. Definition of UM3-GET Service

특정 오브젝트의 특정 애트리뷰트 값과 해당 오브젝트가 모델링한 실제 현장의 장치의 상태를 일치시키기 위해, 매니저가 에이전트를 통해 현장의 상태정보 값을 가져오기 위해 UM3-GET 서비스의 오퍼레이션들을 이용합니다.

The operations of the UM3-GET service are used to read the status information from the field through a manager or agent for the purpose of synchronizing the values of specific attributes of a specific object and the actual status of the device in the field that is modeled by the corresponding object.

13.2.2. Definition of UM3-SET Service

특정 오브젝트의 특정 애트리뷰트의 값을 변경함으로써 현장 장치의 상태를 변경하는 동기화 과정을 위해 UM3-SET 서비스에 속한 오퍼레이션들을 사용합니다

The operations of the UM3-SET service are used to synchronize the status of devices in the field for the corresponding changes of a specific attribute of a specific object.

13.2.3. Definition of UM3-CREATE service

현장에 새로운 장비가 설치되고 이 장비가 구성자산 및 구성자원으로 등록되기 위해서는 UM3-CREATE 서비스에 속한 오퍼레이션들을 활용합니다

The operations of the UM3-CREATE service are used to register new equipment installed in the field as a configuration asset and configuration resource.

13.2.4. Definition of UM3-DELETE service

특정 장치가 철거되거나 더 이상 사용하지 않는 등의 단계로 들어갔을 때 UM3-DELETE 서비스의 오퍼레이션들을 사용하여 정보모델에서 이를 삭제합니다

The operations of the UM3-DELETE service are used to delete the information model when a specific device is

removed or not used anymore.

13.2.5. Definition of UM3-GET-CANCEL service

UM3-GET-CANCEL 은 이미 요청되어 에이전트에 의해 수행 중인 UM3-GET 서비스의 오퍼레이션에 대한 취소과정으로 정의합니다. 해당 서비스의 오퍼레이션은 대용량 데이터의 송수신을 유발하게 되는 UM3-GET 서비스의 경우 이를 중간에 중단시키기 위한 목적을 갖고 있습니다. 즉, 매우 짧은 시간에 수행되는 UM3-GET 서비스의 오퍼레이션의 경우 UM3-GET-CANCEL 서비스의 오퍼레이션은 그 영향을 미치지 못할 수도 있습니다.

UM3-GET-CANCEL is defined as the process to cancel the operations of the UM3-GET service currently running by agent from the previous request. The purpose is to stop the UM3-GET service in case the operation of the corresponding service triggers the transfer of a large amount of data. The operation of the UM3-GET-CANCEL service may not affect the UM3-GET service if it finishes within a short period of time.

13.2.6. Definition of UM3-EVENT-REPORT service

UM3-EVENT-REPORT 서비스는 정해진 기준에 따라 특정 이벤트의 발생 혹은 감지를 보고하는 오퍼레이션들로 구성되어 있습니다. UM3-EVENT-REPORT 서비스의 오퍼레이션들은 요청 (Request) 및 응답 (Report) 관련 등 2가지 종류의 오퍼레이션들로 구성되어 있습니다

The UM3-EVENT-REPORT service comprises of operations that reports the generation of detection of specific events based on the given conditions. The operations of the UM3-EVENT-REPORT service include two types including the ones related to Request and others related to Report.

UM3-EVENT-REPORT 서비스는 위에 정의한 것과 같이 특정 이벤트의 발생을 감지하고 보고토록 요청하는 오퍼레이션들에 의해 에이전트 내부의 특정 기능이 작동하게 됩니다. 그러나, UM3 프로토콜의 UM3-EVENT-REPORT 서비스를 구현하는 방법에 있어서, UM3-SET 서비스의 오퍼레이션들을 이용하여 정보모델에 정의한 클래스들을 파라메터로 넘기거나 혹은 특정 클래스의 애트리뷰트 값을 지정하여 특정 조건을 검출해낼 수도 있습니다.

The UM3-EVENT-REPORT service enables specific functions inside the agent by the operations that request detection and reporting of specific event as defined above. The UM3-EVENT-REPORT service of the UM3 protocol, however, can be implemented in such a way that the class defined in the information model can be passed as a parameter by using the operations of the UM3-SET service or specific conditions can be detected by assigning an attribute value of a specific class.

지금까지 정의한 방법을 event driven 방식의 이벤트 검출 방법으로 정의합니다

The method defined this way is defined as an event driven type detection.

그러나 본 권고안이 정의하는 서비스 운용 환경에 있어서, 모든 컴퓨터가 일정 규모 이상의 사양으로 만들어져 있지 않을 수도 있으므로, 경우에 따라서는 UM3-SET 서비스와 UM3-EVENT-REPORT 서비스의 일부 오퍼레이션들의 선택적 구현을 통해 본 권고안이 정의하는 통신서비스모델을 현장에 적용할 수도 있습니다

Since not all computers meet certain specification in terms of the operation environment for the services defined in this recommendation, the communication service model defined in this recommendation can be applied to the field by selectively implementing some operations of the UM3-SET services and UM3-EVENT-REPORT services if required.

또한 본 권고안은 UM3-EVENT-REPORT 방식의 오퍼레이션들에 대한 구현 혹은 기능의 장착 여부를 필수요소 (Mandatory) 가 아닌 선택요소 (Optional) 로 규정합니다. 이는 경우에 따라 매우 단순한 저사양의 부품으로 만들어진 컴퓨터가 센서 혹은 제어장치 등을 연결하여 원격제어 혹은 원격감시 서비스를 제공할 때, 해당 저사양 컴퓨터로는 능동적인 event-driven 방식의 조건 감지 및 보고기능을 구현하기 쉽지 않은 경우도 있기 때문입니다

Implementation or installation of the functions for the UM3-EVENT-REPORT type operations in this recommendation are specified as Optional, not Mandatory requirements. This is because it may not be easy to implement active event-driven type condition detection and reporting function with underpowered computers when sensor or remote control devices are connected to underpowered computers and remote control or remote monitoring service is provided.

상기와 같은 경우 능동적인 이벤트 검출 기능을 갖고 있지 않은 에이전트를 대상으로 매니저가 상태 정보 값을 가져온 후, 해당 값에 대한 임계치 (threshold) 등의 조건을 적용하여 이벤트를 검출하게 됩니다

For agents that do not have an active event detection function as in the above case, the manager reads the status value and detects events by applying conditions like threshold to the corresponding value.

이러한 방식을 polling 방식으로 정의합니다

This type is defined as polling method.

본 권고안은 상기 UM3-EVENT-REPORT 서비스에 정의된 오퍼레이션들을 모두 UM3-SET 과 UM3-GET 서비스에 정의된 오퍼레이션들만을 이용해서 구현할 수도 있는 것으로 정의합니다

All operations of the UM3-EVENT-REPORT service defined in this recommendation can be implemented by using only the operations defined in the UM3-SET and UM3-GET services.

13.3. Definition of Confirmed and Unconfirmed

본 권고안이 정의하는 오퍼레이션들은 confirmed 와 unconfirmed 오퍼레이션으로 구분됩니다. 본 권고안은 다음과 같이 confirmed 와 unconfirmed 를 관련 용어와 함께 정의합니다.

- Request 는 특정 오퍼레이션에 대한 요청
- Reply 는 해당 오퍼레이션에 대한 응답
- Confirmed 는 상기 request 에 대해 reply 과정을 거쳐 세션을 종료하는 형식
- Unconfirmed 는 상기 request 에 대해 reply 과정 없이 세션을 종료하는 형식

따라서, unconfirmed 의 경우 오퍼레이션을 요청한 서비스 요청자가 응답을 기다려도 서비스 제공자는 응답을 제공할 의무가 없습니다. 마찬가지로 confirmed 의 경우 서비스 요청자는 서비스 제공자의 응답을 기다리고, 자신이 요청한 오퍼레이션의 결과를 확인하고 다음 단계의 절차로 넘어갈 수 있습니다.

단, unconfirmed 의 경우 reply 과정은 생략할 수 있으나 반드시 UM3Ack 형식의 APDU 를 송신하여 해당 오퍼레이션을 정상적으로 수신하였음을 알려야 합니다.

Operations defined in this recommendation can be classified as confirmed and unconfirmed operations. Terms related to the confirmed and unconfirmed operations are defined as follows in this recommendation.

- Request is the request to specific operation.
- Reply is the response to the corresponding operation.
- Confirmed means that the session is terminated after getting a reply from the above request.
- Unconfirmed means that session is terminated without getting a reply from the above request.

Therefore, the service provider does not have to reply, even when the service requestor requesting the operation is waiting for the response in case of unconfirmed operation. On the other hand, the service requester waits for the response from the service provider in case of confirmed operation, and then it checks the results of the requested operation and moves on to the next stage.

For an unconfirmed case, the reply process can be skipped, but the APDU of UM3Ack type should be transmitted to notify that the corresponding operation is received correctly.

13.4. Definition of UM3 Session

본 권고안이 정의하는 세션 (session) 은 상기 오퍼레이션이 에이전트 혹은 매니저로 UM3 APDU 를 전송한 후 그 응답을 수신하는 방법을 기준으로 아래와 같은 2 가지 형태로 분류합니다.

- Connection oriented UM3 session
- Connectionless UM3 session

Connection oriented UM3 session 은 데이터를 송신한 후 응답 APDU 를 수신한 후 연결을 끊는 경우로 정의합니다.

Sessions defined in this recommendation can be classified into two types as follows, depending on the method to receive the response after sending DM3 APDU to the agent or manager from the operations defined above.

- Connection oriented UM3 session
- Connectionless UM3 session

Connection oriented UM3 session is defined as the case in which the connection is terminated upon reception of response APDU after sending data.

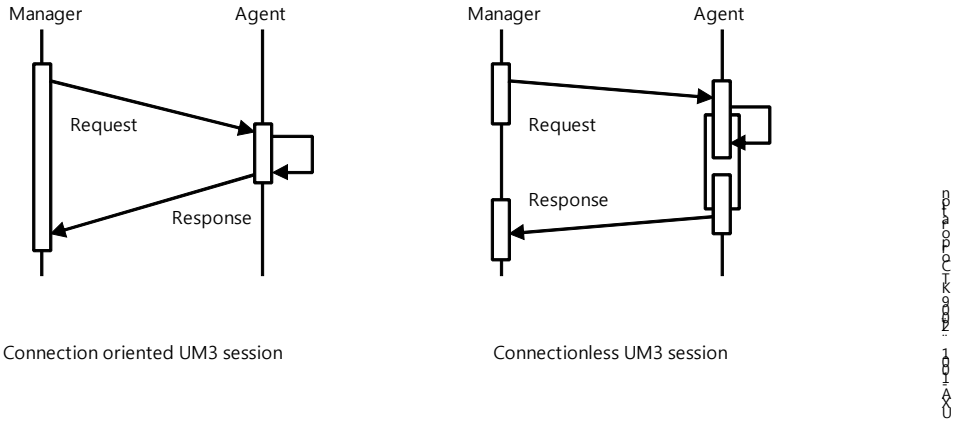


그림 11-Connection oriented UM3 session 과 connectionless UM3 session [Connection oriented UM3 session and connectionless UM3 session]

본 권고안이 정의하는 UM3 프로토콜은 상기 응답을 하나의 오퍼레이션으로 보고 별도의 오퍼레이션 OperationResponse 로 정의합니다. OperationResponse 오퍼레이션은 다른 오퍼레이션들과 마찬가지로 UM3 SER 인코딩 시 UM3ClassIdentifier 와 UM3ObjectName 애트리뷰트의 값을 기록한 후 전송합니다.

The UM3 protocol defined in this recommendation considers the above response as an operation, and it defines a separate OperationResponse operation. The OperationResponse operation records the values of UM3ClassIdentifier and UM3ObjectName during UM3 SER encoding and transmits the data like other operations.

Connectionless UM3 session 은 데이터를 송신한 후 응답을 기다리지 않고 바로 연결을 끊는 경우로 정의합니다. 즉, 응답은 응답을 송신하는 측이 별도의 세션을 설정하고, 해당 세션을 이용하여 데이터를 송신함으로써 이루어지는 경우로 정의합니다. 이러한 모든 데이터 송수신 과정은 TCP/IP 프로토콜의 응용계층 (application layer) 을 기준으로 정의합니다.

The connectionless UM3 session is defined as the case in which connection is terminated without waiting for the response after transmitting data. In other words, a separate session is established by the receiver that receives the response, and the data transmission is performed by using the corresponding session. All these data transmissions and reception processes are defined as the application layer of TCP/IP protocol as reference.

13.5. Mandatory and Optional Operation of UM3 Service

본 권고안은 오브젝트 그룹 (group) 혹은 애트리뷰트의 그룹을 대상으로 하는 오퍼레이션의 지원여부도 선택적 요소 (optional) 로 정의합니다. 이는 2 개 이상 복수개의 오브젝트 혹은 애트리뷰트에 대한 오퍼레이션은 1 개의 오브젝트 혹은 애트리뷰트에 대한 오퍼레이션의 반복적인 실행으로 동일한 결과를 얻을 수 있기 때문입니다. 다만, 대부분의 경우 매니저가 설치된 고사양의 컴퓨터 사이의 통신을 위한 기능은 상기 선택요소로 정의된 오퍼레이션 들을 모두 구현하는 것을 권장합니다.

This recommendation specifies the support of operations for object groups or attribute groups as optional requirements. This is because the same result as the operations for multiple objects or attributes can be achieved by repeating the operation for one object or one attribute. It is recommended to implement all operations specified as optional functions required for communication with high performance computers with the installed manager in most cases.

표 48-UM3 서비스의 분류와 UM3 프로토콜의 오퍼레이션에서 보는 바와 같이 UM3 프로토콜은 6개의 서비스와 모두 22개의 오퍼레이션으로 정의되어 있습니다. 이들 중 선택요소로 정의된 오퍼레이션들을 제외한 8개의 오퍼레이션들 만으로도 비즈니스 혹은 상위 서비스가 필요로 하는 모든 데이터 송수신 수요를 만족시킬 수 있습니다.

As shown in Table 47, six services and 22 operations are defined in UM3 protocol. All data transfer requirements for business or high level services can be satisfied with these operations without the ones specified as optional.

하지만 매니저가 설치된 컴퓨터에 과도한 부하가 걸리거나, 혹은 매니저용 컴퓨터 자체의 하드웨어 성능이 만족스럽지 못할 경우에는 에이전트용 컴퓨터의 일부 사양을 상향 조정하여 선택요소로 명기된 오퍼레이션들을 구현하고 이들을 적극 활용할 것을 권고합니다.

It is highly recommended to implement and use optional operations by upgrading the computer for an agent if the computer with the manager is overloaded or hardware of the manager computer is underperforming.

13.6. UM3 Operation and RDN

이 장에서 정의하는 UM3 오퍼레이션들은 앞서 정의한 바와 같이 매니저와 에이전트 간의 정보의 동기화를 위한 오퍼레이션들입니다. 즉, 매니저와 에이전트가 동일한 형식과 종류의 정보모델을 저장 관리하고, 이들간의 정보를 동기화 시켜 관리함으로써 원격의 환경 혹은 장치의 상태 등의 정보를 사용자가 원할 때 즉시 제공하는 것을 목적으로 하고 있습니다.

The UM3 operations defined in this chapter are for the synchronization of information between manager and agent as defined earlier. The purpose is to immediately provide the information about a remote environment or states of devices when the information is requested by the user. The same format and type of information model is stored by manager and agent, and the information is synchronized between them.

정보의 동기화 과정에는 매니저와 에이전트가 관리하는 정보모델의 오브젝트에 대한 검색과정을 수반합니다. 정보모델 내부의 오브젝트에 대한 검색은 클래스 타입과 해당 오브젝트의 이름을 기준으로 이루어집니다. 이 때 오브젝트의 이름은 앞서 정의한 UM3 RDN 혹은 UM3 DN 을 활용해야 합니다.

The search process for the objects of information model managed by manager and agent is accompanied during the process of information synchronization. Search operations for objects inside the information model are performed using the class type and name of the corresponding object as reference. In this case, the UM3 RDN or UM3 DN defined earlier should be used for the name of objects.

그림 12-UM3 오퍼레이션과 RDN 의 관계 [은 통신 혹은 데이터 송수신 기능을 갖고 있지 안은 센서와 원격제어장치가 게이트웨이에 연결되어 있으며, 해당 게이트웨이들은 상위 게이트웨이에 연결되어 있고, 해당 상위 게이트웨이는 다시 서버와 연결되어 있는 전형적인 원격관리 서비스 시스템의 구조를 보여주고 있습니다. 해당 시스템을 구성하고 있는 오브젝트들의 containment tree 는 그림 13-Containment tree 의 예 [과 같습니다

In Fig. 12, the sensor or remote control device that does not have a communication or data transfer function is connected to the gateway. The corresponding gateway is connected to the upper level gateway, and it is again connected to the server, which is a typical remote management service system. The containment tree of objects configuring the corresponding system is shown in Fig. 13.

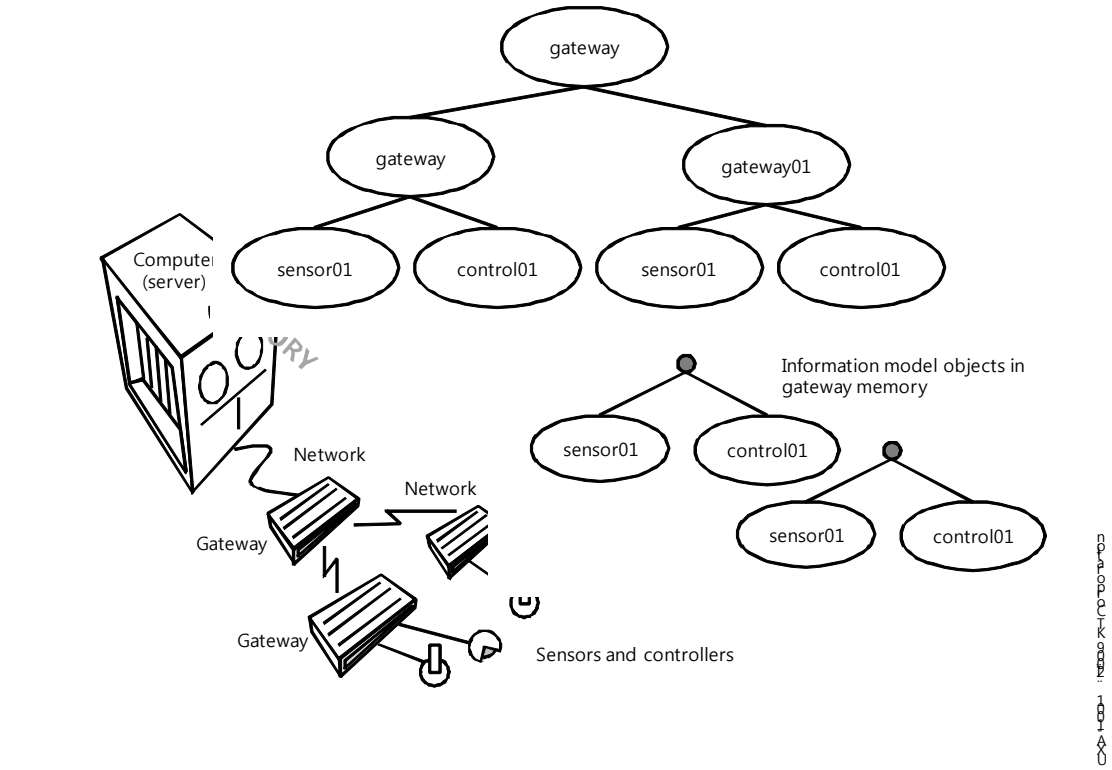


그림 12-UM3 오퍼레이션과 RDN 의 관계 [Relationship between UM3 operation and RDN]

서버 즉, 컴퓨터가 관리하는 정보모델의 오브젝트들에 대한 각 오브젝트 레벨 별 UM3 RDN 은 ‘gateway’, ‘gateway’, ‘gateway01’, ‘sensor01’, ‘control01’, ‘sensor01’, ‘control01’ 로 정의됩니다. 만약 서버를 레벨 1으로 정의하고 상위 게이트웨이를 레벨 2로 정의한다면, 레벨 2에 대한 UM3 RDN 은 ‘gateway’, ‘gateway.gateway’, ‘gateway.gateway01’, ‘gateway.gateway.sensor01’, ‘gateway.gateway.control01’, gateway.gateway01.sensor01’, ‘gateway.gateway01.control01’ 로 정의됩니다

The UM3 RDN for each object level for the objects of information model managed by server computer is defined as ‘gateway’, ‘gateway’, ‘gateway01’, ‘sensor01’, ‘control01’, ‘sensor01’, and ‘control01’. If server is defined as level 1 and upper level gateway is defined as level 2, UM3 RDN for level 2 is defined as ‘gateway’, ‘gateway.gateway’, ‘gateway.gateway01’, ‘gateway.gateway.sensor01’, ‘gateway.gateway.control01’, gateway.gateway01.sensor01’, and ‘gateway.gateway01.control01’.

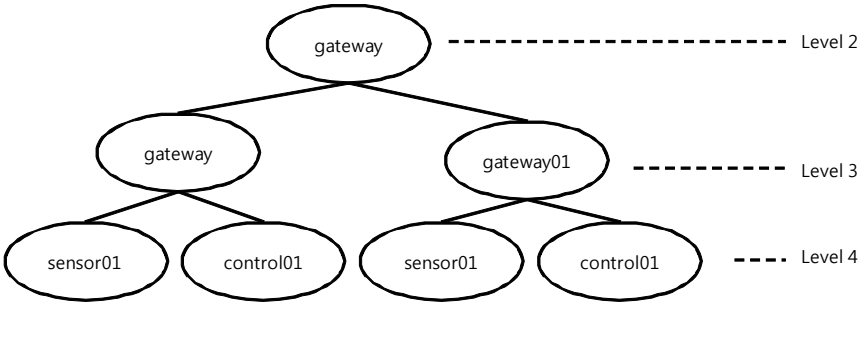


그림 13-Containment tree 의 예 [Example of Containment tree]

앞서 정의한 바와 같이 UM3 서비스관리 정보모델과 응용서비스 정보모델의 오브젝트들은 um3ClassIdentifier 와 um3ObjectName 애트리뷰트들을 공통적으로 갖고 있습니다. 특히 UM3ObjectName 타입으로 정의된 um3ObjectName 애트리뷰트는 각 오브젝트가 속한 레벨을 기준으로 정의된 UM3 RDN 으로 정의된 오브젝트의 이름을 저장하고 있습니다. 예를 들어 그림 12-UM3 오퍼레이션과 RDN 의 관계 [과 그림 13-Containment tree 의 예 [의 최하위 좌측의 센서는 아래와 같은 오브젝트 이름 즉, 해당 센서가 속한 레벨 4 에 대한 UM3 RDN 을 아래와 같은 이름으로 갖고 있습니다.

As defined earlier, objects of the UM3 service management information model and application service information model have um3ClassIdentifier and um3ObjectName as common attributes. The um3ObjectName attribute defined with UM3ObjectName type contains the name of the object defined with UM3 RDN, which is defined based on the level to which the object belongs. For example, the sensor on the left at the lowest level in Fig. 12 and Fig. 13 has the following object name: it has UM3 RDN for level 4, and the corresponding sensor to which it belongs has the following name.

um3ObjectName UM3ObjectName ::= “sensor01”

상기 오브젝트에 대한 UM3 DN 은 서버의 UM3 RDN 을 ‘server’ 라고 정의한다면, ‘server.gateway.gateway.sensor01’ 로 정의할 수 있습니다. 만약 상위 게이트웨이가 관리하고 있는 정보모

델에서 상기 오브젝트를 검색하고자 한다면, ‘server.gateway.gateway.sensor01’ 로 표현되는 UM3 DN 을 이용하거나 혹은, 상위 게이트웨이가 속해 있는 레벨인 레벨 2에 대한 UM3 RDN 인 ‘gateway.gateway.sensor01’ 를 이용하여 해당 오브젝트를 찾아야 합니다.

If the UM3 RDN of the server is defined as ‘server’, then UM3 DN of the above object can be defined as ‘server.gateway.gateway.sensor01’. To search for the above object from the information model managed by the upper level gateway, UM3 DN expressed as ‘server.gateway.gateway.sensor01’ or ‘gateway.gateway.sensor01’, UM3 RDN for the level 2 to which the upper level gateway belongs should be used to search the corresponding object.

UM3 오퍼레이션은 대부분의 경우 특정 오브젝트 혹은 오브젝트 그룹 (UM3 object group) 에 대한 오퍼레이션들로 정의됩니다. 따라서, 특정 단일 오브젝트나 혹은 오브젝트 그룹에 속한 여러 오브젝트들을 찾기 위해서는 원칙적으로 UM3 DN 정보를 이용해야 합니다. 그러나, 그림 13-Containment tree 의 예 [에서 보는 바와 같이 매니저가 속한 물리적 자원인 ‘server’ 오브젝트를 기준으로, 에이전트가 속한 상위 게이트웨이 ‘server.gateway’ 와 그 하위 게이트웨이 인 ‘server.gateway.gateway’ 하위에 해당 오브젝트인 ‘server.gateway.gateway.sensor01’ 이 존재하고 있습니다. 따라서, 서버는 상위 게이트웨이에 ‘gateway.gateway.sensor01’ 이라는 UM3 RND 만을 넘겨주어도 해당 상위 게이트웨이는 해당 오브젝트가 자신이 관리하는 하위 게이트웨이 ‘gateway’ 하위에 존재하는 ‘sensor01’ 오브젝트라는 것을 알 수 있습니다.

Most of the UM3 operations can be defined as operations for a specific object or object group (UM3 object group). Therefore, UM3 DN information should be used as a general rule to search single object or multiple objects that belong to an object group. As shown in the Fig. 13, from the ‘server’ object, physical resource to which the manager belongs as reference, there is a ‘server.gateway.gateway.sensor01’ object under the upper level gateway ‘server.gateway’ to which the agent belongs and its lower level gateway, ‘server.gateway.gateway’. Therefore, when service passes UM3 RND of ‘gateway.gateway.sensor01’ to the upper level gateway, the corresponding upper level gateway can recognize that the corresponding object is the ‘sensor01’ object that is under the lower level gateway ‘gateway’ that is managed by the corresponding object.

본 권고안이 정의하는 UM3 오퍼레이션들을 이용하여 특정 오브젝트를 찾아 특정 작업을 수행하고자 할 때, 오퍼레이션을 호출하는 매니저는 에이전트가 속한 레벨을 기준으로 해당 레벨에 대한 검색 대상 오브젝트의 UM3 RDN 을 하위의 에이전트에게 넘겨주어야 합니다.

When a specific object is to be searched and to perform specific tasks using the UM3 operations defined in this recommendation, the manager calling the operation should pass UM3 RDN of the search target object for the corresponding level based on the level to which the agent belongs to the lower level agent.

예를 들어 그림 12-UM3 오퍼레이션과 RDN 의 관계 [에서 UM3 RDN ‘server’ 를 갖는 컴퓨터에 매니저가 설치되어 있고, UM3 RDN ‘gateway’ 를 갖고 있는 상위 게이트웨이에 에이전트가 설치되어 있다고 가정합니다. 이러한 경우, 에이전트는 하위 게이트웨이 ‘gateway’ 와 ‘gaeway01’ 및 그 하위의 장치들을

포함하는 정보모델을 관리하게 됩니다.

For example, let's assume that the manager is installed in the computer with UM3 RDN 'server' in Fig. 12, and the agent is installed in the upper level gateway that has UM3 RDN 'gateway'. In this case, the agent manages the information model including the lower level gateway devices, 'gateway' and 'gateway01', in the lower level.

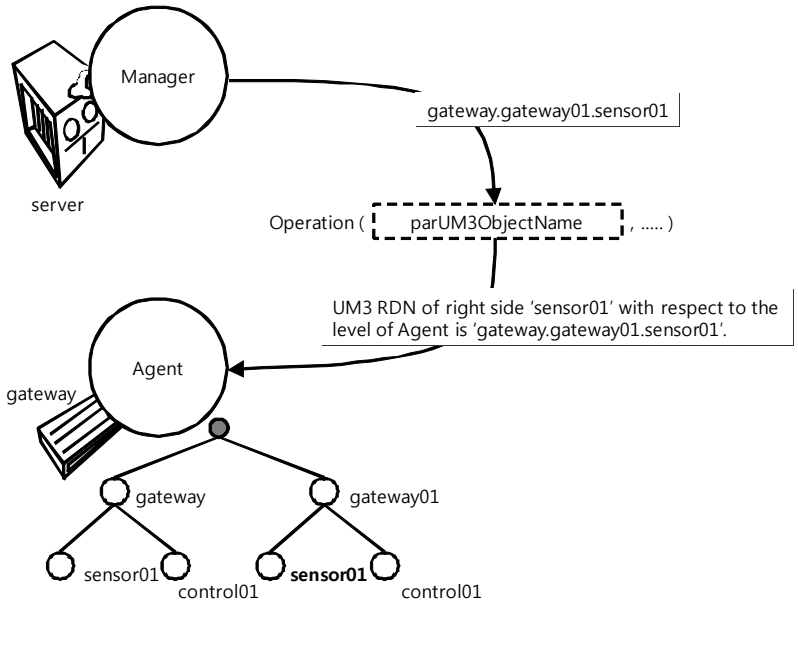


그림 14-UM3 오퍼레이션과 UM3 RDN 의 구성 [Configuration of UM3 operation and UM3 RDN]

만약 매니저가 UM3 DN 'server.gateway.gateway01.sensor01' 오브젝트에 대한 특정 오퍼레이션을 호출 할 경우, 매니저는 해당 UM3 오퍼레이션에 에이전트가 속한 레벨을 기준으로 하는 UM3 RDN 'gateway.gateway01.sensor01' 로 표현되는 오브젝트의 UM3 RDN 을 그림 14-UM3 오퍼레이션과 UM3 RDN 의 구성 [과 같이 넘겨주어야 합니다.

If the manager calls a specific operation for UM3 DN 'server.gateway.gateway01.sensor01', then the manager should pass UM3 RDN of the object that is expressed as UM3 RDN 'gateway.gateway01.sensor01' with the level to which the agent belongs as reference to the corresponding UM3 operation as shown in Fig. 14.

상기 예를 구현 측면의 관점에서 살펴본다면, UM3 오퍼레이션은 매니저에 의해 호출되고, 그 파라메터들 중 하나인 parUM3ObjectName 에 ‘gateway.gateway01.sensor01’ 이라는 값의 UM3 RDN 을 넘겨받게 됩니다. UM3 오퍼레이션은 ‘gateway.gateway01.sensor01’ 이라는 UM3 RDN 을 구성하는 엘리먼트들 중 맨 앞의 ‘gateway’ 를 에이전트가 설치된 오브젝트의 UM3 RDN 으로 인식합니다. 따라서, UM3 오퍼레이션은 해당 에이전트가 설치된 상위 게이트웨이의 IP Address 와 Port no. 를 테이블 혹은 데이터 저장소에서 찾아 해당 IP Address 의 Port no. 로 ‘gateway.gateway01.sensor01’ 이라는 UM3 RDN 을 송신합니다.

If we look at the above example from the perspective of implementation, UM3 operation is called by manager, and it receives UM3 RDN with the value of ‘gateway.gateway01.sensor01’ in parUM3ObjectName, one of the parameters. UM3 operation recognizes the ‘gateway’ at the beginning of the elements configuring UM3 RDN called ‘gateway.gateway01.sensor01’ as UM3 RDN of the object in which the agent is installed. Therefore, UM3 operation finds the IP address and port number of the upper level gateway in which the corresponding agent is installed, and sends UM3 RDN called ‘gateway.gateway01.sensor01’ by using the IP address and port number.

상기 UM3 APDU 를 매니저로부터 수신한 에이전트는 parUM3ObjectName 파라메터에 기록되어 전송된 ‘gateway.gateway01.sensor01’ UM3 RDN 중 첫 번째 엘리먼트인 ‘gateway’ 가 자신의 UM3 RDN 과 동일한 것을 확인합니다. 그리고, 첫 번째 엘리먼트를 제외한 ‘gateway01.sensor01’ UM3 RDN 을 이용하여 자신이 관리하는 정보모델에서 해당 오브젝트를 찾은 후, UM3 오퍼레이션이 지시하는 작업을 수행하게 됩니다.

When the agent receives UM3 APDU from the manager as described above, it checks that the ‘gateway’, the first element of the ‘gateway.gateway01.sensor01’ UM3 RDN that is stored and sent as the parUM3ObjectName parameter, is the same as its own UM3 RDN. It then runs the task instructed by UM3 operation after finding the corresponding object from the information model managed by itself by using ‘gateway01.sensor01’ UM3 RDN, excluding the first element.

13.7. Classification and Definition of UM3 Operation

본 권고안이 정의하는 UM3 프로토콜의 오퍼레이션의 구성은 표 48-UM3 서비스의 분류와 UM3 프로토콜의 오퍼레이션과 같습니다. 표 48-UM3 서비스의 분류와 UM3 프로토콜의 오퍼레이션에서 ‘C/U’로 표기된 오퍼레이션들은 그 응답방식을 confirmed 혹은 unconfirmed 두 가지 방식으로 모두 구현할 수 있음을 뜻합니다. ‘O’로 표기된 항목은 선택적으로 구현할 수 있는 항목들이며, 본 권고안은 앞서 기술한 바와 같이 가급적 모든 오퍼레이션에 대한 구현을 권고하고 있습니다.

The operations of UM3 protocol defined in this recommendation are shown in Table 47. ‘C/U’ in Table 47 means that the corresponding operation can be implemented in both confirmed or unconfirmed methods. Implementation of the items with ‘O’ is an optional requirement, and it is recommended to implement all operations if possible as

mentioned earlier.

오퍼레이션은 13.1에서 기술하고 정의한 것과 같이 특정 클래스에 속한 멤버 오퍼레이션 혹은 메써드(method)로 정의하지 않고 하나의 독립된 개체로 정의합니다. 이는 UM3 오퍼레이션들에 대해 UM3 SER 인코딩을 위한 클래스아이디를 부여하여, 이들 오퍼레이션들의 인코딩 결과와 UM3 기본 타입 혹은 정보모델의 오브젝트들에 대한 인코딩 결과와 구분하기 위한 목적을 갖고 있습니다

Operation is defined as an independent object instead of a member operation or method that belongs to a specific class as described and defined in 13.1. The purpose is to assign class ID for UM3 SER encoding for UM3 operations and to identify the encoding results of these operations and the encoding results of UM3 primitive type or object of the information model.

표 48-UM3 서비스의 분류와 UM3 프로토콜의 오퍼레이션
[Classification of UM3 service and operations of UM3 protocol]

Service	**Operation**	**M/O**[2]	**C/U**[3]	**Class ID**
UM3-GET	GetObjectValue	M	C	1
	GetObjectAttributeValue	M	C	2
	GetObjectGroupValue	O	C	3
	GetObjectAttributeGroupValue	O	C	4
UM3-SET	SetObjectValue	M	C	5
	SetObjectAttributeValue	M	C	6
	SetObjectGroupValue	O	C	7
	SetObjectAttributeGroupValue	O	C	8
UM3-CREATE	CreateObject	M	C	9
	CreateObjectGroup	O	C	10
UM3-DELETE	DeleteObject	M	C	11
	DeleteObjectGroup	O	C	12
UM3-CANCEL-GET	CancelGetObjectValue	M	C	13
	CancelGetObjectGroupValue	O	C	14
	CancelGetObjectAttributeValue	M	C	15
	CancelGetObjectAttributeGroupValue	O	C	16
UM3-EVENT-REPORT	ChangeOfAttributeValueReport	O	C	17
	ConditionDetectedReport	O	C	18
	RequestChangeOfAttributeValueReport	O	C	19
	RequestConditionDetectedReport	O	C	20
	CancelEventReport	O	C	21
	OperationResponse	M	U	0

[2] Mandatory / Optional
[3] Confirmed / Unconfirmed

상기 오퍼레이션에 대한 정의를 다른 관점에서 해석한다면 그림 15-UM3 오퍼레이션에 대한 클래스로의 모델링 개념 [와 같습니다. 즉, GetObjectValue 오퍼레이션을 하나의 독립된 개체인 클래스로 간주한다면, 해당 클래스는 오퍼레이션과 동일한 클래스 명칭을 갖고 있으며, UM3ClassIdentifier 와 UM3ObjectName 타입의 애트리뷰트를 갖고 있는 것으로 볼 수 있습니다. 이 때 UM3ClassIdentifier 타입의 애트리뷰트의 값은 1011_{10}이 상수값으로 저장되며, UM3ObjectName 타입의 애트리뷰트에는 앞서 기술한 UM3 세션 번호가 기록되게 됩니다.

Fig. 15 is another view of the definition of these operations. If the GetObjectValue operation is considered as a class of indecent object, then it can be considered that the corresponding class has the same class name as the operation and it has the attributes of UM3ClassIdentifier and UM3ObjectName type. The value of the UM3ClassIdentifier type attribute is stored as a constant of 1011_{10}, and the UM3 session number that is described earlier is stored in the UM3ObjectName type attribute.

UM3 세션번호는 매니저가 전송하는 패킷에 부여되는 고유번호이며, 특정 UM3 APDU 를 다른 UM3 APDU 로부터 구분하기 위한 식별자입니다.

The UM3 session number of a unique number is assigned to the packet transmitted by the manager, and it is used for identifying a specific UM3 APDU from other UM3 APDU.

본 권고안이 정의하는 UM3 SER 인코딩 룰은 세션번호를 오퍼레이션 오브젝트의 오브젝트 이름과 동일시 하여 인코딩하는 것으로 정의합니다. 이러한 UM3 오퍼레이션에 대한 단일 멤버 오퍼레이션을 갖는 클래스로의 모델링 개념은 0장에서 다루게 될 UM3 SER 인코딩 및 디코딩에 따른 UM3 APDU의 이해를 쉽게 접근할 수 있게 해줍니다.

The UM3 SER encoding rule in this recommendation is to use the session number as the object name of the operation object. This modeling concept with single member operation of UM3 operations enables easier understanding of UM3 APDU associated with UM3 SER encoding and decoding that will be covered in Chapter 15.

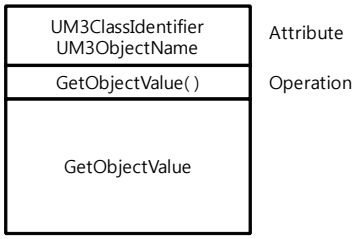


그림 15-UM3 오퍼레이션에 대한 클래스로의 모델링 개념 [Concept of modeling of UM3 operation as class]

이상과 같은 UM3 오퍼레이션의 ASN.1 정규표현식으로의 표현은 다음과 같습니다.

This UM3 operation can be defined by using ASN.1 regular expressions as follows.

```
UM3-OPERATION ::= CLASS {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &ParameterType
}
```

상기 ASN.1 정규표현식이 나타내는 뜻은 다음과 같습니다.

The above ASN.1 regular expression can be interpreted as follows.

UM3 오퍼레이션은 UM3ClassIdentifier 타입의 um3OperationClassIdentifier 변수를 갖습니다. 해당 변수는 UM3 SER 인코딩 룰에 따라 UM3 APDU의 T 필드로 변환됩니다.

The UM3 operation has the um3OperationClassIdentifier variable of UM3ClassIdentifier type. The corresponding variable is converted to T field of UM3 APDU according to the UM3 SER encoding rule.

UM3 오퍼레이션은 UM3ObjectName 타입의 um3OperationObjectName 변수를 갖습니다. 해당 변수는 동일한 타입의 오퍼레이션들로부터 해당 오퍼레이션을 구분하기 위해 유일한 값이 주어져야 하며, 이를 UNIQUE 라는 키워드로 명기하고 있습니다. 해당 변수는 UM3 APDU 의 N 필드에 기록됩니다.

The UM3 operation has the um3OperationObjectName variable of UM3ObjectName type. A unique value should be given to identify the corresponding operation from other operations of the same type, and it is specified with the UNIQUE keyword. The corresponding variable is stored in the N field of UM3 APDU.

위의 UM3 오퍼레이션의 N 필드에 기록되는 오브젝트의 이름 um3OperationObjectName 은 UM3 세션 아이디를 나타내며 그 타입은 UM3ObjectName 타입 즉, CHARACTER STRING 타입을 나타냅니다.

The object name um3OperationObjectName stored in the N field of UM3 operation represents the UM3 session ID, and it is a UM3ObjectName type i.e. CHARACTER STRING type.

&ParameterType 표현은 UM3 오퍼레이션들에게 주어지는 파라메터들을 나타내며, 해당 파라메터는 본 권고안이 정의하는 모든 타입들이 가능합니다.

&ParameterType expression is for parameters given to UM3 operation, and any type defined in this recommendation can be used for the corresponding parameter.

13.8. OperationResponse Operation

UM3 오퍼레이션은 컴퓨터 프로그래밍 언어의 function 등과는 달리 직접 APDU 로 변환되어 상대측으로 전송되는 패킷을 의미하기도 합니다. 프로그래밍 언어의 function 이 그 수행 결과를 리턴하는 것은 UM3 오퍼레이션이 결과를 리턴 하는 것과는 다른 의미입니다. 즉, UM3 프로토콜이 정의하는 오퍼레이션의 리턴은 매니저가 송신한 오퍼레이션 APDU 를 수신한 에이전트가 해당 APDU 가 가리키는 특정 수신절차를 수행한 후 그 결과를 다시 매니저로 송신할 경우 해당 APDU 를 리턴 값으로 정의합니다.

Unlike the functions in computer programming languages, UM3 operation also means the packets that are converted to APDU and sent to the other end. Returning result by functions in other programming language has a different meaning from the results returned by UM3 operations. When an agent receives an operation APDU sent by a manager, it performs a certain task defined by the corresponding APDU, and the result is returned to the corresponding manager. In this case, the corresponding APDU is defined as a return value for the operations defined in the UM3 protocol.

UM3 프로토콜은 특정 오퍼레이션에 대한 리턴을 위해 OperationResponse 오퍼레이션을 별도로 정의합니다. 즉, 매니저가 송신한 특정 오퍼레이션 APDU 에 대해, 에이전트는 해당 오퍼레이션을 수행하고 그 결과를 OperationResponse 오퍼레이션을 이용하여 다시 매니저로 리턴하는 방식으로 정의되어 있습니다.

The UM3 protocol defines OperationResponse operation separately for the purpose of returning to a specific operation. In other words, the agent receives a specific operation APDU from the manager, performs the corresponding operation, and returns the result to the manager by using the OperationResponse operation.

일반적으로 특정 기능이 수행될 경우 그 결과는 성공 혹은 실패로 정의될 수 있습니다. 수행 결과가 성공일 경우, 해당 결과가 성공임을 나타내는 파라메터와 성공에 따른 부가적인 파라메터들이 필요합니다.

The result can be defined as success or failure when certain functions are performed in general. When the operation is a success, the parameter indicating the success of the corresponding operation and additional parameters associated with the successful operation are required.

그 수행 결과가 실패인 경우에도, 실패를 나타내는 파라메터와 실패에 따른 그 원인 등을 나타내는 파라메터가 필요합니다. 혹은 실패의 원인을 명확하게 전달하기 위해 추가적인 파라메터가 더 필요할 수도 있습니다.

When operation fails, the parameters indicating the failure of operation and other parameters associated with the cause of failures are required. Additional parameters may be required to clarify the cause of failure.

표 49은 OperationResponse 오퍼레이션의 파라메터 구조를 나타내는 한 예입니다. 표 49에서 '요청(REQUEST)' 항목은 OperationResponse 오퍼레이션이 결과를 리턴하도록 세션을 시작한 요청 오퍼레이션 입니다. 해당 요청 오퍼레이션은 OperationResponse 오퍼레이션을 제외한, 본 권고안이 정의하는 모든 UM3 서비스에 속한 오퍼레이션이 모두 적용될 수 있습니다. '성공' 및 '실패'로 표기된 행은 상기 '요청'으로 표기된 오퍼레이션에 대한 응답을 전송하는 OperationResponse 오퍼레이션의 파라메터 구조를 나타냅니다. 본 권고안은 오퍼레이션의 파라메터의 구조 즉, 타입과 파라메터의 이름 및 이들의 순서 등을 구분할 때 이를 signature 라는 이름으로 구분합니다.

표 49 OperationResponse operation. The 'REQUEST' item in Table 49 is the operation to request the session where the result is returned by OperationResponse operation. The corresponding request operation can be applied to all operations of UM3 services defined in this recommendation. The rows with SUCCESS or FAIL contain the structure of parameters for the OperationResponse operation that reruns the response for the operation in the REQUEST row. In this recommendation, the name called signature is used to identify the structure of parameters for the operation i.e. type, name of parameters, and their sequences.

본 권고안이 정의하는 오퍼레이션은 객재지향 개념에서 빌어온 polymorphism 개념을 활용합니다. 즉, 표 49 에서 보는 바와 같이 성공 및 실패의 두 가지 경우 모두 동일한 오퍼레이션이지만 서로 상이한 타입의 파라메터들을 갖고 있습니다. 즉, 서로 다른 signature 를 갖고 있으며 본 권고안은 거의 대부분의 객체지향 프로그래밍 언어들이 정의하는 것처럼 이를 operation overloaded 로 정의합니다.

The concept of polymorphism borrowed from the object oriented concept is used for the operations defined in this

recommendation. Both cases of SUCCESS and FAIL are the same operation as shown in Table 49, but they have different types of parameters. In other words, they have different signatures, and it is defined as an operation overloading in this recommendation as it is defined in most of other object oriented programming languages.

표 49-OperationResponse 오퍼레이션의 구조 [Structure of OperationResponse operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	- -	- -	
SUCCESS	UM3Boolean	parResult: TRUE	
SUCCESS	UM3Boolean ANY DEFINED BY UM3 ASN.1 module	parResult: TRUE parAnyTypeValue	OVLD 1
SUCCESS	UM3Boolean UM3ObjectList	parResult: TRUE parUM3ObjectList	OVLD 2
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult : FALSE parUM3OperationErrorCode parErrorElementIndex	OVLD 4
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16 UM3ClassIdentifier UM3ObjectName	parResult : FALSE parUM3OperationErrorCode parErrorElementIndex parUM3AttributeClassIdentifier parUM3AttributeObjectName	OVLD 5
FAIL	UM3Boolean UM3OperationErrorCode UM3ObjectList	parResult: FALSE parUM3OperationErrorCode parUM3ObjectList	OVLD 6
FAIL	UM3Boolean UM3OperationErrorCode UM3ClassIdentifier UM3ObjectName	parResult: FALSE parUM3OperationErrorCode parUM3ClassIdentifier parUM3ObjectName	OVLD 7
FAIL	UM3Boolean UM3OperationErrorCode UM3ClassIdentifier UM3ObjectName UM3ClassIdentifier UM3ObjectName	parResult: FALSE parUM3OperationErrorCode parUM3ObjectClassIdentifier parUM3ObjectObjectName parUM3AttributeClassIdentifier parUM3AttributeObjectName	OVLD 8
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16 UM3UnsignedInteger16	parResult: FALSE parUM3OperationErrorCode parErrorObjectElementIndex parErrorAttributeElementIndex	OVLD 9

표 49 의 OVLD 1 과 OVLD 2, OVLD 3 등은 overloaded 의 약자이며 이는 그 위의 '성공' 오퍼레이션에 대한 overloaded 오퍼레이션들로 정의합니다. 즉, 동일한 오퍼레이션의 이름 OperationResponse 를 갖고 있으나 그 용도에 따라 해당 오퍼레이션을 구성하는 파라메터의 종류와 순서가 다른 경우를 OVLD 로 정의합니다.

OVLD in OVLD 1, OVLD 2, and OVLD 3 stands for overloaded, and they are overloaded operations for SUCCESS. They have same names for OperationResponse operation, but the type and sequence of parameters of the corresponding operation are defined differently depending on the requirement. In this case, it is defined as OVLD.

특정 오퍼레이션에 대한 결과를 성공으로 기록하여 전송할 경우, 표 49 의 '성공'으로 구분된 행들에서와 같이 parResult 파라메터의 값은 항상 TRUE 로 지정되어 전송됩니다. 특히 첫 번째 행의 '성공' signature 는 UM3Ack 와 동일한 의미로 사용되는 OperationResponse 오퍼레이션의 signature 입니다. 본 권고안이 정의하는 OperationResponse 오퍼레이션의 '성공' signature 는 3가지 종류가 있으며, 각각은 parResult 파라메터와 더불어 parAnyTypeValue 혹은 parUM3ObjectList 파라메터를 갖고 있습니다.

When the result of a specific operation is SUCCESS and the result is returned, the value of the parResult parameter is set to TRUE when it is returned as shown in the rows with SUCCESS in Table 49. The SUCCESS signature in the first column is the signature used for OperationResponse operation and has the same meaning as UM3Ack.

'실패'로 구분된 7 개의 행은 본 권고안이 정의하는 7개의 '실패' signature 를 나타냅니다. 7개의 '실패' signature 는 본 권고안이 정의하는 모든 오퍼레이션에 대한 응답 signature 로 사용할 수 있습니다. 즉, 해당 오퍼레이션의 결과가 '실패'일 경우 상기 7 개의 '실패' signature 들 중 하나로 그 응답을 처리할 수 있음을 뜻합니다. 각 signature 에 대한 설명은 각 오퍼레이션에 대한 정의를 기술한 13.10 절부터 13.30 절의 각 오퍼레이션에 대한 정의중 '실패 signature' 에 대한 정의에서 참조할 수 있습니다.

위의 OperationResponse 오퍼레이션에 대한 ASN.1 정규표현식으로의 표현은 아래와 같습니다.

There are three types of SUCCESS signatures for the Response operations defined in this recommendation, and each one includes the parResult parameter along with the parAnyTypeValue or parUM3ObjectList parameter.

Seven rows with FAIL show the seven FAIL signatures defined in this recommendation. Any of the seven FAIL signatures can be used for the response signature of any operation defined in this recommendation. In other words, one of seven FAIL signatures can be used as a response when the result of the corresponding operation is FAIL. Refer to the definition of 'FAIL signature' in the definition of each operation from Section 13.10 to Section 13.30 where each operation is defined.

OperationResponse operation can be defined by using ASN.1 regular expressions as follows.

OperationResponse ::=

```
        OperationResponseSuccessAck |
        OperationResponseSuccessOvld1 |
        OperationResponseSuccessOvld2 |
        OperationResponseFailureOvld3 |
        OperationResponseFailureOvld4 |
        OperationResponseFailureOvld5 |
        OperationResponseFailureOvld6 |
        OperationResponseFailureOvld7 |
        OperationResponseFailureOvld8 |
        OperationResponseFailureOvld9

OperationResponseSuccessAck ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT TRUE
}

OperationResponseSuccessOvld1 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT TRUE,
        &parAnyTypeValue                ANY DEFINED BY UM3 ASN.1 module
}

OperationResponseSuccessOvld2 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT TRUE,
        &parUM3ObjectList               UM3ObjectList
}

OperationResponseFailureOvld3 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode
}

OperationResponseFailureOvld4 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parErrorElementIndex           UM3UnsignedInteger16
}

OperationResponseFailureOvld5 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
```

```
        &parErrorElementIndex           UM3UnsignedInteger16,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
        &parUM3ObjectName               UM3ObjectName
}

OperationResponseFailureOvld6 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parUM3ObjectList               UM3ObjectList
}

OperationResponseFailureOvld7 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
        &parUM3ObjectName               UM3ObjectName
}

OperationResponseFailureOvld8 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
        &parUM3ObjectName               UM3ObjectName
        &parUM3AttributeClassIdentifier UM3ClassIdentifier,
        &parUM3AttributeObjectName      UM3ObjectName
}

OperationResponseFailureOvld9 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parErrorObjectElementIndex     UM3UnsignedInteger16,
        &parErrorAttributeElementIndex  UM3UnsignedInteger16
}
```

이상과 같은 OperationResponse 오퍼레이션이 UM3 APDU 로 변환되는 모든 경우 즉, OperationResponseFailureOvld9 등과 같은 오브젝트들의 UM3OperationClassIdentifier 값은 모두 동일한 0 의 값을 갖고 있어야 합니다.

When the OperationResponse operation is converted to UM3 APDU, the UM3OperationClassIdentifier values of all

objects like OperationResponseFailureOvld9 should have the same value of 0.

13.9. OperationResponse Operation and UM3Ack

경우에 따라 매니저의 요청에 따른 OperationResponse 오퍼레이션이 오퍼레이션에 대한 인지여부 만을 리턴하는 경우가 있습니다. 특히, RequestChangeOfAttributeValueReport, RequestConditionDetectedReport 오퍼레이션의 경우 각 오퍼레이션에 대한 응답은 별도의 UM3 세션에서 ChangeOfAttributeValueReport, ConditionDetectedReport 오퍼레이션에 의해 이루어지게 됩니다.

Sometimes, the OperationResponse operation returns only the acknowledge response to the operation requested by the manager when required. Especially in the case of RequestChangeOfAttributeValueReport or RequestConditionDetectedReport operation, the response to each operation is made through ChangeOfAttributeValueReport or ConditionDetectedReport operation.

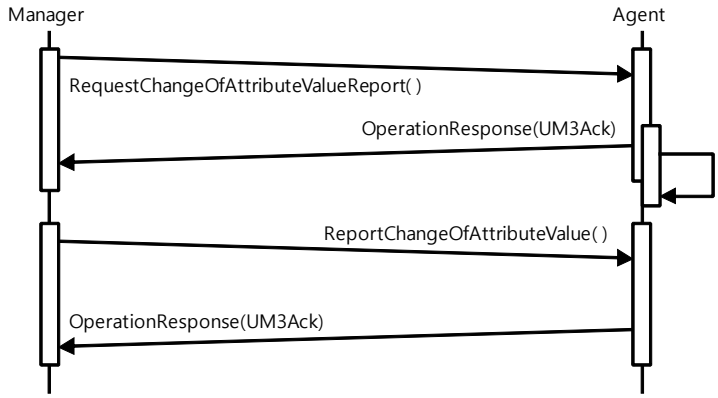


그림 16-OperationResponse 오퍼레이션과 UM3Ack 의 활용
Use of OperationResponse operation and UM3Ack]

그림 16-OperationResponse 오퍼레이션과 UM3Ack 의 활용 은 RequestChangeOfAttributeValueReport 오퍼레이션에 대한 에이전트의 응답 순서를 보여주고 있습니다. 즉, RequestChangeOfAttributeValueReport 오퍼레이션은 그 결과를 connectionless UM3 session 의 형태로 리턴받아야 합니다. 따라서, 에이전트는 해당 오퍼레이션에 대한 인지 여부만을 먼저 리턴해야하며 이때 OperationResponse 오퍼레이션의 UM3Ack 파라메터를 이용합니다. 또한 매니저가 ChangeOfAttributeValueReport 오퍼레이션을 통해 이벤

트 리포트를 수신하였을 경우에도 해당 APDU 의 정상수신을 나타내는 OperationResponse 오퍼레이션의 UM3Ack 파라메터를 이용합니다.

OperationResponse 오퍼레이션의 UM3Ack 파라메터는 다음과 같이 정의됩니다.

그림 16-OperationResponse 오퍼레이션과 UM3Ack 의 활용 shows the procedure of response for the agent to the RequestChangeOfAttributeValueReport operation. The RequestChangeOfAttributeValueReport operation should receive the return response in the form of a connectionless UM3 session. Therefore, the agent should return the acknowledge response to the corresponding operation first, and the UM3Ack parameter of the OperationResponse operation should be used. When the manager receives the event report through the ChangeOfAttributeValueReport operation, the UM3Ack parameter of the OperationResponse operation that indicates the correct reception of the corresponding APDU is used.

The UM3Ack parameter of the OperationResponse operation is defined as follows.

```
OperationResponse ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT TRUE
}
```

즉, OperationResponse 오퍼레이션의 parResult 파라메터가 항상 TRUE 값으로 설정되어 전송되는 경우, 이를 UM3Ack 로 정의합니다.

If the parResult parameter of the OperationResponse operation is always set to TRUE when it is returned, then it is defined as UM3Ack.

13.10. GetObjectValue operation

GetObjectValue 오퍼레이션은 매니저가 에이전트를 통해 서비스관리 정보모델 혹은 응용서비스 정보모델에 정의되어 있는 특정 오브젝트 전체의 값 즉, 모든 애트리뷰트 값들을 가져오기 위해 사용합니다. 이는 앞서 설명한 바와 같이 원격의 센서 혹은 제어장치의 상태정보 값을 가져와 매니저가 관리하는 정보모델내의 오브젝트들의 값과 동기화시키기 위한 오퍼레이션입니다.

값을 가져오기 위한 오브젝트의 지정은 UM3ClassIdentifier 와 UM3ObjectName 타입의 파라메터를 이용해 지정합니다.

The GetObjectValue operation is used for the manager to read the entire values of a specific object i.e. values of all

attributes, defined in the service management information model or application service information model through the agent. This operation is used to read the status information of the remote sensor or control devices or to synchronize the values of the objects in the information model managed by the manager as explained earlier.

The object to read the value is determined by using the parameters of UM3ClassIdentifier and UM3ObjectName types.

13.10.1. Structure of Signature and Return Type

GetObjectValue 오퍼레이션의 파라메터 타입, 리턴 타입 및 값 등은 표 50-GetObjectValue 오퍼레이션과 OperationResponse 오퍼레이션의 signature와 같습니다. 표 50-GetObjectValue 오퍼레이션과 OperationResponse 오퍼레이션의 signature에서 '구분'은 각각 '요청 (REQUEST)', '성공 (SUCCESS)', '실패 (FAIL)'로 명기되어 있으며, '요청'은 오퍼레이션을 요청할 때 필요한 파라메터의 타입과 이름 혹은 이름과 값을 나타냅니다. '성공'은 해당 오퍼레이션의 '요청'에 따라 그 결과를 송신함에 있어서, OperationResponse 오퍼레이션이 '성공'이라는 결과를 송신할 때의 파라메터 타입과 이름 혹은 이름과 그 값을 의미합니다. 마찬가지로 GetObjectValue

Parameter type, return type, and values of the GetObjectValue operation are shown in Table 49. Each row in the table is identified as REQUEST, SUCCESS, or FAILURE in the first column. The REQUEST row contains the type and name or name and value of parameters required to request the operation. The SUCCESS rows contains the type and name or name and value of the parameters required to return the SUCCESS result for OperationResponse operation based on the REQUEST from the corresponding operation. The FAIL rows contain the configuration of parameters in a similar way when the result of the GetObjectValue operation is FAIL.

표 50-GetObjectValue 오퍼레이션과 OperationResponse 오퍼레이션의 signature
[Signature of GetObjectValue operation and OperationResponse operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ClassIdentifier UM3ObjectName	parUM3ClassIdentifier parUM3ObjectName	
SUCCESS	UM3Boolean ANY DEFINED BY UM3 ASN.1 module	parResult: TRUE parAnyTypeValue	OVLD 1
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16 UM3ClassIdentifier UM3ObjectName	parResult : FALSE parUM3OperationErrorCode parErrorElementIndex parUM3AttributeClassIdentifier parUM3AttributeObjectName	OVLD 5

위와 같이 signature 를 구분하고 설명하는 방식은 이하 모든 오퍼레이션에 대해 동일하게 적용됩니다.

다음은 GetObjectValue 오퍼레이션의 ASN.1 정규표현식에 의한 정의입니다.

The method of classification and description shown in the table above applies to all operations described below.

The GetObjectValue operation is defined using ASN.1 regular expressions as follows.

```
GetObjectValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
        &parUM3ObjectName               UM3ObjectName
}
```

GetObjectValue 오퍼레이션의 signature 를 구성하는 UM3ClassIdentifier 와 UM3ObjectName 클래스 오브젝트는 UM3ObjectIndicator 타입을 이용하여 그 정보를 전달 할 수도 있습니다. 그러나, UM3 APDU 를 구성함에 있어서, 본 권고안은 가급적 불필요한 인코딩의 횟수를 줄이기 위해 요청 signature 에서 보는 바와 같이 단순히 UM3ClassIdentifier 와 UM3ObjectName 오브젝트를 바로 파라메터로 넘겨주도록 정의하였습니다.

UM3ClassIdentifier and UM3ObjectName class objects used for the signature of the GetObjectValue operation can send the information by using the UM3ObjectIndicator type. It is defined simply so that UM3ClassIdentifier and UM3ObjectName objects are passed as shown in the request signature to reduce unnecessary encoding operation in this recommendation.

위에서 정의한 GetObjectValue 오퍼레이션의 ASN.1 정규표현식으로의 정의에 있어서 parUM3OperationClassIdentifier 와 parUM3ObjectName 파라메터는 앞서 정의한 오퍼레이션의 클래스 아이디와 오브젝트 이름이며, 이는 오퍼레이션의 APDU 를 구성할 때 각각 T 필드와 N 필드로 기록됩니다. parUM3OperationClassIdentifier 와 parUM3ObjectName 파라메터에 대한 정의와 설명은 표 50-GetObjectValue 오퍼레이션과 OperationResponse 오퍼레이션의 signature에서는 생략하였습니다. 해당 파라메터 항목의 생략은 모두 공통된 사항이므로 이후 모든 오퍼레이션의 정의에서 동일하게 생략합니다.

In the above definition of the GetObjectValue operation using an ASN.1 regular expression, the parUM3OperationClassIdentifier and parUM3ObjectName parameter are the class ID and object name of the operations defined earlier, and they are stored in the T field and N field when APDU is configured. Definition and description of parUM3OperationClassIdentifier and parUM3ObjectName parameter are not included in Table 49. Since the corresponding parameters are common, they are not included in all subsequent definition of operations.

13.10.2. Request Signature

13.10.2.1. parUM3ClassIdentifier

특정 오브젝트의 클래스아이디를 나타냅니다. 해당 오브젝트는 물리적 자원을 모델링한 정보모델의 오브젝트뿐만 아니라 논리적 자원을 나타내는 오브젝트들 까지, 본 권고안이 정의하는 모든 오브젝트들의 클래스 아이디가 그 값으로 지정될 수 있습니다.

This contains the class ID of a specific object. The class ID of any object defined in this recommendation including the objects of information model that models physical resources as well as logical resources can be used for the value of this object.

13.10.2.2. parUM3ObjectName

특정 오브젝트의 이름을 나타냅니다. 해당 애트리뷰트의 이름은 UM3 SER 인코딩 룰을 이용하여 송신할 때도 ‘parUM3ObjectName’ 이라는 값이 그대로 전송되어야 합니다.

This contains the name of a specific object. The value of ‘parUM3ObjectName’ should be passed without modification when passing the name of the corresponding attribute by using the UM3 SER encoding rule.

13.10.3. Success Signature OVLD 1

13.10.3.1. parResult: TRUE

GetObjectValue 오퍼레이션의 결과가 성공했을 경우 UM3Enumerated 클래스 타입의 파라메터 parResult 의 값이 TRUE 로 지정되어 전송됩니다.

When the result of GetObjectValue operation is a success, the value of the parameter parResult of UM3Enumerated class type is set to TRUE and the result is returned.

13.10.3.2. parAnyTypeValue

GetObjectValue 오퍼레이션의 요청 signature에는 본 권고안이 정의하는 모든 정보모델의 클래스아이디가 파라메터로 주어질 수 있습니다. 즉, 애트리뷰트의 값을 모두 갖고 오기 위한 클래스에는 그 제한

이 없습니다. GetObjectValue 오퍼레이션의 결과에 따라 리턴되는 값은 모든 종류의 클래스 오브젝트가 리턴 가능합니다. 즉, ANY DEFINED BY UM3 ASN.1 module 의 타입으로 정의됩니다.

The class ID of any information model defined in this recommendation can be assigned to the request signature of the GetObjectValue operation as a parameter, and there is no restriction in the class to read any value of attributes. Any type of class object can be returned depending on the result of the GetObjectValue operation. In other words, it is defined as the type of ANY DEFINED BY in the UM3 ASN.1 module.

리턴되는 오브젝트는 클래스가 인스턴스화 된 결과이며, 따라서 모든 애트리뷰트들도 그 값을 갖고 있는 형태로 리턴됩니다.

The returned object is the result of instantiation of the class, and all attributes are returned with the values.

본 권고안이 정의하는 UM3 SER 인코딩 룰을 적용할 경우 해당 오브젝트와 그 오브젝트를 구성하는 애트리뷰트들은 모두 15.24과 15.25에서 정의한 방식으로 그 값들을 인코딩하여 송신하게 됩니다.

When applying the UM3 SER encoding rules defined in this recommendation, the values of all corresponding objects and attributes configuring the object are encoded with the method defined in Section 15.24 and 15.25 and the result is returned.

13.10.4. FAIL Signature OVLD 3

13.10.4.1. parResult: FALSE

GetObjectValue 오퍼레이션의 결과가 실패했을 경우, UM3Boolean 타입의 parResult 파라메터의 값은 FALSE 로 기록되어 전송됩니다.

When the result of GetObjectValue operation fails, the value of parameter parResult of UM3Boolean type is set to FALSE and the result is returned.

13.10.4.2. parUM3OperationErrorCode

UM3OperationErrorCode 타입의 오브젝트에는 오퍼레이션이 실패한 원인이 기록되어 송신됩니다. 실패한 원인의 종류에는 여러 가지가 있을 수 있으며 해당 실패원인을 나타내는 코드는 160 페이지의 표 40-UM3OperationErrorCode 타입의 코드별 의미와 같습니다.

The Object of UM3OperationErrorCode type contains the cause of failure when it is returned. There could be many reasons of failure, and the codes that represent the corresponding failure are shown in Table 40 in the page 129.

13.10.5. FAIL Signature OVLD 5

13.10.5.1. parResult : FALSE

GetObjectValue 오퍼레이션의 결과가 실패했을 경우, UM3Boolean 타입의 parResult 파라메터의 값은 FALSE 로 기록되어 전송됩니다.

When the result of GetObjectValue operation fails, the value of parameter parResult of UM3Boolean type is set to FALSE and the result is returned.

13.10.5.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 오브젝트에는 오퍼레이션이 실패한 원인이 기록되어 송신됩니다. 실패한 원인의 종류에는 여러 가지가 있을 수 있으며 해당 실패원인을 나타내는 코드는 160 페이지의 표 40-UM3OperationErrorCode 타입의 코드별 의미와 같습니다.

The Object of UM3OperationErrorCode type contains the cause of failure when it is returned. There could be many reasons of failure, and the codes that represent the corresponding failure are shown in Table 40 in the page 129.

13.10.5.3. parErrorElementIndex

여러 개의 애트리뷰트를 처리하는 과정에 있어서 맨 첫 번째 처리 대상 애트리뷰트를 0 번으로 지정하였을 때, 해당 첫 번째 애트리뷰트를 기준으로 몇 번째 애트리뷰트에서 오류가 발생했는지를 나타내는 인덱스 입니다. 참고로 본 권고안이 정의하는 클래스의 애트리뷰트들은 인코딩 시에 순서에 관계없이 인코딩이 이루어집니다. 따라서, 상기 parErrorElementIndex 가 나타내는 순서와 더불어 아래의 parUM3AttributeClassIdentifier 와 parUM3AttributeObjectName 파라메터를 이용해서 해당 애트리뷰트의 정확한 정보를 리턴합니다.

When the first target attribute for processing is set to 0 while processing multiple attributes, it is an index that indicates which attribute caused the error from the first attribute. As reference information, the attributes of the classes defined in this recommendation are encoded without a specific order. Therefore, accurate information about the corresponding attribute can be returned using the order indicated by this parErrorElementIndex and the parUM3AttributeClassIdentifier and parUM3AttributeObjectName parameter described below.

13.10.5.4. parUM3AttributeClassIdentifier

오류가 발생한 애트리뷰트의 클래스 아이디 값을 갖고 있습니다.

This contains the class ID of the attribute that caused an error.

13.10.5.5. parUM3AttributeObjectName

오류가 발생한 애트리뷰트의 오브젝트 이름을 갖고 있습니다. 해당 파라메터에 기록되는 값은 해당 애트리뷰트가 속한 레벨을 기준으로 구성된 UM3 RDN 으로 정의합니다.

This contains the object name of the attribute that caused an error. The value stored in the corresponding parameter is defined as UM3 RDN configured with the level that the corresponding attribute belongs to as reference.

13.10.6. Handling Procedure at the Receiver

GetObjectValue 오퍼레이션 명령을 수신한 에이전트는 자신이 관리하는 정보모델에서 parUM3ClassIdentifier 값의 클래스아이디와 parUM3ObjectName 값의 오브젝트 이름을 갖는 오브젝트를 찾은 후, 해당 오브젝트를 OperationResponse 오퍼레이션의 파라메터로 넘겨주어 결과를 송신합니다.

When the agent receives the GetObjectValue operation command, it searches for the object with the class ID of the parUM3ClassIdentifier value and object name of the parUM3ClassIdentifier value from the information model it manages and returns the corresponding object as a parameter to the OperationResponse operation.

만약 GetObjectValue 오퍼레이션 APDU를 통해 송신된 클래스아이디와 오브젝트 이름에 해당하는 오브젝트가 없을 경우, 오브젝트의 상태가 서비스 중지, 고장 등 정상적이지 않은 상태일 경우에는 해당 상태 값에 해당하는 코드를 UM3OperationErrorCode 타입 오브젝트의 값으로 지정한 후 매니저 측으로 전송합니다. 즉, 상기와 같은 경우에는 실패 signature OVLD 1 을 이용하여 리턴 값을 전송합니다.

If there is no object corresponding to the class ID and object name that is received through the GetObjectValue operation APDU and if the status of object is in an abnormal state such as service suspension or malfunction, then the corresponding error code of the corresponding state is stored in the value of the UM3OperationErrorCode type object and it is returned to the manager. In other words, the return value is sent using the FAIL signature OVLD 1.

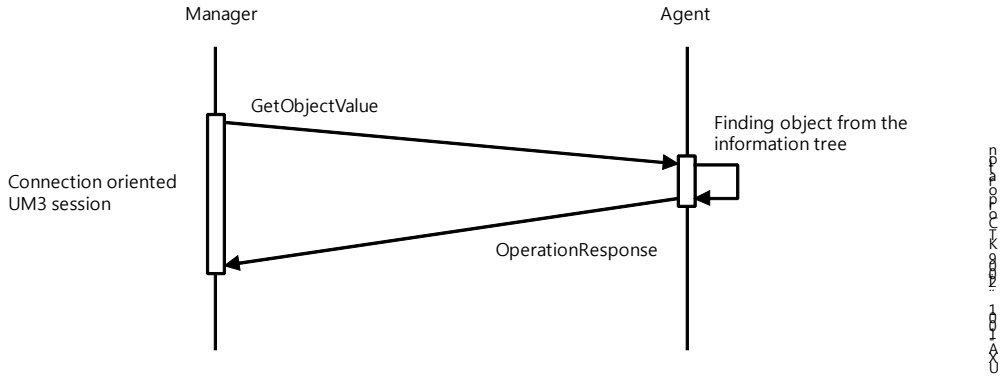


그림 17-GetObjectValue 오퍼레이션의 시퀀스다이어그램 [Sequence Diagram of GetObjectValue operation]

앞서의 예와는 달리 특정 오브젝트를 찾은 후 해당 오브젝트의 애트리뷰트 값들을 하나씩 인코딩 하는 과정 즉, 애트리뷰트 값들을 가져오기 위한 수행과정 중 에러가 발생했다면 실패 signature OVLD 5 가 사용됩니다.

Unlike the previous example, FAIL signature OVLD 5 should be used if there is error while encoding the values of the corresponding object, one by one, after finding the specific object i.e. while reading the attribute values.

구현 측면에서 상기 실패 signature OVLD 5 를 사용하는 경우에 있어서, 시스템의 리소스 즉, 메모리가 모자라는 경우, 우선순위가 더 높은 에이전트 내부의 처리 등에 의한 경우가 해당됩니다. 다만, 거의 대부분의 경우 컴퓨터 장치의 전원이 끊어지거나 혹은 하드웨어 장치의 심각한 에러가 발생하지 않는 한 표 50-GetObjectValue 오퍼레이션과 OperationResponse 오퍼레이션의 signature 의 실패 signature OVLD 5 가 사용되는 경우는 거의 없습니다.

Examples of this FAIL signature OVLD 5 in a practical system include cases in which the system runs out of memory or failure is caused by another agent with higher priority. FAIL signature OVLD 5, however, as shown in Table 49 is seldom used unless there is a fatal system error such as a computer system power outage or serious error in hardware devices.

13.11. GetObjectAttributeValue operation

GetObjectAttributeValue 오퍼레이션은 매니저가 에이전트를 통해 서비스관리 정보모델 혹은 응용서비스 정보모델에 정의되어 있는 특정 오브젝트의 특정 애트리뷰트의 값을 가져오기 위해 사용합니다. 마찬가지로 이는 원격 장치의 상태 값을 가져와 매니저가 관리하는 정보모델내의 해당 오브젝트의 애트리뷰트의 값을 가져온 값으로 동기화시키기 위한 목적으로 사용합니다.

The GetObjectAttributeValue operation is used for the manager to read the value of a specific attribute of an object defined in the service manage information model or application service information model through an agent. In a similar way, it is also used for the purpose of synchronization of the value of attributes of the corresponding object in the information model managed by the manager with the status value obtained from the remote device.

값을 가져오기 위한 오브젝트와 해당 오브젝트가 갖고 있는 애트리뷰트의 지정은 오브젝트를 위한 UM3ClassIdentifier 와 UM3ObjectName 타입의 파라메터 및 애트리뷰트를 위한 UM3ClassIdentifier 와 UM3ObjectName 타입의 파라메터를 이용해 지정합니다.

The object and attributes of the corresponding object for reading the value should be assigned by using the parameters of UM3ClassIdentifier and UM3ObjectName type for the parameter and attributes of the UM3ClassIdentifier and UM3ObjectName type for the corresponding object.

13.11.1. Structure of Signature and Return Type

GetObjectAttributeValue 오퍼레이션의 파라메터 타입, 리턴 타입 및 값 등은 표 51-GetObjectAttributeValue 오퍼레이션과 OperationResponse 오퍼레이션의 signature 과 같습니다.

Parameter type, return type, and value of the GetObjectAttributeValue operation are shown in Table 50.

표 51-GetObjectAttributeValue 오퍼레이션과 OperationResponse 오퍼레이션의 signature
[Signature of GetObjectAttributeValue operation and OperationResponse operation]

Classification	Type of Parameter	Name and Value of Parameter	Characteristics
REQUEST	UM3ClassIdentifier UM3ObjectName UM3ClassIdentifier UM3ObjectName	parUM3ClassIdentifier parUM3ObjectName parUM3AttributeClassIdentifer parUM3AttributeObjectName	
SUCCESS	UM3Boolean ANY DEFINED BY UM3 ASN.1 module	parResult: TRUE parAnyTypeValue	OVLD 1
FAIL	UM3Boolean UM3OperationErrorCode	parResult : FALSE parUM3OperationErrorCode	OVLD 3

다음은 위의 GetObjectAttributeValue 오퍼레이션의 ASN.1 정규표현식에 의한 정의입니다.

GetObjectAttributeValue operation can be defined by using ASN.1 regular expressions as follows.

```
GetObjectAttributeValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier              UM3ClassIdentifier,
        &parUM3ObjectName                   UM3ObjectName,
        &parUM3AttributeObjectIdentifier    UM3ClassIdentifier,
        &parUM3AttributeObjectName          UM3ObjectName
}
```

GetObjectAttributeValue 오퍼레이션은 특정 오브젝트의 한 개의 애트리뷰트 값을 가져오기 위한 용도로 사용됩니다.

The GetObjectAttributeValue operation is used for the purpose of reading value of one attribute of a specific object.

13.11.2. Request Signature

13.11.2.1. parUM3ClassIdentifier

가져오고자 하는 애트리뷰트가 속한 오브젝트의 클래스아이디를 나타냅니다.

This contains the class ID of the object that has the attributes to read.

13.11.2.2. parUM3ObjectName

가져오고자 하는 애트리뷰트가 속한 오브젝트의 오브젝트 이름을 나타냅니다. 앞서 정의한 바와 같이 에이전트가 속한 레벨을 기준으로 에이전트 레벨에 대한 UM3 RDN 이 저장되게 됩니다.

This contains the object name of the object to which attributes to read belong. As defined earlier, UM3 RDN corresponding to the agent level based on the level to which the agent belongs is stored as reference.

13.11.2.3. parUM3AttributeClassIdentifier

가져오고자 하는 애트리뷰트의 클래스 아이디를 나타냅니다.

This contains the class ID of the attribute to read.

13.11.2.4. parUM3AttributeObjectName

가져오고자 하는 애트리뷰트의 오브젝트 이름 즉, 애트리뷰트 이름을 나타냅니다. 애트리뷰트의 이름은 RDN (Relative Distinguished Name) 으로 지정되며 그 형식은 앞서 정의한 바와 같습니다. 특히 애트리뷰트의 이름은 본 권고안에서 정의하는 클래스 애트리뷰트의 이름을 그대로 사용해야 하며, 해당 표준이 지켜지지 않을 경우 통신 상대방은 해당 애트리뷰트를 구분하고 찾아낼 수 없게 됩니다.

This contains the object name of the attribute to read i.e. name of attribute. The name of the attribute is assigned as RDN (Relative Distinguished Name), and the format is as defined earlier. In particular, the name of the attribute defined in this recommendation should be used without modification, and it'd be impossible to identify and find the attribute at the other end of communication if this standard is not followed.

13.11.3. SUCCESS Signature OVLD 1

13.11.3.1. parResult: TRUE

성공했을 경우 parResult 파라메터의 값이 TRUE 로 지정되어 전송됩니다.

When the operation is a success, the value of the parResult parameter is set to TRUE and the result is returned.

13.11.3.2. parAnyTypeValue

OperationResponse 오퍼레이션의 파라메터로 GetObjectAttributeValue 오퍼레이션이 요청한 해당 애트리뷰트 오브젝트가 리턴됩니다. 해당 파라메터는 본 권고안이 정의하는 모든 종류의 타입들이 저장될 수 있습니다. 즉, ANY DEFINED BY UM3 ASN.1 module 의 타입으로 정의됩니다.

만약 가져오고자 하는 애트리뷰트의 타입이 UM3 기본 타입이 아닌 클래스 타입 혹은 복합형식의 타입일 경우에도, 에이전트는 본 권고안이 정의하는 UM3 SER 인코딩 룰에 의해 해당 클래스 타입 혹은 복항형식의 타입을 인코딩 한 후 그 데이터를 송신하게 됩니다.

The attribute object requested by the GetObjectAttributeValue operation is returned as a parameter of the OperationResponse operation.

Any type defined in this recommendation can be used as the parameter. In other words, the type is defined as the ANY DEFINED BY UM3 ASN.1 module.

If the type of the attribute to read is class type or complex type instead of UM3 primitive type, then the agent encodes the class type or complex format type according to the UM3 SER encoding rule defined in this recommendation and sends data.

13.11.4. FAIL Signature OVLD 3

13.11.4.1. parResult: FALSE

오퍼레이션이 실패했을 경우 parResult 파라메터의 값은 FALSE 로 지정되어 전송됩니다.

When an operation fails, the value of the parResult parameter is set to FALSE and the result is returned.

13.11.4.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 오브젝트에는 오퍼레이션이 실패한 원인이 기록되어 전송됩니다. 에러코드의 종류와 그 내용의 구성은 160 페이지의 표 40-UM3OperationErrorCode 타입의 코드별 의미와 같습니다.

The cause of failure for the operation is stored in the object of the UM3OperationErrorCode class type and the result is returned. The types and contents of the error codes are shown in Table 40 on page 129.

13.11.5. Handling Procedure at the Receiver

GetObjectAttributeValue 오퍼레이션에 명령을 수신한 에이전트는 자신이 관리하는 정보모델에서 parUM3ClassIdentifier 파라메터가 갖고 있는 값의 클래스아이디와 parUM3ObjectName 파라메터가 갖고 있는 값의 오브젝트를 찾습니다. 해당 오브젝트를 찾은 후 다시 parAttributeClassIdentifier 값의 클래스 아이디와 parAttributeObjectName 값을 갖고 있는 애트리뷰트를 찾아내어 해당 애트리뷰트 오브젝트를 OperationResponse 오퍼레이션의 파라메터로 넘겨주어 결과를 송신합니다.

When the agent receives the GetObjectValue operation command, it searches for the object with the class ID of the parUM3ClassIdentifier value and object name of the parUM3ClassIdentifier value from the information model it manages. It returns the corresponding object as a parameter to the OperationResponse operation.

만약 해당 오브젝트 혹은 애트리뷰트를 찾을 수 없거나, 다른 이유로 인해 정상적인 결과값의 리턴이 불가능할 경우에는 OperationResponse 오퍼레이션의 실패 signature 를 이용해 에러의 종류 등의 정보를 매니저로 송신합니다.

If the corresponding object or attribute cannot be found or the result cannot return normally for any other reason, then the error information including the type of error is returned to the manager by using the FAIL signature of the OperationResponse operation.

상기 parUM3AttributeObjectName 과 parUM3ObjectName 파라메터에는 DN 형태의 이름이 주어지게 됩니다. 즉, 그림 18-RDN과 DN [의 presentValue 는 mySensor 오브젝트의 애트리뷰트 입니다. 또한 mySensor 오브젝트는 myGateway 오브젝트의 애트리뷰트로 정의되어 있습니다. 이 때

GetObjectAttributeValue 오퍼레이션을 통해 가져오고자 하는 애트리뷰트의 값이 presentValue 애트리뷰트의 값이라면, 해당 presentValue 애트리뷰트의 DN 인 'myGateway.mySensor.presentValue' 를 parUM3AttributeObjectName 파라메터의 값으로 넘겨주어야 합니다. 이러한 DN 의 활용 방법은 본 권고안이 정의하는 모든 오퍼레이션에 동일하게 적용되어야 합니다.

The name in DN format will be stored in the parUM3AttributeObjectName and parUM3ObjectName parameter. In other words, the presentValue is the attribute of the mySensor object. The mySensor object is also defined as an attribute of the myGateway object. In this case, if the value of the attribute to read through the GetObjectAttributeValue operation and is the value of the presentValue attribute, 'myGateway.mySensor.presentValue', then the DN of the corresponding presentValue attribute should be passed to the value of the parUM3AttributeObjectName parameter. This method of using DN should be applied to all operations defined in this recommendation.

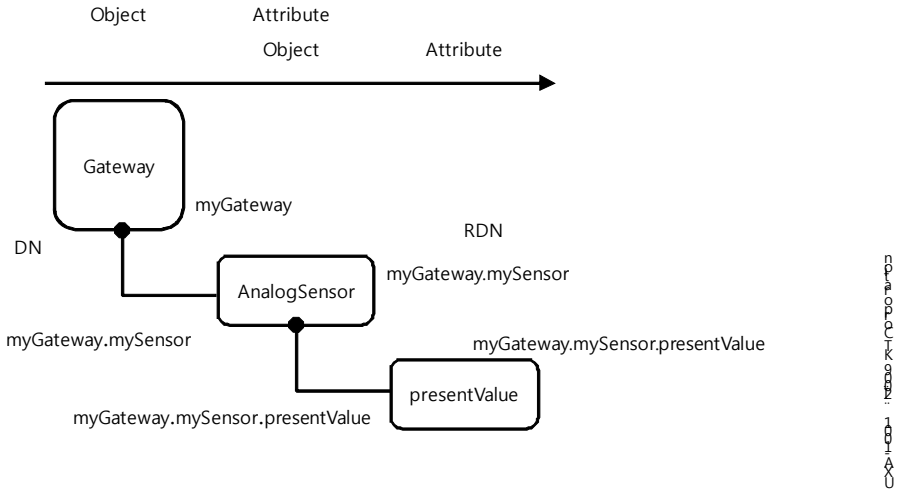


그림 18-RDN과 DN [RDN and DN]

그림 19-GetObjectAttributeValue 오퍼레이션의 시퀀스다이어그램 [에는 GetObjectAttributeValue 오퍼레이션과 이에 대한 응답 OperationResponse 오퍼레이션이 connectionless UM3 session 의 형태로 데이터를 주고 받는 시퀀스다이어그램 (sequence diagram) 이 표현되어 있습니다. 이는 UM3 프로토콜을 사용하는 모든 매니저와 에이전트는 본 권고안이 정의한 connection oriented UM3 session 과 connectionless UM3 session 의 두 가지 형태로 데이터를 송수신 할 수 있으며, 그 방식의 결정은 구현의 측면에서 결정할

수 있음을 나타냅니다. 따라서, 그림 19-GetObjectAttributeValue 오퍼레이션의 시퀀스다이어그램 [과는 달리 GetObjectAttributeValue 오퍼레이션은 connection oriented UM3 session 의 형태로 데이터를 송수신할 수도 있으며 이는 매니저와 에이전트간의 송수신 세션의 방식을 미리 결정하여 구현하는 것으로 정의합니다.

그림 19-GetObjectAttributeValue 오퍼레이션의 시퀀스다이어그램 [is a sequence diagram showing that the GetObjectAttributeValue operation and the corresponding OperationResponse operation exchanges data in the form of a connectionless UM3 session. All managers and agents using UM3 protocol can use two types of sessions, including connection oriented UM3 session and connectionless UM3 session as defined in this recommendation for exchanging data. The method can be determined from the perspective of implementation. Unlike Fig. 19, the GetObjectAttributeValue operation can exchange data in the form of a connection oriented UM3 session, and the method of data transfer session between manager and the agent should be determined in advance.

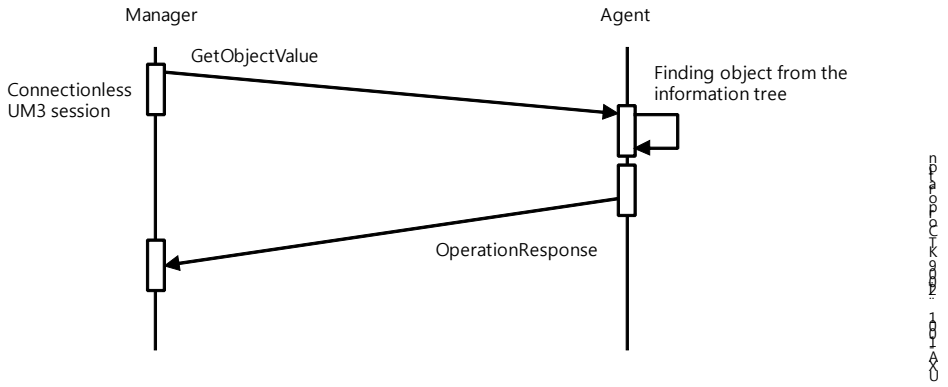


그림 19-GetObjectAttributeValue 오퍼레이션의 시퀀스다이어그램
[Sequence diagram of GetObjectAttributeValue operation]

단, 본 권고안이 정의하는 오퍼레이션들 중 UM3-EVENT-REPORT 서비스를 제외한 나머지 서비스 분류에 속한 모든 오퍼레이션들은 connection oriented UM3 session 방식을 사용해야 합니다.

All operations defined in this recommendation that belong to the service categories other than UM3-EVENT-REPORT service, however, should use the connection oriented UM3 session method.

13.12. GetObjectGroupValue operation

GetObjectGroupValue 오퍼레이션은 1 개 이상의 복수개의 오브젝트에 대해 해당 오브젝트들이 갖고 있는 모든 애트리뷰트의 값들을 가져오기 위해 사용됩니다. 해당 오퍼레이션은 가져오기 위한 오브젝트를 선택할 수는 있으나 해당 오브젝트의 애트리뷰트들을 선택적으로 가져올 수는 없습니다. 이는 GetObjectGroupValue 오퍼레이션의 파라메터로 주어지는 파라메터의 특성에도 반영되어 있습니다. 즉, GetObjectGroupValue 오퍼레이션은 특정 오브젝트가 갖고 있는 애트리뷰트의 값에 따라 오브젝트들을 선택하고 해당 오브젝트들이 갖고 있는 모든 애트리뷰트의 값들을 가져오는 오퍼레이션입니다.

The GetObjectGroupValue operation is used to read all attribute values of more than one object. The operation can select the object to read, but the attributes of the corresponding object cannot be selectively obtained. This is also reflected in the characteristics of the parameters given for the GetObjectGroupValue operation. The GetObjectGroupValue operation selects objects based on the values of the attributes of a specific object and reads all the attributes of the corresponding objects.

13.12.1. Structure of Signature and Return Type

GetObjectGroupValue 오퍼레이션의 파라메터 타입, 리턴 타입 및 그 값 등은 표 52-GetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조 와 같습니다. 표 52-GetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 GetObjectGroupValue 오퍼레이션은 크게 3 가지 요청 signature 로 구분됩니다. 그 중 첫 번째는 가져오고자 하는 오브젝트의 클래스 아이디와 오브젝트 이름을 직접 지정하는 방법이며, 두 번째는 특정 오브젝트를 지정하지 않고 조건문을 이용하는 방법이며, 마지막 세 번째는 조건문을 적용할 오브젝트의 범위를 명시적으로 나타낸 후 조건문을 함께 전달하는 방법입니다.

The parameter types, return types, and values of the GetObjectGroupValue operation are shown in Table 51. As shown in Table 51, the GetObjectGroupValue operation is largely classified into three REQUEST signatures. The first method is to assign the class ID and object name of the object to read, the second method is to use the conditional statements without assigning the specific object, and the last method is to specify the scope of the objects to explicitly apply the conditional statements and pass the data with the conditional statements.

표 52-GetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조

[Structure of signature and return type of GetObjectGroupValue operation]

Classification	Type of Parameter	Name and Value of Parameter	Characteristics
REQUEST	UM3ObjectList	parUM3ObjectList	UM3ObjectIndicator element
REQUEST	UM3ObjectValueCondition	parUM3ObjectValueCondition	OVLD 1
REQUEST	UM3ObjectList	parUM3ObjectList	OVLD 2, UM3ObjectIndicator element
	UM3ObjectValueCondition	parUM3ObjectValueCondition	

SUCCESS	UM3Boolean UM3ObjectList	parResult: TRUE parUM3ObjectList	OVLD 2
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult: FALSE parUM3OperationErrorCode parErrorObjectElementIndex	OVLD 4, 1a, 3a
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16 UM3ClassIdentifier UM3ObjectName	parResult: FALSE parUM3OperationErrorCode parErrorObjectElementIndex parUM3AttributeClassIdentifier parUM3AttributeObjectName	OVLD 5, 1b, 3b
FAIL	UM3Boolean UM3OperationErrorCode UM3ClassIdentifier UM3ObjectName	parResult: FALSE parUM3OperationErrorCode parUM3ObjectClassIdentifier parUM3ObjectObjectName	OVLD 7, 2a
FAIL	UM3Boolean UM3OperationErrorCode UM3ClassIdentifier UM3ObjectName UM3ClassIdentifier UM3ObjectName	parResult: FALSE parUM3OperationErrorCode parUM3ObjectClassIdentifier parUM3ObjectObjectName parUM3AttributeClassIdentifier parUM3AttributeObjectName	OVLD 8, 2b

표 52-GetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 GetObjectGroupValue 오퍼레이션은 3가지 종류의 요청 signature를 지원해야 합니다. 즉, UM3ObjectList 타입의 파라메터에 기록된 엘리먼트들의 정보를 참조하여 오퍼레이션을 수행하고 그 결과로 해당 오브젝트들의 인스턴스를 리턴하는 경우가 있습니다. 또한 특정 오브젝트를 지정하지 않고 UM3ObjectValueCondtion 클래스의 오브젝트를 파라메터로 넘길 경우 해당 조건을 만족하는 모든 오브젝트를 리턴 하는 경우가 있습니다.

표 52-GetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조GetObjectGroupValue operation should support three types of REQUEST signature. There is a case that operation is performed with the information of the elements stored in the parameters of the UM3ObjectList type and the instance of the corresponding object is returned as the result. And also all objects that meet the conditions when the object of UM3ObjectValueCondtion class is passed as a parameter without specifying specific object.

마지막으로는 오브젝트의 대상 범위와 조건을 모두 지정하여 특정 오브젝트들 중 특정 조건을 만족하는 오브젝트들 만을 리턴 하는 경우가 있습니다.

Lastly, the target range of objects and conditions are defined. Only the objects in the range that meet the specific conditions can be returned.

본 권고안은 이렇게 동일한 오퍼레이션의 signature 만을 다르게 나타내야 하는 경우를 OVLD (overloaded) 로 표현합니다. 본 권고안이 정의하는 오퍼레이션은 그 파라메터의 타입과 순서에 따라 signature 가 구분됩니다. 즉, 오퍼레이션의 이름이 동일하다고 해도 각 파라메터의 순서가 다르거나 타입이 다를 경우에는 별개의 오퍼레이션으로 간주합니다. 본 권고안이 정의하는 UM3 프로토콜을 지원하는 소프트웨어 혹은 하드웨어 시스템은 모든 overloaded 오퍼레이션을 명확하게 구현하고 지원해야 합니다.

In this recommendation, the same operation with different signatures is referred as OVLD (overloaded). The signatures of the operations defined in this recommendation are classified based on the type and sequence of the parameters. In other words, operations with the same name and different sequences of parameter types are considered as separate operations. All software and hardware systems that support the UM3 protocol defined in this recommendation should clearly implement and support the overloaded operations.

표 52-GetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 3 가지 종류의 요청 signature 들 중 첫 번째인 UM3ObjectList 타입으로 파라메터가 주어질 경우, 해당 오퍼레이션에 대한 실패의 원인이 특정 오브젝트에 있을 경우에는 해당 오브젝트가 파라메터로 주어진 오브젝트들 중 몇 번째 오브젝트인가의 정보만으로도 해당 오류 발생 오브젝트를 확인할 수 있습니다. 그러나 특정 오브젝트의 애트리뷰트의 값을 가져오는 도중 발생하는 오류는 해당 애트리뷰트의 클래스 아이디와 오브젝트 이름을 함께 리턴 해야 매니저가 해당 오류 발생 애트리뷰트의 정보를 확인할 수 있게 됩니다.

If a parameter is given as UM3ObjectList type, the first REQUEST signature out of three types shown in Table 51, and the cause of failure for the corresponding operation is a specific object, then the object causing the error can be checked with the information about the index of the object in the list of objects passed as parameter. Errors that occur while reading the attribute values from specific object, however, can only be checked if the class ID and object name of the corresponding attribute are returned so that the manager can check the information of the attributes causing errors.

두 번째로 조건문 만이 주어졌을 경우에는 특정 오브젝트나 혹은 특정 오브젝트의 애트리뷰트 값을 가져오는 도중 발생하는 오류에 대해, 매니저가 해당 오브젝트나 애트리뷰트를 확인하기 위해서는 반드시 해당 오브젝트와 애트리뷰트의 클래스아이디와 오브젝트이름이 리턴되어야 합니다.

Secondly, if only the conditional statements are given, then the class ID and object name of the corresponding object and the attribute should be returned for the manager to check the object or attribute for the errors occurring while reading the attribute values from a specific object or attribute of a specific object.

세 번째로 오브젝트 리스트 즉, 조건문을 적용할 대상 오브젝트들의 범위와 조건문이 함께 주어질 경우에는, 오류가 발생하는 오브젝트는 해당 오류 발생 오브젝트의 번호 혹은 인덱스 만으로도 확인이 가능하며, 특정 오브젝트의 애트리뷰트에서 발생한 오류에 대해서는 해당 애트리뷰트가 속한 오브젝트의 클래스아이디와 오브젝트 이름, 해당 애트리뷰트의 클래스아이디와 오브젝트 이름에 관한 정보가 모두 필요하게 됩니다.

Thirdly, when both the list of objects specifying the range of objects to apply the conditional statement and the conditional statements are provided, then the object that causes the error can be identified by the number of the index of the object. More information, including class ID and object name of the object to which the corresponding attributes belong and the class ID and object name of the corresponding attribute, is required for an error that has occurred in an attribute of a specific object.

따라서, 이상과 같은 오류 발생 위치에 따른 실패 signature 는 모두 6 개로 구분되어야 하며, 오류 발생 위치에 관한 정보를 리턴 하지 않고 오류 발생 사실만을 리턴하는 signature 까지 모두 7 개의 signature 가 필요하게 됩니다. 단, UM3ObjectList 타입의 파라메터가 주어지는 두 가지 종류에 대한 실패 signature 는 해당 오브젝트의 인덱스 만을 리턴할 수 있으므로 전체적으로는 모두 5 개의 실패 signature 가 필요합니다.

다음은 에 표기된 OVLD 오퍼레이션들에 대한 의미를 다시 한 번 정리한 결과 입니다.

Therefore, the FAIL signature based on the error source location should be classified into six types, a total of seven signatures, including the one that only returns the information about the error occurrence without returning the information of the error source location, are required. Since there are only two types of FAIL signatures that provide the UM3ObjectList type parameter can return the index of the corresponding object, then five FAIL signatures are required overall.

The following table summarizes the meanings of OVLD operations.

표 53-GetObjectGroupValue 오퍼레이션의 요청 signature 별 실패 signature 의 유형
[Type of FAIL signature by request signature of GetObjectGroupValue operation]

Parameter \ Error Source Location	Object Error (a)	Attribute Error (b)
Object List Parameter (1)	FAIL signature OVLD 4	FAIL signature OVLD 5
Condition Parameter (2)	FAIL signature OVLD 7	FAIL signature OVLD 8
Object List and Condition Parameter (3)	FAIL signature OVLD 4	FAIL signature OVLD 5

다음은 상기 GetObjectGroupValue 오퍼레이션 중 요청 signature 및 요청 signature OVLD 1 과 OVLD 2

를 을 ASN.1 정규표현식으로 표현한 경우입니다.

The REQUEST signature and REQUEST signature OVLD1 and OVLD 2 of the GetObjectGroupValue operation can be defined by using ASN.1 regular expressions as follows.

```
GetObjectGroupValue ::=
        GetObjectGroupValueOvld1 |
        GetObjectGroupValueOvld2 |
        GetObjectGroupValueOvld3

GetObjectGroupValueOvld1 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &parUM3ObjectList               UM3ObjectList
}

GetObjectGroupValueOvld2 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &parUM3ObjectValueCondition     UM3ObjectValueCondition
}

GetObjectGroupValueOvld3 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &parUM3ObjectList               UM3ObjectList,
        &parUM3ObjectValueCondition     UM3ObjectValueCondition
}
```

지금까지 정의한 모든 UM3 오퍼레이션들과 마찬가지로 GetObjectGroupValue 오퍼레이션도 um3OperationClassIdentifier 와 um3OperationObjectName 애트리뷰트를 갖고 있습니다. 이들은 UM3 SER 을 이용해 인코딩 될 때 오퍼레이션 APDU 의 T 필드와 N 필드의 값으로 기록되게 됩니다.

상기 ASN.1 정규표현식에서 ListValuePair 타입은 GetObjectGroupValue 오퍼레이션의 파라메터를 정의하기 위해 임시로 사용한 타입으로 UM3ObjectList 와 UM3ObjerctValueCondition 타입 등 두 개의 파라메터로 구성됩니다.

Like the other UM3 operations defined above, the GetObjectGroupValue operation also has the um3OperationClassIdentifier and um3OperationObjectName attribute. They are stored in the T field and N field of the operation APDU when encoded by using UM3 SER.

The ListValuePair type in the above ASN.1 regular expression is used temporarily to define the parameters of the GetObjectGroupValue operation, and it contains two UM3ObjectList and UM3ObjectValueCondition type

parameters.

13.12.2. REQUEST signature

13.12.2.1. parUM3ObjectList

UM3ObjectList 타입의 파라메터 입니다. 가져오고자 하는 그룹에 속하는 오브젝트들의 클래스아이디와 오브젝트 이름을 UM3ObjectIndicator 타입을 이용하여 지정합니다. 즉, 가져오고자 하는 오브젝트들의 클래스 아이디와 오브젝트 이름만을 지정하며, 가져오고자 하는 애트리뷰트는 지정할 수 없습니다. 앞서 정의한 바와 같이 특정 오브젝트 만을 지정할 수 있으며, 특정 오브젝트를 지정한 후에는 해당 오브젝트의 모든 애트리뷰트들을 모두 가져오게 됩니다.

This is a parameter of UM3ObjectList type. class ID and object name of the objects that belong to the group to read are assigned by using the UM3ObjectIndicator type. In other words, only the class ID and object name of the objects to read can be assigned, and the attributes cannot be assigned. As defined earlier, only specific objects can be assigned, and all attributes of the corresponding object can be obtained after assigning a specific object.

13.12.3. REQUEST signature OVLD 1

13.12.3.1. parUM3ObjectValueCondition

UM3ObjectValueCondition 클래스 타입의 오브젝트이며 그룹에 속하는 오브젝트들의 값을 기준 값과 비교하기 위한 기준 값, 조건 등의 정보를 담고 있습니다.

This is an object of UM3ObjectValueCondition class type, and it contains information like the reference value and conditions to be used as a reference or comparison of objects that belong to the group.

13.12.4. REQUEST signature OVLD 2

해당 OLVD 오퍼레이션은 상기 요청 signature 들의 조합으로 이루어집니다. 즉, 대상이 되는 오브젝트 그룹을 직접 지정하고, 해당 그룹 내에서 조건을 비교하여 필요한 오브젝트들을 선택하는 기능을 제공하는 OLVD 오퍼레이션 입니다.

The corresponding OLVD operation is comprised of the combination of the REQUEST signatures. In other words, this OLVD operation provides the functions to directly assign a target object group and to select the required objects after comparing the conditions in the corresponding group.

13.12.4.1. parUM3ObjectList

UM3ObjectList 타입의 파라메터이며 그룹의 대상이 되는 오브젝트들의 클래스아이디, 오브젝트 이름 등이 UM3ObjectList 타입의 엘리먼트인 UM3ObjectIndicator 타입의 변수에 저장되어 있습니다.

This is a parameter of UM3ObjectList type. The class ID and object name of the objects of the target group are stored in the UM3ObjectIndicator type variable, an element of UM3ObjectList type.

```
UM3ObjectList ::= UM3 SEQUENCE OF UM3 SEQUENCE {
        um3ClassIdentifierIndicator        UM3ClassIdentifier,
        um3ObjectNameIndicator             UM3ObjectName
}
```

상기 ASN.1 정규표현식은 parUM3ObjectList 파라메터의 타입을 나타낸 표현입니다. 즉, UM3ObjectList 타입이 UM3 SEQUENCE OF 타입이며, UM3ObjectIndicator 타입이 UM3 SEQUENCE 타입으로 정의되어 있으므로 parUM3ObjectList 파라메터에 국한 해서 해당 ASN.1 정규표현식과 같이 표현할 수 있습니다. 앞서 정의한 UM3 SEQUENCE OF 와 UM3 SEQUENCE 타입의 정의와 같이 두 타입 모두 엘리먼트의 순서에 의미가 있는 점에 유의해야 합니다.

The ASN.1 regular expression above defines the type of parUM3ObjectList parameter. Since the UM3ObjectList type is UM3 SEQUENCE OF and UM3ObjectIndicator type is UM3 SEQUENCE in this definition, it can only be expressed as the corresponding ASN.1 regular expression for the parUM3ObjectList parameter. Like the UM3 SEQUENCE OF and UM3 SEQUENCE type that were defined earlier, the sequence has significance for both types.

13.12.4.2. parUM3ObjectValueCondition

UM3ObjectValueCondition 클래스 타입의 오브젝트이며 비교를 위한 기준 값, 비교 선택 조건 등의 정보가 저장되어 있습니다.

This is an object of UM3ObjectValueCondition class type, and it contains information like the reference value and comparison conditions for comparison operation.

13.12.5. SUCCESS signature OVLD 2

앞서 정의한 바와 같이 OperationResponse 오퍼레이션의 signature 를 나타내며, GetObjectGroupValue 오퍼레이션에 대한 성공 결과를 리턴 하는 경우입니다.

This is a signature of the OperationResponse operation as defined earlier, and the SUCCESS result is returned to the GetObjectGroupValue operation.

13.12.5.1. parResult: TRUE

오퍼레이션이 성공하였음을 나타내는 UM3Boolean 타입의 파라메터 입니다. 해당 파라메터의 값은 default 값인 TRUE 가 기록되어 전송됩니다.

This is the parameter of UM3Boolean type that indicates the success of operation. The value of the corresponding parameter is set to TRUE and the result is returned.

13.12.5.2. parUM3ObjectList

GetObjectGroupValue 오퍼레이션의 실행결과 만들어진 오브젝트 그룹의 오브젝트들을 나타냅니다. 즉, UM3 SEQUENCE OF 타입을 갖는 UM3ObjectList 타입의 오브젝트이며, 그 엘리먼트는 선택된 혹은 조건을 만족하는 모든 오브젝트들의 나열로 이루어집니다. 이 때 해당 오브젝트들은 에이전트가 관리하는 해당 오브젝트의 모든 애트리뷰트들과 그 값이 채워진 완전한 형태로 인코딩되어 리턴되어야 합니다.

This contains the objects in the object group that is created as a result of GetObjectGroupValue operation. In other words, it is an object of UM3ObjectList type with UM3 SEQUENCE OF type, and it contains the list of all objects that are selected or meet the conditions. In this case, all attributes and the values of the corresponding object that is managed by the agent should be completely filled before encoding.

13.12.6. FAIL signature OVLD 3

해당 실패 signature 는 오류가 발생한 개략적인 원인 만을 parUM3OperationErrorCode 에 기록하여 그 값을 리턴하는 OperationResponse 오퍼레이션의 signature 입니다. 실패한 원인이 특정 오브젝트나 혹은 특정 오브젝트의 특정 애트리뷰트에 있지 않을 경우 사용하는 signature 입니다. 경우에 따라서는 오류가 발생한 오브젝트나 애트리뷰트의 정보를 리턴하지 않고 오퍼레이션의 수행 결과만을 리턴하기 위해서 사용되기도 합니다.

This is a signature of the OperationResponse operation that stores rough cause of error in the parUM3OperationErrorCode and returns the values. This signature can be used to return the result of operation without returning the information of object or attribute causing errors when necessary.

13.12.6.1. parResult : FALSE

오퍼레이션의 수행 결과가 실패임을 나타내는 파라메터입니다. UM3Boolean 타입의 파라메터이며 그 default 값은 FALSE 로 기록되어 전송됩니다.

This is a parameter that indicates the failure of an operation. This parameter is UM3Boolean type, the default value is set to FALSE, and the result is returned.

13.12.6.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.12.7. FAIL signature OVLD 4

실패 signature OVLD 4 는 231 절과 13.12.4 절의 요청 signature, 요청 signature OVLD 2 에 대응하는 실패 signature 입니다. 요청 signature 에 포함된 UM3ObjectList 타입의 파라메터에는 가져오고자 하는 오브젝트들의 클래스아이디와 오브젝트 이름이 기록되어 있습니다. 따라서, 해당 파라메터를 구성하는 엘리먼트들의 순서를 리턴할 경우 매니저는 어떤 오브젝트에서 오류가 발생했는지를 쉽게 확인할 수 있습니다. 즉, 오퍼레이션의 수행 도중 오브젝트와 관련된 오류가 발생할 경우 해당 오브젝트의 정보를 함께 리턴하기 위한 실패 signature 입니다.

FAIL signature OVLD 4 corresponds to the REQUEST signature and REQUEST signature OVLD 2 in Section 13.12.2 and 13.12.4. The class ID and object name of the objects to read are stored in the UM3ObjectListType parameter included in the REQUEST signature. Therefore, the manager can easily check which object is associated with the error when returning the sequence of elements that configures the corresponding parameter. This FAIL signature is for returning the values with the information of the corresponding object error related to the object that occurs during operation.

13.12.7.1. parResult : FALSE

오퍼레이션의 수행 결과가 실패임을 나타내는 파라메터입니다. UM3Boolean 타입의 파라메터이며 그 default 값은 FALSE 로 기록되어 전송됩니다.

This parameter indicates the failure of an operation. It is UM3Boolean type, the default value is set to FALSE, and the result is returned.

13.12.7.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.12.7.3. parErrorObjectElementIndex

오류가 발생한 오브젝트가 parUM3ObjectList 파라메터를 구성하는 엘리먼트들 중 몇 번째인가를 나타내는 인덱스 정보입니다. 타입은 UM3UnsignedInteger16 타입으로 정의됩니다.

This contains index information of the index of the object associated with the error among the elements in the parUM3ObjectList parameter. This type is defined with UM3UnsignedInteger16.

13.12.8. FAIL signature OVLD 5

실패 signature OVLD 5 는 오브젝트의 리스트 혹은 오브젝트의 리스트와 조건문으로 주어진 파라메터를 이용하여 오브젝트 그룹을 가져오고자 할 때, 특정 오브젝트의 특정 애트리뷰트에서 오류가 발생할 경우, 해당 오류발생 오브젝트와 애트리뷰트의 정보를 리턴하기 위해 사용하는 signature 입니다.

FAIL signature OVLD 5 is used to return the information of the object and attribute associated with the error if an error occurs in a specific attribute of a specific object when reading an object group using the parameters given as the list of an object or list of objects and conditions.

실패 signature OVLD 5 는 13.12.3 절의 요청 signature OVLD 1 과 13.12.4 절의 요청 signature OVLD 2 에 대응하는 실패 signature 입니다.

FAIL signature OVLD 5 corresponds to the REQUEST signature OVLD 1 in Section 13.12.3 and the REQUEST signature OVLD 2 in Section 13.12.4.

특정 조건문을 적용하여 오브젝트를 찾는 동안 특정 오브젝트에서 오류가 발생할 경우에는 다음 13.12.9 절에서 정의하는 실패 signature OVLD 4 를 이용하여 그 결과를 리턴합니다.

If an error occurs in a specific object while searching for an object using specific conditions, then the result is returned using the FAIL signature OVLD 4 defined in Section 13.12.9.

즉 해당 실패 signature OVLD 3 은 애트리뷰트와 관련된 오류가 발생하였을 때 해당 애트리뷰트와 해당 애트리뷰트가 속한 오브젝트의 정보를 리턴하기 위한 실패 signature 입니다.

In other words, this FAIL signature OVLD 3 is used to return the information of the attributes and the object to which the attributes belong when an error related to the attribute occurs.

13.12.8.1. parResult : FALSE

오퍼레이션의 수행 결과가 실패임을 나타내는 파라메터입니다. UM3Boolean 타입의 파라메터이며 그 default 값은 FALSE 로 기록되어 전송됩니다.

This parameter indicates the failure of an operation. It is UM3Boolean type, the default value is set to FALSE, and the result is returned.

13.12.8.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.12.8.3. parErrorObjectElementIndex

오류가 발생한 애트리뷰트가 어떤 오브젝트에 속해있는 가를 나타내기 위한 정보입니다. 즉, 오브젝트가 parUM3ObjectList 파라메터를 구성하는 엘리먼트들 중 몇 번째인가를 나타내는 인덱스 정보입니다. 타입은 UM3UnsignedInteger16 타입으로 정의됩니다.

This contains index information of the index of the object associated with the error among the elements in the parUM3ObjectList parameter. This type is defined with UM3UnsignedInteger16.

13.12.8.4. parUM3AttributeClassIdentifier

오류가 발생한 애트리뷰트의 클래스 아이디를 나타내며, 그 클래스 타입은 UM3ClassIdentifier 입니다.

This contains the class of the attribute associated with the error, and the class type is UM3ClassIdentifier.

13.12.8.5. parUM3AttributeObjectName

오류가 발생한 애트리뷰트의 오브젝트 이름을 나타내며, 그 클래스 타입은 UM3ObjectName 입니다.

This contains the object name of the attribute that caused an error, and the class type is UM3ObjectName.

13.12.9. FAIL signature OVLD 7

해당 실패 signature OVLD 7 은 요청 signature 의 파라메터가 조건문으로만 이루어진 경우, 오퍼레이션의 수행 도중 오브젝트에서 오류가 발생하였을 때 해당 결과를 리턴하기 위한 실패 signature 입니다. 즉, 요청 signature 에는 오브젝트 리스트에 관한 정보가 없으며 오직 조건문 만이 주어지므로 해당 오퍼레이션의 대상은 에이전트가 관리하는 모든 오브젝트가 그 대상이 됩니다. 해당 오퍼레이션을 수행 도중 오브젝트와 관련된 오류가 발생할 경우에는 해당 오브젝트의 리스트가 요청 signature 의 파라메터로 주어지지 않았으므로, 오브젝트의 클래스아이디와 오브젝트 이름을 모두 리턴해야 합니다.

FAIL signature OVLD 7 is used to return a result if an error occurs in an object during operation and if the parameter of REQUEST signature contains only conditional statements. Because only the conditional statements are provided without the information of the object list in the REQUEST signature, all objects that are managed by the agent are the target of the corresponding operation. If an error related to an object occurs during the corresponding operation, then all of class IDs and object names of the object should be returned because the list of corresponding objects is not

provided as a parameter of the REQUEST signature.

13.12.9.1. parResult : FALSE

오퍼레이션의 수행 결과가 실패임을 나타내는 파라메터입니다. UM3Boolean 타입의 파라메터이며 그 default 값은 FALSE 로 기록되어 전송됩니다.

This parameter indicates the failure of an operation. It is UM3Boolean type, the default value is set to FALSE, and the result is returned.

13.12.9.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.12.9.3. parUM3ObjectClassIdentifier

오퍼레이션 수행 도중 오류가 발생한 오브젝트의 클래스 아이디를 나타냅니다.

This contains the class ID of the object that caused an error during operation.

13.12.9.4. parUM3ObjectObjectName

오퍼레이션 수행 도중 오류가 발생한 오브젝트의 오브젝트 이름을 나타냅니다.

This contains the object name of the object that caused an error during operation.

13.12.10. FAIL signature OVLD 8

요청 signature 의 파라메터가 조건문 만으로 주어진 상태에서 오퍼레이션의 수행 도중 특정 오브젝트의 특정 애트리뷰트에 의한 에러가 발생하는 경우에 사용하는 실패 signature 입니다.

This FAIL signature is used if an error occurs by a specific attribute of a specific object during operation when only conditional statements are provided as a parameter of the REQUEST signature.

13.12.10.1. parResult : FALSE

오퍼레이션의 수행 결과가 실패임을 나타내는 파라메터입니다. UM3Boolean 타입의 파라메터이며 그 default 값은 FALSE 로 기록되어 전송됩니다.

This parameter indicates the failure of an operation. It is UM3Boolean type, the default value is set to FALSE, and

the result is returned.

13.12.10.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.12.10.3. parUM3ObjectClassIdentifier

오류가 발생한 애트리뷰트가 속한 오브젝트의 클래스 아이디를 나타냅니다.

This contains the class ID of the attribute that caused an error during operation.

13.12.10.4. parUM3ObjectObjectName

오류가 발생한 애트리뷰트가 속한 오브젝트의 오브젝트 이름을 나타냅니다.

This contains the object name of the object that has the attribute that caused an error during operation.

13.12.10.5. parUM3AttributeClassIdentifier

오류가 발생한 애트리뷰트의 클래스 아이디를 나타냅니다.

This contains the class ID of the attribute that caused an error.

13.12.10.6. parUM3AttributeObjectName

오류가 발생한 애트리뷰트의 오브젝트 이름 즉, 애트리뷰트 이름을 나타냅니다.

This contains the object name of the attribute i.e. attribute name associated with the error.

13.12.11. Handling Procedure at the Receiver

GetObjectGroupValue 오퍼레이션 APDU 를 수신한 수신 측의 에이전트는 파라메터로 주어진 조건문에 따라 조건을 적용하여 해당 오브젝트 그룹의 오브젝트들을 찾아내어 매니저 측으로 송신합니다. GetObjectGroupValue 오퍼레이션이 적용되기 위해서 에이전트는 자신이 관리하는 정보모델 내부의 모든 오브젝트들에 대한 정보를 갖고 있어야 합니다. 이를 구현하는 방법은 index tree, database 등 여러 가지가 있을 수 있으습니다. 단, 본 권고안은 해당 내용의 구현에 관해서는 그 방법, 프로그래밍 언어의 종류 등에 제한을 두지 않습니다.

When the agent receives the GetObjectGroupValue operation APDU, it searches for the objects of the corresponding object group by using the conditions in the conditional statements provided as parameters, and sends the results to the manager. To apply the GetObjectGroupValue operation, the agent should have information of all objects in the information model managed by the agent. There are many way of implementation, including index tree and database. This recommendation does not put any restriction on the method or type of programming languages in terms of the implementation.

본 권고안이 정의하는 정보모델의 Gateway 및 Server 등 하드웨어 장치를 모델링한 클래스에는 maxAPDULength 등과 같은 한 번에 주고 받을 수 있는 UM3 APDU 의 최대 길이를 나타내는 애트리뷰트가 있습니다. 즉, 한 번에 통신으로 송신하거나 수신할 수 있는 패킷의 최대길이를 나타내는 정보를 갖고 있습니다. 상기 GetObjectGroupValue 오퍼레이션의 경우 해당 그룹에 속해 있는 모든 오브젝트들의 애트리뷰트 값들도 함께 송신 되므로 그 크기가 매우 클 수 있으며, 경우에 따라서는 한 번에 주고 받을 수 있는 최대 패킷의 크기를 넘어서는 경우도 있을 수 있습니다.

Classes that model the hardware devices, such as gateway and server of the information model defined in this recommendation, have the attribute that contains the maximum length of UM3 APDU that can be exchanged each time, such as maxAPDULength. In case of the GetObjectGroupValue operation, the size could become very big since the attributes of all objects in the corresponding group are sent together, and could sometimes exceed the maximum packet size that can be exchanged each time.

매니저와 에이전트는 최대 송수신 가능 패킷의 크기에 관한 정보를 수시로 확인 할 수 있어야 합니다. 또한 필요할 경우 과 같이 여러 개의 패킷으로 나누어 송신하거나 수신할 수 있는 방법을 갖고 있어야 합니다. 본 권고안이 정의하는 UM3 프로토콜은 앞서 정의한 바와 같이 connection oriented UM3 session 과 connectionless UM3 session 의 두 가지 경우 모두 다음 그림 20-Maximum APDU 크기에 따른 패킷의 분할 전송 [과 같은 형태의 데이터 송수신 절차를 적용해야 합니다.

The manager and the agent should be able to check the information of the maximum size of the packet that can be exchanged frequently. They should also have a method to send or receive large amounts of data by splitting them into smaller packets if required. The data exchange procedure shown in Fig. 20 should be applicable to both connection oriented UM3 sessions and connectionless UM3 sessions as defined earlier in the UM3 protocol in this recommendation.

즉, 매니저가 에이전트에 데이터를 요청하는 경우 반드시 maxAPDU 와 같은 에이전트가 한 번에 송수신할 수 있는 데이터의 크기를 확인합니다. 다음으로 에이전트로부터 수신한 데이터를 확인하여 L 필드의 값이 현재 수신한 데이터의 길이보다 더 큰 경우, 에이전트는 다음 번 APDU 를 송신할 단계에 들어가 있다는 뜻입니다. 따라서, 매니저는 바로 연결을 끊지 않고 에이전트가 다음 패킷을 전송할 때까지 기다려야 합니다. 즉, 매니저와 에이전트는 항상 수신한 데이터의 L 필드를 확인하고 해당 필드의 값이 현재 수신한 값보다도 더 클 경우 추가적인 데이터의 수신 작업을 진행해야 함을 뜻합니다.

When the manager requests data from the agent, it should check the size of data that can be transferred each time, like maxAPDU. It checks the data received from the agent, and if the value of L field is greater than the length of the received data, then it means that the agent is in the stage of sending the next APDU. Therefore, the manager should not close the connection: it should wait for the next packet from the agent. The manager and the agent should always check the L field of the received data, and additional data transfers should be performed if the value is greater than the size of the received data.

GetObjectGroupValue 와 같은 오퍼레이션의 경우, 해당 오퍼레이션의 요청 signature 들 중 오브젝트의 리스트와 조건문이 함께 주어지는 경우, 오브젝트 리스트의 엘리먼트를 오직 1 개의 오브젝트로 제한하여 사용할 수도 있습니다. 즉, parObjectList 파라메터의 엘리먼트를 1 개로 지정할 경우, 해당 오퍼레이션은 GetObjectValue 오퍼레이션에 조건 파라메터를 함께 넘기는 것과 동일한 수행 결과를 얻을 수 있게 됩니다.

In case of operation like GetObjectGroupValue, if both the object list and conditional statements are provided by the REQUEST signature, then the element of the object list can be limited to only one object. When the number of elements of the parObjectList parameter is set to 1, the same result can be achieved as in the case of passing condition parameters to the GetObjectValue operation.

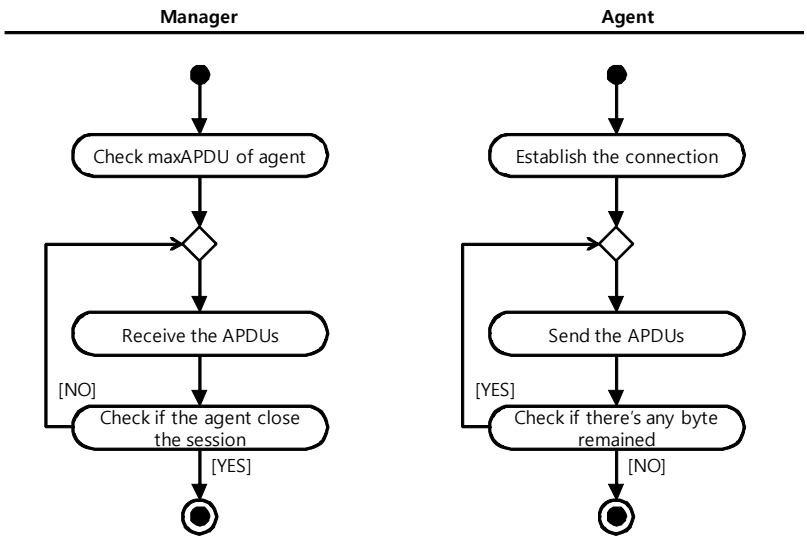


그림 20-Maximum APDU 크기에 따른 패킷의 분할 전송
[Split packet transfer due to the size limitation of Maximum APDU]

13.13. GetObjectAttributeGroupValue operation

본 권고안이 정의하는 GetObjectAttributeGroupValue 오퍼레이션은 한 개의 오브젝트에 속한 여러 개의 애트리뷰트들 중 복수개의 애트리뷰트 값들을 한 번에 가져오기 위해 사용합니다. 즉, 여러 개의 오브젝트에 속한 복수의 애트리뷰트의 값들을 가져오기 위한 오퍼레이션이 아님에 유의해야 합니다.

The GetObjectAttributeGroupValue operation defined in this recommendation is used to get the values of multiple attributes of one object in one operation. Caution is required in that it is not the operation to obtain the values of multiple attributes of multiple objects.

13.13.1. Structure of Signature and Return Type

GetObjectAttributeGroupValue 오퍼레이션의 signature 와 리턴 타입은 다음과 같습니다.

The signature and return type of the GetObjectAttributeGroupValue operation are as follows.

표 54-GetObjectAttributeGroupValue 오퍼레이션의 signature와 리턴 타입
[Signature and return type of the GetObjectAttributeGroupValue operation]

Classification	Type of Parameter	Name and Value of Parameter	Characteristics
REQUEST	UM3ClassIdentifier UM3ObjectName UM3ObjectList	parUM3ClassIdentifier parUM3ObjectName parUM3AttributeObjectList	 UM3ObjectIndicator element
SUCCESS	UM3Boolean UM3ObjectList	parResult: TRUE parUM3ObjectList	OVLD 2
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult: FALSE parUM3OperationErrorCode parErrorElementIndex	OVLD 4

GetObjectAttributeGroupValue 오퍼레이션에는 overloaded 오퍼레이션이 정의되어 있지 않습니다.

다음은 GetObjectAttributeGroupValue 오퍼레이션의 ASN.1 정규표현식으로의 표현입니다.

Overloaded operation is not defined in the GetObjectAttributeGroupValue operation.

The GetObjectAttributeGroupValue operation can be defined by using ASN.1 regular expressions as follows.

```
GetObjectAttributeGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier              UM3ClassIdentifier,
        &parUM3ObjectName                   UM3ObjectName,
        &parUM3AttributeObjectList          UM3ObjectList
}
```

상기 ASN.1 정규표현식으로 정의한 바와 같이 parUM3AttributeObjectList 파라메터는 UM3ObjectList 타입의 파라메터입니다. 즉, 선택하고자 하는 애트리뷰트의 정보를 나타내게 됩니다.

As defined with ASN.1 regular expressions, the parUM3AttributeObjectList parameter is UM3ObjectList type, and it contains the information of the attribute to select.

13.13.2. REQUEST signature

13.13.2.1. parUM3ClassIdentifier

가져오고자 하는 복수개의 애트리뷰트가 속한 오브젝트의 클래스 아이디를 나타냅니다.

This contains the class ID of the object that has multiple attributes to read.

13.13.2.2. parUM3ObjectName

가져오고자 하는 복수의 애트리뷰트가 속한 오브젝트의 오브젝트 이름을 나타냅니다.

This contains the object name of the object that has multiple attributes to read.

13.13.2.3. parUM3AttributeObjectList

UM3ObjectList 타입의 파라메터이며, 가져오고자 하는 애트리뷰트의 클래스아이디와 오브젝트 이름을 나타냅니다. 이 때 UM3ObjectList 타입의 엘리먼트는 UM3ObjectIndicator 타입으로 정의합니다. 이는 단순히 애트리뷰트의 타입과 해당 애트리뷰트의 이름 만을 파라메터로 넘겨 해당 애트리뷰트의 값을 가져오기 위함입니다.

This parameter is UM3ObjectList type, and it contains the class ID and object name of the attribute to read. In this case, the element of the UM3ObjectList type is UM3ObjectIndicator type. The purpose is to get the value of the corresponding attribute by passing the type and name of the corresponding attributes as only a parameter.

13.13.3. SUCCESS signature OVLD 2

13.13.3.1. parResult : TRUE

성공했을 경우 parResult 파라메터의 값이 TRUE 로 지정되어 전송됩니다

When the operation is a success, the value of the parResult parameter is set to TRUE and the result is returned.

13.13.3.2. parUM3ObjectList

하나의 특정 오브젝트가 갖고 있는 1 개 이상의 복수의 애트리뷰트의 값들이 parUM3ObjectList 파라메터에 기록되어 전송됩니다. 이때 UM3ObjectList 타입은 UM3 SEQUENCE OF 타입이며 해당 복합형식 타입을 구성하는 엘리먼트들은 본 권고안이 정의하는 모든 타입의 값들이 기록될 수 있어야 합니다.

Values of more than one attribute of a specific object are stored in the parUM3ObjectList parameter and the result is returned. In this case, the UM3ObjectList type is UM3 SEQUENCE OF type, and the elements of the corresponding complex format type should be able to store values of any type as defined in this recommendation.

13.13.4. FAIL signature OVLD 4

13.13.4.1. parResult : FALSE

실패했을 경우 parResult 파라메터의 값이 FALSE 로 지정되어 전송됩니다.

When the operation fails, the value of parResult parameter is set to FALSE and the result is returned.

13.13.4.2. parUM3OperationErrorCode

실패의 원인을 나타내는 코드를 기록하며 그 타입은 UM3OperationErrorCode 입니다.

This contains the code that indicates the cause of failure, and the type is UM3OperationErrorCode.

13.13.4.3. parErrorElementIndex

해당 오퍼레이션의 요청 signature 에는 UM3ObjectList 타입의 parUM3AttributeObjectList 파라메터가 주어집니다. 즉, UM3 SEQUENCE OF 타입의 파라메터 값이 주어지므로 파라메터의 엘리먼트의 순서에 의미가 있는 경우입니다. 즉, UM3UnsignedInteger16 타입으로 지정된 parErrorElementIndex 값을 참조할 경우 parUM3AttributeObjectList 의 몇 번째 엘리먼트에서 오류가 발생 했는지를 확인할 수 있게 됩니다. 단, parErrorElementIndex 파라메터가 나타내는 오류는 오퍼레이션의 처리 과정 중 처음으로 오류가 발생한 엘리먼트의 인덱스로 정의합니다. 즉, GetObjectAttributeGroupValue 오퍼레이션과 복수의 오브젝트 혹은 애트리뷰트 오브젝트를 다루는 다른 모든 UM3 오퍼레이션들은 맨 처음으로 오류가 발생한 오브젝트 혹은 애트리뷰트 오브젝트에 대한 인덱스 정보를 parErrorElementIndex 에 저장하는 것으로 정의합니다

The parUM3AttributeObjectList parameter of UM3ObjectList type is given to the REQUEST signature of the corresponding operation. Since the parameter of UM3 SEQUENCE OF type is provided, it means that the sequence of elements in the parameter has significant meaning. For example, it can be used to check which element in the parUM3AttributeObjectList is associated with the error by using the parErrorElementIndex value defined with UM3UnsignedInteger16 type. The error indicated by the parErrorElementIndex parameter though the index of the element is the first error that occurred during the operation. All other UM3 operations handling the GetObjectAttributeGroupValue operation or multiple objects or attributes of an object has the index information of the object or attribute object that is associated with the first error that occurred in the parErrorElementIndex by definition.

13.13.5. Handling Procedure at the Receiver

앞서 정의한 바와 같이 UM3 프로토콜은 복수의 오브젝트들의 모든 애트리뷰트를 가져올 필요가 있을 경우, 이를 GetObjectGroupValue 오퍼레이션을 통해 처리할 수 있습니다. 그러나, 복수개의 오브젝트들을 대상으로 해당 오브젝트 별로 상이한 종류의 복수의 애트리뷰트들을 가져오는 오퍼레이션은 별도로 정의되어 있지 않습니다. 따라서, 이러한 기능이 필요할 경우에는 모든 오브젝트를 대상으로 개별적인 GetObjectAttributeGroupValue 오퍼레이션을 이용하여 필요한 애트리뷰트의 값들을 가져와야 합니다.

As defined earlier, the GetObjectGroupValue operation can be used to get all attributes of multiple objects in the UM3 protocol. The operation to read multiple attributes of different types for each object for multiple objects, however, is not defined separately. In this case, the values of the required attributes should be obtained by using an individual GetObjectAttributeGroupValue operation for all objects if this function is required.

에이전트는 매니저가 송신한 GetObjectAttributeGroupValue 오퍼레이션의 파라메터 parUM3ClassIdentifier와 parUM3ObjectName 파라메터가 가리키는 오브젝트를 찾습니다. 해당 오브젝트를 찾았을 경우 parUM3AttributeObjectList 파라메터가 가리키는 애트리뷰트들의 값을 모두 읽어 매니저에게 전송하는 과정을 거치게 됩니다. 주어진 파라메터의 오브젝트를 찾는데 실패했을 경우, 혹은 해당 애트리뷰트를 찾는데 실패했을 경우, parUM3OperationErrorCode 파라메터를 이용해 실패 원인을 기록하고 이를 매니저로 전송합니다. 혹은 parErrorElementIndex 파라메터를 이용하여 오류가 발생한 애트리뷰트 오브젝트의 정확한 정보를 리턴 할 수도 있습니다.

The agent searches for the object pointed by the parameter parUM3ClassIdentifier and parUM3ObjectName parameter of the GetObjectAttributeGroupValue operation provided by the manager. When it is found, it reads all the values of attributes pointed by the parUM3AttributeObjectList parameter and sends the data to the manager. If it fails to find the object or corresponding attribute, then it stores the cause of failure by using the parUM3OperationErrorCode parameter and sends it to the manager. Otherwise, accurate information about the attribute object that caused the error can be returned by using the parErrorElementIndex parameter.

13.14. SetObjectValue operation

SetObjectValue 오퍼레이션은 특정 오브젝트에 대해, 파라메터로 주어지는 오브젝트에 명기되어 포함된 애트리뷰트의 값을 에이전트가 관리하는 해당 오브젝트의 애트리뷰트 값으로 설정하기 위해 사용됩니다. 즉, GetObjectValue 오퍼레이션은 해당 오브젝트의 모든 애트리뷰트에 대한 오퍼레이션이지만, SetObjectValue 오퍼레이션은 그 값을 설정하기 위한 애트리뷰트를 파라메터에 포함된 애트리뷰트로 국한하여 오퍼레이션이 수행되어야 합니다.

The SetObjectValue operation is used to set the value of the attribute included in the object provided as a parameter of a specific object with the value of the attribute of the corresponding object managed by the agent. In other words, the GetObjectValue operation is the operation about all attributes of the corresponding object, but the SetObjectValue operation for writing the value should be limited to the attributes included in the parameter.

해당 오퍼레이션을 수행하기 전에 매니저는 반드시 에이전트가 해당 오브젝트를 갖고 있는지를 확인해야 하며, 또한 해당 오브젝트의 애트리뷰트 리스트를 조사하여 어떤 애트리뷰트들이 해당 오브젝트에 속해 있는지를 명확하게 확인 한 후 해당 오퍼레이션을 수행해야 합니다.

The manager should check whether the agent has the corresponding object before running the corresponding operation. It should check the type of attributes of the corresponding object by clearly searching the attribute list before running the corresponding operation.

GetObjectValue 오퍼레이션과는 달리 SetObjectValue 오퍼레이션은 특정 애트리뷰트가 존재하지 않을 경우 해당 애트리뷰트의 값을 설정할 수 없으며, 이는 경우에 따라 심각한 오류를 발생시키는 원인으로 작용할 수도 있습니다. 따라서, 매니저는 반드시 에이전트가 관리하는 정보모델의 특정 오브젝트와 해당 오브젝트의 애트리뷰트의 리스트에 관한 정보를 정확하게 유지 관리하고 있어야 합니다.

Unlike the GetObjectValue operation, the SetObjectValue operation cannot set the value of an attribute if it does not exist: this could cause fatal error. Therefore, the manager should maintain accurate information about the list of a specific object and attributes of the corresponding object of the information model managed by the agent.

13.14.1. Structure of the Signature and Return Type

SetObjectValue 오퍼레이션의 파라메터와 리턴 타입은 다음과 같습니다.

The parameters and return type of the SetObjectValue operation are as follows.

Table 55 – Structure of signature and return type of SetObjectValue operation [SetObjectValue 오퍼레이션의 signature 와 리턴 타입의 구조]

Classification	Type of Parameter	Name and Value of Parameter	Characteristics
REQUEST	UM3ClassIdentifier UM3ObjectName ANY DEFINED BY UM3 ASN.1 module	parUM3ClassIdentifier parUM3ObjectName parAnyTypeValue	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3
FAIL	UM3Boolean UM3OperationErrorCode	parResult : FALSE parUM3OperationErrorCode	OVLD 4

	UM3UnsignedInteger16	parErrorElementIndex

상기 요청 signature 와 성공 및 실패 signature 에서 보는 바와 같이 SetObjectValue 오퍼레이션은 GetObjectValue 오퍼레이션과는 반대로 요청 signature 에 설정에 필요한 값을 넘겨줍니다. OperationResponse 오퍼레이션은 SetObjectValue 오퍼레이션이 성공하였을 경우 단순히 성공여부만을 파라메터로 넘겨줍니다.

다음은 상기 SetObjectValue 오퍼레이션의 ASN.1 정규표현식으로의 정의입니다.

As shown in the REQUEST, SUCCESS, and FAIL signature in the table, the SetObjectValue operation passes the values required for settings to the REQUEST signature as opposed to the GetObjectValue operation. The OperationResponse operation simply passes the success indicator as a parameter when the SetObjectValue operation is a success.

The SetObjectValue operation can be defined as ASN.1 regular expressions as follows.

```
SetObjectValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier            UM3ClassIdentifier,
        &parUM3ObjectName                 UM3ObjectName,
        &parAnyTypeValue                  ANY DEFINED BY UM3 ASN.1 module
}
```

ANY DEFINED BY UM3 ASN.1 module 은 본 권고안에서 정의한 모든 타입의 클래스들이 파라메터로 정의될 수 있다는 뜻입니다.

ANY DEFINED BY UM3 ASN.1 module means that any type of class defined in this recommendation can be defined as parameter.

13.14.2. REQUEST signature

SetObjectValue 오퍼레이션의 signature 는 설정하고자 하는 오브젝트의 클래스 아이디, 오브젝트 이름 및 해당 오브젝트를 구성하는 애트리뷰트의 값들로 구성됩니다.

The signature of the SetObjectValue operation has the class ID and object name of the object to set and the values of the attributes to configure the corresponding object.

13.14.2.1. parUM3ClassIdentifier

설정하고자 하는 오브젝트의 클래스 아이디를 나타냅니다.

This contains the class ID of the object to set.

13.14.2.2. parUM3ObjectName

설정하고자 하는 오브젝트의 오브젝트 이름을 나타냅니다.

This contains the object name of the object to set.

13.14.2.3. parAnyTypeValue

설정하고자 하는 오브젝트의 인스턴스를 나타냅니다. 즉, 해당 오브젝트가 갖고 있는 모든 애트리뷰트의 값들이 설정된 오브젝트의 값을 나타냅니다. 이 때 parAnyTypeValue 파라메터는 본 권고안이 정의하는 모든 클래스 타입이 가능하며, 해당 오브젝트의 애트리뷰트 값들은 정상적인 형태로 그 값이 설정되어 있어야 합니다. 즉, DEFAULT 값이 설정되어 있지 않은 경우를 제외하고는 모든 애트리뷰트의 값들이 초기화된 상태로 넘겨져야 합니다.

This is the instance of the object to set. In other words, it contains the value of the object with all values of the attributes that are set. Any class type defined in this recommendation can be used for the parAnyTypeValue parameter, and the values of the attributes of the corresponding object should be set in normal format. The values of all attributes except the ones without default values should be initialized when they are passed.

13.14.3. SUCCESS signature

13.14.3.1. parResult : TRUE

SetObjectValue 오퍼레이션이 성공적으로 수행되었을 경우 UM3Boolean 타입의 parResult 파라메터에는 TRUE 값이 기록되어 전송됩니다.

When the SetObjectValue operation is a success, the parResult parameter of the UM3Boolean type should be set to TRUE and the result is returned.

13.14.4. FAIL signature OVLD 3

13.14.4.1. parResult : FALSE

SetObjectValue 오퍼레이션의 수행이 실패했을 경우, UM3Booleean 타입의 parResult 파라메터에는

FALSE 값이 기록되어 전송됩니다. 또한, 실패의 원인은 parUM3OperationErrorCode 에 설정되어 전송됩니다.

When the SetObjectValue operation fails, the parResult parameter of UM3Booleean type is set to FALSE. The cause of failure is stored in the parUM3OperationErrorCode, and the result is returned.

13.14.4.2. parUM3OperationErrorCode

parUM3OperationResultCode 에는 SetObjectValue 오퍼레이션이 실패한 이유가 기록되어 전송됩니다.

The cause of failure of the SetObjectValue operation is stored in the parUM3OperationResultCode, and the result is returned.

13.14.5. FAIL signature OVLD 4

13.14.5.1. parResult : FALSE

SetObjectValue 오퍼레이션의 수행이 실패했을 경우, UM3Booleean 타입의 parResult 파라메터에는 FALSE 값이 기록되어 전송됩니다. 또한, 실패의 원인은 parUM3OperationErrorCode 에 설정되어 전송됩니다.

When the SetObjectValue operation fails, the ParResult parameter of UM3Booleean type is set to FALSE, and the result is returned. The cause of failure is stored in the parUM3OperationErrorCode, and the result is returned.

13.14.5.2. parUM3OperationErrorCode

parUM3OperationResultCode 에는 SetObjectValue 오퍼레이션이 실패한 이유가 기록되어 전송됩니다.

The cause of failure of the SetObjectValue operation is stored in the parUM3OperationResultCode, and the result is returned.

13.14.5.3. parErrorElementIndex

상기 parAnyTypeValue 에는 완전한 형태의 UM3 오브젝트가 애트리뷰트의 값들을 갖고 기록되어 있습니다. 이는 parAnyTypeValue 에 기록된 오브젝트의 애트리뷰트의 기록 순서를 매니저가 관리할 수 있다는 얘기이며 해당 애트리뷰트의 순서에 따라 오퍼레이션을 수행 하는 도중 에러가 발생할 경우 해당 애트리뷰트의 순서에 관한 정보를 알려줌으로써 어떤 애트리뷰트에서 오류가 발생했는지를 확인할 수 있게 됩니다.

The parAnyTypeValue contains the complete form of UM3 the object with the values of attributes. This means that

the manager can manage the sequence of the storage of attributes of the object stored in the parAnyTypeValue. When an error occurs during operation, the attribute associated with the error can be identified with the information about the sequence of the corresponding attribute.

13.14.6. Handling Procedure at the Receiver

SetObjectValue 오퍼레이션 APDU 를 수신한 에이전트는 parUM3ClassIdentifer 와 parUM3ObjectName 을 이용하여 자신이 관리하는 정보모델의 오브젝트들로부터 해당 오브젝트를 찾아 냅니다. 해당 오브젝트를 찾은 후에는 parAnyTypeValue 파라메터로 주어진 해당 오브젝트의 각 애트리뷰트의 값들을 모두 자신이 관리하는 오브젝트의 애트리뷰트의 값들로 새롭게 설정한 후 오퍼레이션의 수행을 종료합니다.

When an agent receives the SetObjectValue operation APDU, it searches for the object from the objects of information model managed by it, using the parUM3ClassIdentifier and parUM3ObjectName. After the corresponding object is found, the values of each attribute of the corresponding object provided as parAnyTypeValue parameters are newly set by the values of the attributes of the object managed by itself, and then the operation is terminated.

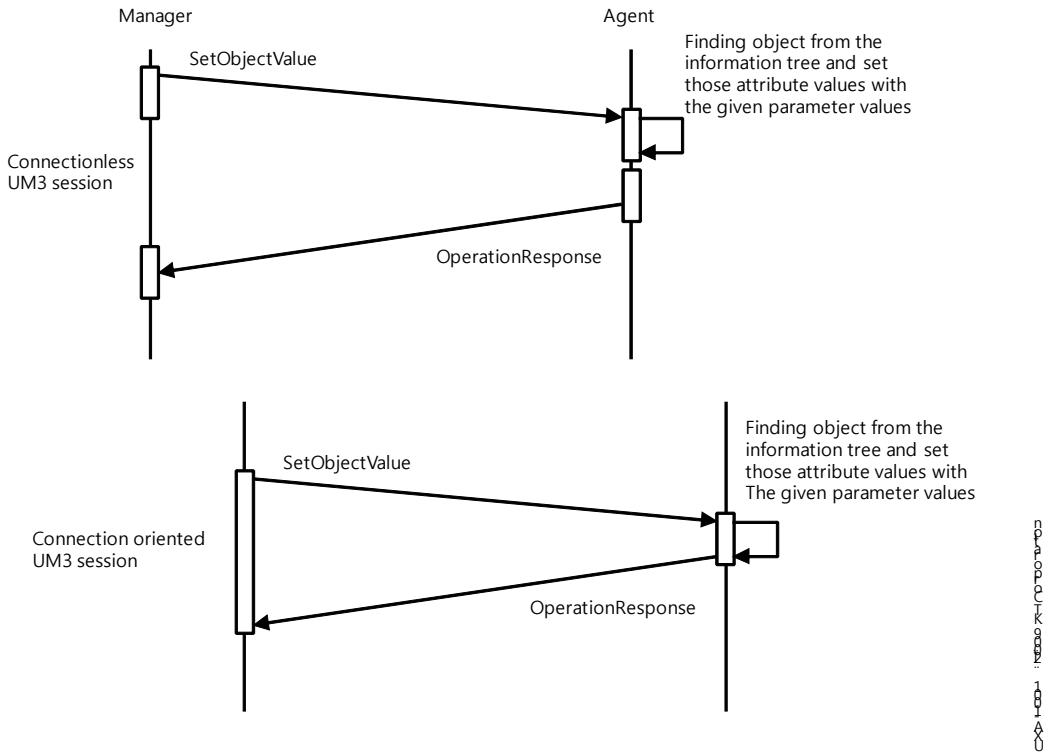


그림 21-SetObjectValue 오퍼레이션의 시퀀스다이어그램 [Sequence diagram of the SetObjectValue operation]

이 때 파라메터로 주어지는 오브젝트에는 선택적인 애트리뷰트와 그 값이 설정되어 있습니다. 즉, 해당 오브젝트의 모든 애트리뷰트 값이 기록되어 있는 것이 아니라, 해당 오브젝트의 애트리뷰트들 중 일부만이 기록되어 있을 수도 있습니다. .

The object provided as parameter in this case has the optional attributes and their values. Not all values of the attributes of the corresponding object are stored, and only some of the attributes may have the values.

앞서 정의한 모든 UM3 오퍼레이션들과 마찬가지로 SetObjectValue 오퍼레이션에 대한 응답 또한 해당 오퍼레이션이 요청하는 설정작업을 완전히 종료한 후에 전송합니다.

Like other UM3 operations defined earlier, the response to the SetObjectValue operation is sent after completely finishing the settings requested by the corresponding operation.

그림 21-SetObjectValue 오퍼레이션의 시퀀스다이어그램 [은 SetObjectValue 오퍼레이션과 관련된 매니저와 에이전트 간의 통신 시퀀스다이어그램을 나타냅니다. 그림 21-SetObjectValue 오퍼레이션의 시퀀스다이어그램 [에서 보는 바와 같이 UM3 오퍼레이션에 대한 응답 OperationResponse 오퍼레이션은 매니저의 데이터 수신 후 해당 연결을 끊고 별도의 세션을 통해 매니저로 응답을 보내는 경우와, 해당 세션을 유지한 채 그대로 응답을 보내는 경우가 있습니다. 전자의 경우 즉, connectionless UM3 session 은 해당 오퍼레이션의 수행에 대단히 긴 시간이 소요되고 이를 처리하는 동안 다른 오퍼레이션의 수신이 불가능한 경우에 활용할 수 있는 방식입니다.

Fig. 21 shows the communication sequence diagram between the manager and agent related to the SetObjectValue operation. As shown in Fig. 21, there are two types of OperationResponse operation for the response to UM3 operation. In one type, the connection is closed after receiving data from the manager and a response is sent 솟to the manager through a separate session. In the other type, the response is sent while maintaining the session. The first type, a connectionless UM3 session, can be applied in situations in which it takes a long time to run the corresponding operation and other operations cannot be received while processing the current operation.

단, 본 권고안은 두 가지 방식의 세션 설정 중 connection oriented UM3 session 을 사용토록 정의합니다.

In this recommendation, a connection oriented UM3 session should be used among the above two methods of session settings.

13.15. SetObjectAttributeValue operation

SetObjectAttributeValue 오퍼레이션은 특정 오브젝트의 특정 애트리뷰트의 값을 설정하기 위해 사용됩니다. 즉, 단일 애트리뷰트의 값을 설정하기 위해 사용되는 오퍼레이션입니다. 해당 오퍼레이션은 GetObjectAttributeValue 오퍼레이션과 마찬가지로 본 권고안이 정의하는 오퍼레이션들 중 가장 활용 빈도가 높은 오퍼레이션들 중 하나입니다.

The SetObjectAttributeValue operation is used to set the values of a specific attribute of a specific object. In other words, it is used to set the values of a single attribute. Like the GetObjectAttributeValue operation, it is one of the most frequently used operations among the ones defined in this recommendation.

13.15.1. Structure of Signature and Return Type

SetObjectAttributeValue 오퍼레이션의 signature 와 리턴 타입의 구조는 표 56-SetObjectAttributeValue 오퍼레이션의 signature 와 리턴 타입의 구조 와 같습니다.

The structure of Signature and Return Type of the SetObjectAttributeValue operation are shown in Table 55,

표 56-SetObjectAttributeValue 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of SetObjectAttributeValue operation]

Classification	Type of Parameter	Name and Value of Parameter	Characteristics
REQUEST	UM3ClassIdentifier UM3ObjectName UM3AttributeClassIdentifier UM3AttributeObjectName ANY DEFINED BY UM3 ASN.1 module	parUM3ClassIdentifier parUM3ObjectName parUM3AttributeClassIdentifier parUM3AttributeObjectName parAnyTypeValue	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3

표 56-SetObjectAttributeValue 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 SetObjectAttributeValue 오퍼레이션은 그 값을 설정하고자 하는 애트리뷰트가 속한 오브젝트의 클래스 아이디, 오브젝트 이름 및 해당 애트리뷰트의 클래스 아이디와 애트리뷰트의 이름 등이 파라메터로 주어지게 됩니다.

다음은 SetObjectAttributeValue 오퍼레이션을 ASN.1 정규표현식으로 나타낸 결과입니다.

As shown in Table 55, the class ID and object name of the object that has the attribute to set the values and the class IDs and the names of the attributes of the corresponding attribute are provided as parameters.

The SetObjectAttributeValue operation can be defined as ASN.1 regular expressions.

```
SetObjectAttributeValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier              UM3ClassIdentifier,
        &parUM3ObjectName                   UM3ObjectName,
        &parUM3AttributeClassIdentifier     UM3ClassIdentifier,
        &parUM3AttributeObjectName          UM3ObjectName,
        &parAnyTypeValue                    ANY DEFINED BY UM3 ASN.1 module
}
```

13.15.2. REQUEST signature

다른 오퍼레이션들과 마찬가지로 오퍼레이션의 인코딩에 있어서 각 파라메터의 순서는 반드시 본 권고안에 정의된 파라메터의 순서에 따라 인코딩이 이루어져야 합니다.

SetObjectAttributeValue 오퍼레이션은 parUM3ClassIdentifier, parUM3ObjectName 등 오브젝트를 지정하는 파라메터와, parUM3AttributeClassIdentifier, parUM3AttributeObjectName 등의 애트리뷰트를 지정하는 파라메터, 마지막으로 parAnyTypeValue 의 실제 애트리뷰트의 값을 전달하는 파라메터 등으로 구성되어 있습니다.

Like other operations, the parameters should be encoded according to the sequence defined in this recommendation.

The SetObjectAttributeValue operation includes the parameter to assign an object such as parUM3ClassIdentifier and parUM3ObjectName, parameter to assign attribute such as parUM3AttributeClassIdentifier and parUM3AttributeObjectName, and lastly parameters for sending the actual attribute values of parAnyTypeValue.

13.15.2.1. parUM3ClassIdentifier

그 값을 설정하고자 하는 애트리뷰트가 속한 특정 오브젝트의 클래스 아이디를 나타내는 파라메터 입니다.

This parameter contains the class ID of a specific object that has the attribute to set the value.

13.15.2.2. parUM3ObjectName

그 값을 설정하고자 하는 애트리뷰트가 속한 특정 오브젝트의 오브젝트 이름을 나타내는 파라메터 입니다.

This parameter contains the object name of a specific object that has the attribute to set the value.

13.15.2.3. parUM3AttributeClassIdentifier

그 값을 설정하고자 하는 애트리뷰트의 클래스 아이디를 나타냅니다. 해당 애트리뷰트의 클래스 아이디 값은 반드시 그 값을 설정하고자 하는 애트리뷰트의 클래스 아이디 값과 비교 후 연산을 진행해야 하며, 해당 애트리뷰트의 클래스 아이디 값과 상이할 경우 오퍼레이션은 실패 한 것으로 간주되어야 합니다.

This contains the class ID of the attribute to set the value. The class ID of the attribute should be compared with the class ID of the attribute to set the value, and the operation is considered as a failure if the class ID of the corresponding ID is different.

13.15.2.4. parUM3AttributeObjectName

그 값을 설정하고자 하는 애트리뷰트의 애트리뷰트 이름을 나타냅니다.

This contains the attribute name of the attribute to set the value.

13.15.2.5. parAnyTypeValue

그 값을 설정하고자 하는 애트리뷰트의 설정값을 나타냅니다. parAnyTypeValue 는 ANY DEFINED BY UM3 ASN.1 module 로 지정되어 있습니다. 이는 다른 경우와 마찬가지로 본 권고안에서 정의한 모든 형태의 타입들이 지정될 수 있음을 뜻하며 클래스 타입, 기본 타입과 더불어 복합형식의 타입 등이 모두 가능함을 뜻합니다.

This contains the settings of the attribute to set the value. parAnyTypeValue is defined as the ANY DEFINED BY UM3 ASN.1 module. This means that any type defined in this recommendation can be used like the other cases, and any of the class types, primitive types, and complex format types can be allowed.

13.15.3. SUCCESS signature

SetObjectAttributeValue 오퍼레이션이 성공했을 경우 OperationResponse 오퍼레이션의 파라메터는 parResult 파라메터 만으로 구성됩니다.

When the SetObjectAttributeValue operation is a success, the parameter of the OperationResponse operation contains the parResult parameter only.

13.15.3.1. parResult : TRUE

SetObjectAttributeValue 오퍼레이션이 성공할 경우, OperationResponse 오퍼레이션의 UM3Boolean 타입의 파라메터이며, 그 값은 TRUE 로 설정되어 전송됩니다.

When the SetObjectAttributeValue operation is a success, the value of this parameter, which is UM3Boolean type, of the OperationResponse operation is set to TRUE and the result is returned.

13.15.4. FAIL signature OVLD 3

13.15.4.1. parResult : FALSE

SetObjectAttributeValue 오퍼레이션의 수행 결과가 실패일 경우, OperationResponse 오퍼레이션의 UM3Boolean 타입의 파라메터이며, 그 값은 FALSE 로 기록되게 됩니다.

When the SetObjectAttributeValue operation fails, the value of this parameter, which is UM3Boolean type, is set to

FALSE and the result is returned.

13.15.4.2. parUM3OperationErrorCode

parUM3OperationErrorCode 파라메터에는 상기 오퍼레이션의 실패 원인을 나타내는 코드가 기록됩니다.

parUM3OperationErrorCode parameter contains the code that indicates the cause of failure of operation.

13.15.5. Handling Procedure at the Receiver

에이전트는 매니저가 전송한 SetObjectAttributeValue 오퍼레이션에 대해 해당 오브젝트와 애트리뷰트를 자신이 관리하는 정보모델 트리에서 검색합니다. 해당 애트리뷰트를 찾은 후 에이전트는 해당 애트리뷰트의 값을 parAnyTypeValue 파라메터의 값으로 대체하게 됩니다.

When the agent receives the SetObjectAttributeValue operation from the manager, it searches for the corresponding object and attribute from the information tree managed by itself. When the corresponding attribute is found, the agent replaces the value of the corresponding attribute with the value of the parAnyTypeValue parameter.

만약 위의 parUM3AttributeClassIdentifier 의 값이 특정 애트리뷰트의 클래스 아이디 값과 서로 상이하다면, 해당 SetObjectAttributeValue 오퍼레이션은 실패로 간주되어 parResult 파라메터에 FALSE 값을 기록하여 OperationResponse 오퍼레이션의 APDU 를 전송해야 합니다.

If the class ID of a specific attribute is different from the value of the parUM3AttributeClassIdentifier, then the corresponding SetObjectAttributeValue operation is considered as a failure. The value of parResult parameter is set to FALSE, and APDU of the OperationResponse operation should be returned.

13.16. SetObjectGroupValue operation

SetObjectGroupValue 오퍼레이션은 1 개 이상의 오브젝트들에 대해 오퍼레이션의 파라메터로 주어지는 오브젝트의 애트리뷰트들을 에이전트가 관리하는 해당 오브젝트의 애트리뷰트 값으로 재 설정하는 과정을 수행합니다. 앞서 정의한 GetObjectGroupValue 오퍼레이션과 마찬가지로 오브젝트의 리스트와 조건문을 파라메터로 넘겨주어 특정 오브젝트들에 대한 애트리뷰트 값의 재설정 작업을 수행합니다. 특히 오브젝트 리스트를 넘겨주어 애트리뷰트 값의 재설정이 이루어지는 경우, 에이전트가 관리하는 오브젝트의 애트리뷰트와 매니저가 관리하는 오브젝트의 애트리뷰트가 서로 상이할 경우 오퍼레이션의 수행은 실패하게 됩니다. 따라서, 매니저와 에이전트는 항상 최신의 애트리뷰트 리스트 정보를 서로 동기화 하여 애트리뷰트 종류의 불일치에 따른 오류를 미연에 방지할 수 있어야 합니다.

The SetObjectGroupValue operation is used to reset the attributes of the objects provided as parameters of the

operation for multiple objects with the value of the corresponding object managed by the agent. Like the GetObjectGroupValue operation defined earlier, the list of objects and conditional statements is passed as parameters and values or the attributes of the specific object are reset with them. Especially when the attributes values are set with the list of object, the operation will fail if the attribute of the object managed by the agent is different from the one managed by the manager. Therefore, the manager and the agent should be synchronized with the latest information of the attribute list to prevent errors caused by inconsistencies in attribute types.

13.16.1. Structure of Signature and Return Type

SetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조는 표 57-SetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조[과 같습니다. 표 57-SetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조[에서 보는 바와 같이 SetObjectGroupValue 오퍼레이션의 signature 에는 세 가지 유형이 있습니다. 첫 번째는 재설정하고자 하는 오브젝트와 애트리뷰트의 값들을 리스트 형식으로 넘겨주는 형태입니다. 이 때 parUM3ObjectList 타입을 구성하는 엘리먼트들은 재설정하고자 하는 애트리뷰트의 값들을 모두 갖고 있는 완전한 형태의 오브젝트이어야 합니다. 두 번째는 완전한 형태의 오브젝트들과 더불어 조건문을 함께 전송하는 경우입니다.

The structure of signature and return types of the SetObjectGroupValue operation are shown in Table 56. There are three types of signatures in the SetObjectGroupValue operation as shown in the table. The first type is to pass the values of objects and attributes to set in a list form. In this case, the elements configuring the parUM3ObjectList type should be the complete form of object that has all the values of attributes to set. The second type is to pass the complete form of objects and conditional statements.

위와 같은 경우 조건문의 조건을 만족하는 오브젝트들에 대해서만 재설정 과정이 적용되게 됩니다. 단, 조건문을 적용하는 대상은 에이전트가 관리하고 있는 정보모델의 오브젝트들입니다. 세 번째는 상기 오브젝트 리스트와 조건문 그리고 대상이 되는 그룹의 대상 오브젝트를 지정하는 별도의 리스트를 함께 보내는 경우입니다.

In this case, the reset operation applies only to the objects that meet the conditions in the conditional statements. The objects to apply the conditions are objects of information model managed by the agent. The third type is to send the list of objects, conditional statements, and separate list that contains the objects of target group separately.

이상과 같은 오퍼레이션의 수행 중 발생하는 오류는 그 발생지점을 오브젝트와 애트리뷰트로 나누어 리턴 타입을 구성합니다. SetObjectGroupValue 오퍼레이션은 GetObjectGroupValue 오퍼레이션과는 달리 설정하고자 하는 오브젝트의 값 즉, 해당 오브젝트의 모든 애트리뷰트 값들과, 해당 애트리뷰트로 구성되는 모든 재설정 대상 오브젝트들의 리스트를 파라메터로 전송합니다. 해당 값들이 인코딩 되는 과정은 UM3 SEQUENCE OF 타입의 인코딩 룰과 동일한 형식으로 인코딩 되므로 각 엘리먼트들의 순서에 의미가 있고, 따라서 모든 오브젝트와 애트리뷰트들은 그 순서에 따라 식별이 가능하게 됩니다.

Errors occurring during operation are returned using the return type in such a way so that the error source location is divided to object or attribute. Unlike the GetObjectGroupValue operation, the SetObjectGroupValue operation sends the list of all target objects to set including the values of object to set, all attributes of the corresponding object, and the corresponding object as parameters. Since the encoding process is the same as the encoding rule of the UM3 SEQUENCE OF type, the sequence of each element has significant meaning so that all objects and attributes can be identified by their indexes.

표 57-SetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of SetObjectGroupValue operation]

Classification	Type of Parameter	Name and Value of Parameter	Characteristics
REQUEST	UM3ObjectList UM3ObjectList	parUM3ObjectDestinationList parUM3ObjectSourceList	UM3ObjectIndicator element ANY … module element
REQUEST	UM3ObjectValueCondition UM3ObjectList	parUM3ObjectValueCondition parUM3ObjectSourceList	OVLD 1 ANY … module element
REQUEST	UM3ObjectList UM3ObjectValueCondition UM3ObjectList	parUM3ObjectDestinationList parUM3ObjectValueCondition parUM3ObjectSourceList	OVLD 2, UM3ObjectIndicator element ANY … module element
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult : FALSE parUM3OperationErrorCode parErrorObjectElementIndex	OVLD 4
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16 UM3UnsignedInteger16	parResult : FALSE parUM3OperationErrorCode parErrorObjectElementIndex parErrorAttributeElementIndex	OVLD 9

이상과 같은 이유로 SetObjectGroupValue 오퍼레이션은 실패 signature 를 구성하는 파라메터에 클래스 아이디나 혹은 오브젝트 이름을 나타내는 파라메터를 갖고 있지 않습니다.

Because of the reasons described earlier, the SetObjectGroupValue operation does not have a class ID or object name in the parameter that configures the FAIL signature.

실패 signature 를 나타내는 구성은 오브젝트에서 오류가 발생했을 경우 parErrorObjectElementIndex 파라메터에 기록하며, 애트리뷰트에서 오류가 발생했을 경우에는 parErrorObjectElementIndex 파라메터와 parErrorAttributeElementIndex 파라메터를 이용해 그 위치를 기록합니다. 따라서, 실패 signature 의 리턴 타입은 3 가지 유형이 존재하게 됩니다. 즉, 첫 번째로 오류의 발생 지점이 오브젝트나 애트리뷰트와

관계가 없는 경우, 혹은 오류발생 지점에 대한 정보를 생략하는 경우로 오퍼레이션 에러 코드만 리턴하는 경우입니다. 두 번째는 오류의 발생 지점이 오브젝트인 경우이며, 마지막으로는 오류의 발생 지점이 애트리뷰트인 경우입니다.

With this configuration of FAIL signature, the index of the object that caused an error is stored by using the parErrorObjectElementIndex parameter, and the index of the attribute that caused an error is stored by using the parErrorObjectElementIndex parameter and parErrorAttributeElementIndex parameter. Therefore, there are three return types for the FAIL signature. For the first type, the location of error is not related to object or attribute, or information of error source location is skipped, and only the operation error code is returned. For the second type, the error source location is an object. For the last type, the error source location is an attribute.

다음은 위의 SetObjectGroupValue 오퍼레이션을 ASN.1 정규표현식으로 정의한 결과입니다.

The SetObjectGroupValue operation can be defined by using ASN.1 regular expressions as follows.

```
SetObjectGroupValue ::=
        SetObjectGroupValueOvld1 |
        SetObjectGroupValueOvld2 |
        SetObjectGroupValueOvld3

SetObjectGroupValueOvld1 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectDestinationList        UM3ObjectList,
        &parUM3ObjectSourceList             UM3ObjectList
}

SetObjectGroupValueOvld2 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectValueCondition         UM3ObjectValueCondition,
        &parUM3ObjectSourceList             UM3ObjectList
}

SetObjectGroupValueOvld3 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectDestinationList        UM3ObjectList,
        &parUM3ObjectValueCondition         UM3ObjectValueCondition,
        &parUM3ObjectSourceList             UM3ObjectList
}
```

13.16.2. REQUEST signature

13.16.2.1. parUM3ObjectDestinationList

해당 오퍼레이션 SetObjectGroupValue 가 적용될 오브젝트 리스트를 나타냅니다. 리스트 엘리먼트는 UM3ObjectIndicator 타입으로 정의되며 오퍼레이션의 적용범위를 오브젝트의 클래스아이디와 오브젝트 이름의 리스트로 표현합니다.

This contains the object list object to which the SetObjectGroupValue operation is applied. The list element is defined with the UM3ObjectIndicator type, and the range of operation is expressed as the list of class IDs and object names of the objects.

13.16.2.2. parUM3ObjectSourceList

parUM3ObjectSourceList 파라메터는 UM3ObjectList 타입의 파라메터로서, 선별적인 애트리뷰트의 값을 갖고 있는 복수개의 오브젝트들의 리스트 타입입니다. 선별적이라 함은 하나의 오브젝트를 구성하는 애트리뷰트들 중 parUM3ObjectSourceList 파라메터에 기록된 애트리뷰트에 대해서만 에이전트가 관리하는 정보모델의 오브젝트 애트리뷰트들의 값이 재설정 된다는 뜻입니다.

The parUM3ObjectSourceList parameter is UM3ObjectList type, and it is a list type for multiple objects with selective attributes. Selective means that values of the object of the information model managed by the agent are only applied to the attributes specified in the parUM3ObjectSourceList parameter among the attributes configuring the object.

앞서 정의한 바와 같이 UM3ObjectList 타입은 UM3 SEQUENCE OF 타입과 동일하며, 해당 타입의 엘리먼트는 ANY DEFINED BY UM3 ASN.1 module 로 정의됩니다. 즉, 모든 타입의 오브젝트들이 파라메터로 설정될 수 있다는 뜻입니다.

As defined earlier, the UM3ObjectList type is the same as UM3 SEQUENCE OF type, and the element of the corresponding type is defined as the ANY DEFINED BY UM3 ASN.1 module. In other words, any type of object can be used as a parameter.

13.16.3. REQUEST signature OVLD 1

13.16.3.1. parUM3ObjectValueCondition

UM3ObjectValueCondition 클래스 타입의 파라메터이며, 에이전트가 관리하는 정보모델 내부의 오브젝트들에 대하여 parUM3ObjectValueCondition 파라메터로 주어진 조건을 만족하는 경우 해당 오브젝트의

애트리뷰트 값들이 재설정되어야 합니다.

This parameter is UM3ObjectValueCondition class type. When the conditions provided through the parUM3ObjectValueCondition parameter about the objects in the information model is managed by the agent, the value of attributes of the corresponding object should be reset.

13.16.3.2. parUM3ObjectSourceList

상기 13.16.2절의 요청 signature 와 동일한 값으로 기록됩니다. 즉, 재설정할 애트리뷰트 값들로 이루어진 오브젝트들이 UM3ObjectList 타입의 엘리먼트들을 구성합니다. 해당 엘리먼트들의 타입은 ANY DEFINED BY UM3 ASN.1 module 타입으로 정의됩니다.

This contains the same value as the REQUEST signature in Section 13.16.2. The elements of the UM3ObjectList type are objects that have attributes to reset. The type of the elements is defined as ANY DEFINED BY UM3 ASN.1 module type.

13.16.4. REQUEST signature OVLD 2

13.16.4.1. parUM3ObjectDestinationList

UM3ObjectList 타입이며 동일한 타입으로 정의된 parUM3ObjectSourceList 와는 달리 해당 파라메터는 그 엘리먼트로 UM3ObjectIndicator 타입의 엘리먼트를 갖고 있습니다. parUM3ObjectDestinationList 파라메터는 parUM3ObjectValueCondition 과 parUM3ObjectSourceList 파라메터로 주어지는 조건문과 재설정을 위한 오브젝트 값들을 적용하기 위한 오브젝트 그룹의 범위를 명시하기 위해 사용됩니다.

에이전트는 parUM3ObjectDestinationList 파라메터에 기록된 오브젝트들 만을 대상으로 parUM3ObjectValueCondtion 파라메터에 기록된 조건을 적용하며 해당 조건이 만족될 경우, parUM3ObjectSourceList 를 통해 전달된 오브젝트의 값을 이용하여 조건을 만족하는 오브젝트들의 값을 재설정하게 됩니다.

상기 오퍼레이션의 조건과 그 범위를 적용함에 있어서 파라메터로 주어지는 parUM3ObjectSourceList 의 오브젝트들은 재설정하고자 하는 애트리뷰트들 만을 갖고 있습니다. 이는 주어진 범위내의 오브젝트들에 대한 조건을 적용하여 그 결과가 조건을 만족하는 경우에도, parUM3ObjectSourceList 파라메터에 명시되지 않은 오브젝트에 대해서는 해당 오퍼레이션이 적용되지 않을 수도 있음을 뜻합니다.

This is UM3ObjectList type. Unless the parUM3ObjectSourceList is defined with the same type, this parameter has the elements of the UM3ObjectIndicator type. The parUM3ObjectDestinationList parameter is used to specify the conditions provided through the parUM3ObjectValueCondition and parUM3ObjectSourceList parameter and the range of the object group to reset.

The agent only applies the conditions stored in the parUM3ObjectValueCondition parameter to the object specified in the parUM3ObjectDestinationList parameter, and it reset the values of objects that meet the conditions using the object values passed through parUM3ObjectSourceList.

When applying the above conditions and range of operation, the objects in the parUM3ObjectSourceList provided as the parameter has only the attributes to reset. This means that even when the objects within the specified range meeting are tested and results meet the conditions, the corresponding operation may not be applied to the objects that are not explicitly specified in the parUM3ObjectSourceList parameter.

13.16.4.2. parUM3ObjectValueCondition

UM3ObjectValueCondition 클래스 타입의 파라메터이며, 에이전트가 관리하는 정보모델 내부의 오브젝트들에 대하여 parUM3ObjectValueCondition 파라메터로 주어진 조건을 만족하는 경우 해당 오브젝트의 애트리뷰트 값들이 재설정되는 것으로 정의됩니다.

This parameter is UM3ObjectValueCondition class type. When objects in the information model managed by the agent meet the provided conditions, the attributes of the corresponding object are reset.

13.16.4.3. parUM3ObjectSourceList

상기 13.16.2 절의 요청 signature 와 동일한 값으로 정의됩니다. 재설정할 애트리뷰트 값들로 이루어진 오브젝트들이 UM3ObjectList 타입의 엘리먼트들을 구성하며, 해당 엘리먼트들의 타입은 ANY DEFINED BY UM3 ASN.1 module 타입으로 정의됩니다.

This contains the same value as the REQUEST signature in Section 13.16.2. Objects with the attributes to reset configure the elements of UM3ObjectList type, and the type of the corresponding element is defined as ANY DEFINED BY UM3 ASN.1 module type.

13.16.5. SUCCESS signature

상기 오퍼레이션이 성공하였을 경우에는 간략하게 그 결과의 성공 혹은 실패 여부만을 아래의 파라메터에 기록하여 다시 매니저로 전송합니다.

When the above operation is a success, the simple status of success or failure is stored in the following parameter and returned to the manager.

13.16.5.1. parResult : TRUE

상기 오퍼레이션이 성공했을 경우 parResult 파라메터에 TRUE 값을 기록하여 매니저로 다시 전송합니다.

When the above operation is a success, TRUE is set in parResult parameter and returned to the manager.

13.16.6. FAIL signature OVLD 3

SetObjectGroupValue 오퍼레이션이 실패했을 경우 OperationResponse 오퍼레이션의 signature 를 나타냅니다. 해당 signature 는 오퍼레이션의 실패를 나타내는 parResult 파라메터와 실패한 원인을 나타내는 parOperationErrorCode 의 두 가지 파라메터 만으로 구성됩니다.

해당 signature 는 오퍼레이션 수행 실패의 원인이 특정 오브젝트나 애트리뷰트에 있지 않고 다른 원인일 경우에 사용합니다. 혹은 경우에 따라 실패의 원인을 제공한 오브젝트나 애트리뷰트에 관한 정보를 리턴하지 않고 간략하게 결과만을 리턴하기 위한 용도로도 쓰일 수 있습니다. 단, 후자의 경우 매니저는 오퍼레이션이 실패한 원인을 알 수 없으며, 결과적으로 오퍼레이션의 동일한 실패가 반복될 수도 있다는 점에 유의해야 합니다.

This is the signature of OperationResponse operation when SetObjectGroupValue operation fails. This signature is configured with only two parameters: these are the parResult parameter that indicates the failure of an operation and the parOperationErrorCode that contains the cause of failure.

This signature is used when the cause of failure is not a specific object or attribute. It can alternatively be used to return the fail information without returning the information about the object or attribute that caused the failure. In this case, the manager has no means of knowing the cause of failure, and the same failure could be repeated as a result.

13.16.6.1. parResult : FALSE

상기 오퍼레이션이 성공했을 경우 parResult 파라메터에 TRUE 값을 기록하여 매니저로 다시 전송합니다.

When the above operation is a success, TRUE is set to the parResult parameter and it is returned to the manager.

13.16.6.2. parOperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며, 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.16.7. FAIL signature OVLD 4

SetObjectGroupValue 오퍼레이션이 실패했을 경우 signature 를 나타냅니다. 오퍼레이션 실패의 원인이 오브젝트에 있을 경우 해당 오브젝트가 parUM3ObjectSourceList 의 몇 번째 오브젝트인가를 나타내는

parErrorObjectElementIndex 파라메터를 통해 오류 발생 오브젝트의 위치를 매니저로 전송합니다.

This is the signature used when the SetObjectGroupValue operation fails. When an object causes failure of operation, the agent returns the index of the object that caused an error through the parErrorObjectElementIndex parameter that indicates the location of the object in parUM3ObjectSourceList.

13.16.7.1. parResult : FALSE

상기 오퍼레이션이 성공했을 경우 parResult 파라메터에 TRUE 값을 기록하여 매니저로 다시 전송합니다.

When the above operation is a success, TRUE is set to parResult parameter and it is returned to the manager.

13.16.7.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며, 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.16.8. parErrorObjectElementIndex

오류가 발생한 오브젝트의 위치를 나타내며, 파라메터 parUM3ObjectDestinationList 를 구성하는 엘리먼트들 중 첫 번째 엘리먼트를 0 으로 두었을 때 해당 오브젝트가 몇 번째 인가에 관한 정보를 기록합니다.

This indicates the location of the object that caused an error. The index of the first element configuring the parameter parUM3ObjectDestinationList is set to 0 and this index indicates the location of the corresponding object in the list.

13.16.9. FAIL signature OVLD 9

SetObjectGroupValue 오퍼레이션이 실패했을 경우 signature 를 나타내며, 오퍼레이션 실패의 원인이 애트리뷰트에 있을 경우 해당 애트리뷰트가 parUM3ObjectSourceList 의 몇 번째 오브젝트의 몇 번째 애트리뷰트인가를 나타내는 parErrorObjectElementIndex 와 parErrorAttributeElementIndex 파라메터를 통해 오류 발생 애트리뷰트의 위치를 매니저로 전송합니다.

This signature is used when the SetObjectGroupValue operation fails. When an attribute causes failure of operation, the agent returns the index of the attribute that caused an error through the parErrorObjectElementIndex and parErrorAttributeElementIndex parameter that indicates the location of the attribute in parUM3ObjectSourceList.

13.16.9.1. parResult : FALSE

상기 오퍼레이션이 성공했을 경우 parResult 파라메터에 FALSE 값을 기록하여 매니저로 다시 전송합니다.

When the above operation is a success, FALSE is set to parResult parameter and it is returned to the manager.

13.16.9.2. parUM3OperationErrorCode

UM3OperationErrorCode 클래스 타입의 파라메터이며, 오퍼레이션이 실패한 원인을 기록합니다.

This parameter is UM3OperationErrorCode class type, and it contains the cause of failure of operation.

13.16.9.3. parErrorObjectElementIndex

오류가 발생한 애트리뷰트가 속한 오브젝트의 위치를 나타내며, 파라메터 parUM3ObjectDestinationList 를 구성하는 엘리먼트들 중 첫 번째 엘리먼트를 0 으로 두었을 때 해당 오브젝트가 몇 번째 인가에 관한 정보를 기록합니다.

This indicates the location of the object that has an attribute causing an error. The index of the first element configuring the parameter parUM3ObjectDestinationList is set to 0, and this index indicates the location of the corresponding object in the list.

13.16.9.4. parErrorAttributeElementIndex

오류가 발생한 애트리뷰트의 위치를 나타냅니다. 해당 위치 정보는 특정 오브젝트에 *m* 개의 애트리뷰트가 존재할 때, 맨 첫 번째로 디코딩 된 애트리뷰트의 인덱스 번호를 0 으로 두고, 해당 오류 발생 애트리뷰트가 맨 마지막에 디코딩 된 애트리뷰트라면, 해당 애트리뷰트의 인덱스 번호는 *(m -1)* 로 표현하게 됩니다.

This contains the index of the attribute that caused an error. When there are *m* attributes in a specific object, the index of the first decoded attribute is set to 0. If the attribute associated with the error is the last decoded, then the index of the corresponding attributes is expressed as *(m -1)*.

13.16.10. Handling Procedure at the Receiver

앞서 정의한 것과 마찬가지로 SetObjectGroupValue 오퍼레이션은 설정하고자 하는 오브젝트들을 모두 parUM3ObjectSourceList 파라메터에 기록하여 전송합니다. 이 때 각각의 오브젝트들은 그 값을 갱신할 필요가 있는 애트리뷰트들 만을 갖고 있어야 합니다. 이를 수신한 에이전트는 파라메터로 주어진 오브젝트들에 대해서 다음과 같은 과정을 거쳐 그 오브젝트들의 애트리뷰트 값들을 갱신하게 됩니다.

즉, um3ClassIdentifier 와 um3ObjectName 애트리뷰트의 값을 확인하여 파라메터가 가리키는 오브젝트를 찾습니다. 해당 오브젝트를 찾아낸 후 에이전트는 파라메터에 포함된 오브젝트의 애트리뷰트들을 하나씩 확인하고 그 확인한 값으로, 자신이 관리하는 오브젝트의 애트리뷰트 값들을 갱신해 나갑니다. 이러한 과정은 파라메터로 주어진 모든 오브젝트들에 대해 동일하게 적용됩니다.

As defined earlier, the SetObjectGroupValue operation stored the objects to set in the parUM3ObjectSourceList parameter and transmits the data. In this case, each object should only have the attributes to update the values. When the agent receives the data as a parameter, it updates the attribute values of the objects through the following procedure. It checks the values of the um3ClassIdentifier and um3ObjectName attribute and finds the object pointed by the parameter. Once the object is found, the agent checks the attributes of the object included in the parameter one by one and updates the value of the object managed by itself. This procedure is applied in exactly the same way to all objects provided as parameters.

그림 22-SetObjectGroupValue 오퍼레이션의 수행 [는 에이전트가 관리하는 오브젝트의 애트리뷰트의 값들을 갱신함에 있어서, 파라메터로 주어진 오브젝트의 애트리뷰트들에 대해서만 그 값을 갱신하는 과정을 표시하고 있습니다. 즉, Object A 의 애트리뷰트는 attribute a, attribute b, attribute e 등 3개의 애트리뷰트만을 갱신토록 파라메터에 포함되어 있음을 볼 수 있습니다.

그림 22-SetObjectGroupValue 오퍼레이션의 수행 [shows the process of updating the values for only the attributes of objects provided as parameters while updating the values of attributes of the objects managed by the agent. For example, only three attributes of attribute a, attribute b, and attribute e of Object A should be updated as specified by the parameter.

이러한 과정 중 특정 애트리뷰트의 타입이 불일치하거나, 혹은 특정 애트리뷰트가 존재하지 않거나, 혹은 특정 오브젝트의 타입의 불일치, 오브젝트가 존재하지 않는 등의 여러 가지 오류가 발생할 수 있습니다. 따라서, SetObjectValueGroup 오퍼레이션과 같이 특정 그룹의 복수개의 오브젝트를 대상으로 이루어지는 오퍼레이션은 반드시 해당 오퍼레이션을 호출하기 전에 매니저와 에이전트 사이의 정보모델을 구성하는 오브젝트와 해당 오브젝트의 애트리뷰트들에 대한 구성을 일치시킨 후 실행해야 합니다.

There are few sources of error in this process such as inconsistency of the types of attributes, a specific attribute does not exist, inconsistency of the types of specific objects, or a specific object does not exist. Therefore, the manager and the agent should synchronize the configuration of attributes of the target objects and attributes configuring the information between them before calling the operation about multiple objects in a specific group such as in a SetObjectValueGroup operation.

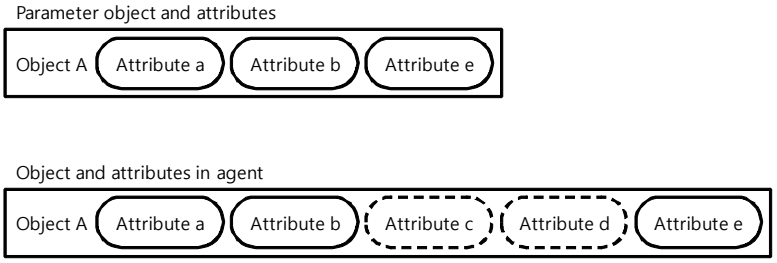


그림 22-SetObjectGroupValue 오퍼레이션의 수행 [Execution of SetObjectGroupValue operation]

13.17. SetObjectAttributeGroupValue operation

SetObjectAttributeGroupValue 오퍼레이션은 한 개의 오브젝트에 속해 있는 복수의 애트리뷰트들에 대해 한 번에 그 값들을 설정하기 위해 사용합니다. 앞서 정의한 GetObjectAttributeGroupValue 와 마찬가지로 SetObjectAttributeGroupValue 오퍼레이션은 복수의 오브젝트에 대한 오퍼레이션이 아닌 한 개의 오브젝트에 대한 오퍼레이션임에 유의해야 합니다.

The SetObjectAttributeGroupValue operation is used to set the values of multiple attributes in an object in one operation. Like the GetObjectAttributeGroupValue defined earlier, the SetObjectAttributeGroupValue operation works over one operation, not multiple objects.

13.17.1. Structure of Signature and Return Type

SetObjectAttributeGroupValue 오퍼레이션은 다음과 같은 요청 signature 와 OperationResponse 오퍼레이션의 signature 를 갖고 있습니다.

표 58-SetObjectAttributeGroupValue 오퍼레이션의 signature 와 리턴 타입 구조 에서 보는 바와 같이 애트리뷰트의 값들을 설정하기 위한 특정 오브젝트를 가리키는 parUM3ClassIdentifier 와 parUM3ObjectName 애트리뷰트를 갖고 있습니다. 그리고, 해당 오브젝트의 애트리뷰트의 리스트를 parUM3ObjectList 에 기록하여 에이전트로 전송합니다.

The SetObjectAttributeGroupValue operation has the following REQUEST signature and the signature of OperationResponse operation.

As shown in Table 57, it has the parUM3ClassIdentifier and parUM3ObjectName attribute that point at a specific object to set the values of attributes. The list of attributes of the corresponding object is stored in the parUM3ObjectList and sent to the agent.

표 58-SetObjectAttributeGroupValue 오퍼레이션의 signature 와 리턴 타입 구조
[Structure of signature and return type of SetObjectAttributeGroupValue operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ClassIdentifier UM3ObjectName UM3ObjectList	parUM3ClassIdentifier parUM3ObjectName parUM3ObjectList	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult: FALSE parUM3OperationErrorCode parErrorElementIndex	OVLD 4

다음은 SetObjectAttributeGroupValue 오퍼레이션의 ASN.1 정규표현식으로의 표현입니다.

SetObjectAttributeGroupValue operation can be defined by using ASN.1 regular expressions as follows.

```
SetObjectAttributeGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier            UM3ClassIdentifier,
        &parUM3ObjectName                 UM3ObjectName,
        &parUM3ObjectList                 UM3ObjectList
}
```

13.17.2. REQUEST signature

13.17.2.1. parUM3ClassIdentifier

애트리뷰트 그룹이 속해 있는 오브젝트의 클래스 아이디를 나타냅니다.

This contains the class ID of the object that has the attribute group.

13.17.2.2. parUM3ObjectName

애트리뷰트 그룹이 속해 있는 오브젝트의 오브젝트 이름을 나타냅니다.

This contains the object name of the object that has the attribute group.

13.17.2.3. parUM3ObjectList

애트리뷰트의 그룹을 나타냅니다. 즉, 갱신하고자 하는 애트리뷰트의 리스트가 UM3ObjectList 타입으로 기록됩니다. 또한 parUM3ObjectList 파라메터를 구성하는 엘리먼트는 ANY DEFINED BY UM3 ASN.1 module 타입으로 정의됩니다. 즉, 모든 타입의 애트리뷰트가 모두 parUM3ObjectList 파라메터의 엘리먼트로 올 수 있습니다.

This contains the group of the attribute. The list of attributes to update is stored as UM3ObjectList type. The elements configuring the parUM3ObjectList parameter are defined as ANY DEFINED BY UM3 ASN.1 module type. In other words, any type of attribute can be used for the element of the parUM3ObjectList parameter.

13.17.3. SUCCESS signature

13.17.3.1. parResult : TRUE

상기 SetObjectAttributeGroupValue 오퍼레이션이 성공하였을 경우, UM3Boolean 타입의 parResult 파라메터에는 TRUE 값이 기록되어 전송됩니다.

When the above SetObjectAttributeGroupValue operation is a success, the parResult parameter of UM3Boolean type is set to TRUE and the result is returned.

13.17.4. FAIL signature OVLD 4

13.17.4.1. parResult : FALSE

상기 SetObjectAttributeGroupValue 오퍼레이션이 실패하였을 경우, UM3Boolean 타입의 parResult 파라메터에는 FALSE 값이 기록되어 전송됩니다.

When the above SetObjectAttributeGroupValue operation is a success, the parResult parameter of UM3Boolean type is set to FALSE and the result is returned.

13.17.4.2. parOperationErrorCode

parOperationErrorCode 파라메터에는 오퍼레이션이 실패한 원인이 기록되어 전송됩니다.

The cause of failure of operation is stored in the parOperationErrorCode parameter and the result is returned.

13.17.4.3. parErrorElementIndex

파라메터로 주어진 parUM3ObjectList 의 애트리뷰트들 중 오퍼레이션의 수행 도중 오류가 발생할 경우, 해당 애트리뷰트의 인덱스 번호를 나타냅니다. 즉, parUM3ObjectList 에 기록된 첫 번째 애트리뷰트의 인덱스 번호를 0 이라고 했을 때, 오류가 발생한 해당 애트리뷰트의 인덱스 번호를 나타냅니다.

If an error occurs during operation from one of attributes in the parUM3ObjectList provided as a parameter, then it contains the index number of the corresponding attribute. If the index of the first attribute stored in the parUM3ObjectList is 0, then it is the index of the corresponding attribute that caused an error.

13.17.5. Handling Procedure at the Receiver

SetObjectAttributeGroupValue 오퍼레이션을 수신한 에이전트는 parUM3ClassIdentifier 와 parUM3ObjectName 파라메터를 이용하여 오퍼레이션을 적용할 오브젝트를 찾습니다. 해당 오브젝트를 찾아낸 후 parUM3ObjectList 파라메터에 기록되어 있는 해당 애트리뷰트의 값들을 자신이 관리하는 오브젝트의 애트리뷰트 값으로 갱신하게 됩니다.

When an agent receives the SetObjectAttributeGroupValue operation, it searches for the object to apply the operation by using the parUM3ClassIdentifier and parUM3ObjectName parameter. Once the corresponding object is found, the values of the corresponding attributes stored in the parUM3ObjectList parameter are updated by the attribute values of the object managed by itself.

SetObjectGroupValue 와 마찬가지로 에이전트는 parUM3ObjectList 파라메터에 기록되어 있는 애트리뷰트에 대해서만 그 값을 갱신합니다. 즉, parUM3ObjectList 파라메터에는 해당 오브젝트의 모든 애트리뷰트가 기록될 필요는 없으며, 단지 갱신이 필요한 애트리뷰트들만 기록되어 전송되어야 합니다.

Like the SetObjectGroupValue, the agent only updates the attributes specified in the parUM3ObjectList parameter. Not all attributes of the corresponding object need to be specified in the parUM3ObjectList parameter, and only the attributes to be updated should be specified and passed.

해당 오퍼레이션의 수행 도중 오류가 발생한 애트리뷰트에 대한 수정 혹은 대응 오퍼레이션의 수행을 통해 오류가 정정되었다 하더라도 parUM3ObjectList 파라메터를 구성하는 다음 애트리뷰트들에 대한 오퍼레이션의 수행 과정에 다시 오류가 발생 할 수도 있습니다. 즉, 에이전트는 실패 signature 를 통해 처음으로 오류가 발생할 경우 바로 오퍼레이션의 수행을 중단하고 오류발생 사실을 리턴해야 합니다.

When error is fixed by modifying the attributes that caused an error or execution of counter operation about the errors

occurring during operation, the error could occur again during operation for the next attribute configuring the parUM3ObjectList parameter. When an error is reported the first time through a FAIL signature, the agent should stop the operation immediately and return the error status.

13.18. CreateObject operation

CreateObject 오퍼레이션은 매니저가 에이전트에게 본 권고안에 정의된 클래스 타입들 중 하나를 생성토록 하는 오퍼레이션입니다. CreateObject 오퍼레이션은 시스템의 운용 중 새로운 장치가 신규로 설치되거나 혹은 다이내믹하게 장치 혹은 오브젝트의 설치 및 철거 혹은 삭제가 필요할 때 DeleteObject 오퍼레이션과 함께 유용하게 사용할 수 있는 오퍼레이션입니다.

The CreateObject operation is used for the manager to instruct the agent to create one of the class types defined in this recommendation. The CreateObject operation can be useful along with the DeleteObject operation when a new device is installed, or objects are installed, removed, or deleted dynamically during system operation.

13.18.1. Structure of Signature and Return Type

CreateObject 오퍼레이션의 signature 와 그 리턴 타입의 구조는 아래와 같습니다.

The structure of signature and return type of the CreateObject operation are as follows.

표 59-CreateObject 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of CreateObject operation]

Classification	Type of Parameter	Name and Value of Parameter	Characteristics
REQUEST	UM3CharacterString ANY DEFINED BY UM3 ASN.1 module	parUM3DnObjectName parAnyTypeValue	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult: FALSE parUM3OperationErrorCode parErrorElementIndex	OVLD 4

CreateObject 오퍼레이션의 요청 signature 는 parUM3DnObjectName, parAnyTypeValue 등으로 구성됩니다.

다음은 상기 ObjectCreate 오퍼레이션의 ASN.1 정규표현식으로의 정의입니다.

The REQUEST signature of the CreateObject operation has parUM3DnObjectName and parAnyTypeValue.

ObjectCreate operation can be defined by using ASN.1 regular expressions as follows.

```
CreateObject ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName UNIQUE,
        &parUM3DnObjectName               UM3CharacterString,
        &parAnyTypeValue                  ANY DEFINED BY UM3 ASN.1 module
}
```

13.18.2. REQUEST signature

13.18.2.1. parUM3DnObjectName

생성하고자 하는 오브젝트의 DN 을 기록합니다. 상기 parUM3ObjectName 은 RDN 을 기록하며 parUM3DnObjectName 에는 DN 을 기록합니다. 앞서 정의한 바와 같이 본 권고안이 정의하는 UM3 DN 은 그림 18-RDN과 DN [RDN and DN 에서 보는 바와 같이 해당 오브젝트의 포함관계 (containment relationship) 을 모두 나타내게 됩니다. 따라서, parUM3DnObjectName 파라메터로 주어지는 DN 을 확인할 경우 parAnyTypeValue 파라메터에 기록된 오브젝트를 어떤 오브젝트 하위에 생성시킬 것인지 그 위치를 명확하게 확인할 수 있습니다. 예를 들어 그림 18-RDN과 DN [에 표시된 AnalogSensor 오브젝트를 생성하기 위해 CreateObject 오퍼레이션을 사용하는 경우 parUM3DnObjectName 파라메터의 값은 아래와 같이 표현할 수 있습니다.

This contains the DN of the object to create. The parUM3ObjectName contains RDN and parUM3DnObjectName contains DN. As defined earlier, UM3 DN defined in this recommendation shows all the containment relationships of the corresponding object as shown in 그림 18-RDN과 DN [RDN and DN. Therefore, the DN provided as parUM3DnObjectName parameter provides clear information about the location to create a new object specified in the parAnyTypeValue parameter, such as the location under a certain object. For example, when the CreateObject operation is used to create an AnalogSensor object shown in Fig. 18, the value of the parUM3DnObjectName parameter can be expressed as follows.

parUM3DnObjectName UM3CharacterString ::= “myGateway.mySensor”

상기 ASN.1 정규표현식에서 ‘myGateway’ 와 ‘mySensor’ 는 각각 RDN 이며, ‘myGateway.mySensor’ 는 DN 으로 정의됩니다.

In this ASN.1 regular expression, ‘myGateway’ and ‘mySensor’ are RDN and ‘myGateway.mySensor’ is DN.

13.18.2.2. parAnyTypeValue

parAnyTypeValue 는 새롭게 생성하고자 하는 오브젝트의 값 즉, 애트리뷰트 값들을 기록합니다. parAnyTypeValue 파라메터는 ANY DEFINED BY UM3 ASN.1 module 타입으로 정의되어 있으며, 본 권고안이 정의하는 모든 타입들이 해당 파라메터의 타입으로 적용 가능합니다. 단 해당 오브젝트의 애트리뷰트의 종류는 parAnyTypeValue 파라메터의 오브젝트에 기록된 애트리뷰트만을 생성하며 그 이외의 애트리뷰트들의 값은 지정되지 않습니다.

parAnyTypeValue contains the values of the object to create i.e. attribute values. The parAnyTypeValue parameter is defined as ANY DEFINED BY UM3 ASN.1 module type, and any type defined in this recommendation can be used as the type of corresponding parameter. Only the attributes specified in the object of parAnyTypeValue parameter are created, and the values of other attributes are not assigned.

13.18.3. SUCCESS signature

13.18.3.1. parResult : TRUE

상기 오퍼레이션이 성공할 경우 OperationResponse 오퍼레이션은 parResult 파라메터에 TRUE 값을 기록하여 응답 패킷을 전송합니다.

When this operation is a success, the OperationResponse operation sets the parResult parameter as TRUE and sends the response packet.

13.18.4. FAIL signature OVLD 4

13.18.4.1. parResult : FALSE

상기 오퍼레이션의 수행이 실패했을 경우 OperationResponse 오퍼레이션은 parResult 파라메터에 FALSE 값을 기록합니다.

When this operation fails, the OperationResponse operation sets the parResult parameter as FALSE.

13.18.4.2. parUM3OperationErrorCode

상기 오퍼레이션의 수행이 실패할 경우 OperationResponse 오퍼레이션의 parUM3OperationErrorCode 에는 해당 오퍼레이션의 실패 원인이 기록됩니다.

When this operation fails, the cause of failure of the operation is stored in the parUM3OperationErrorCode of the OperationResponse operation.

만약 parUM3OperationErrorCode 가 애트리뷰트의 생성과정에서 발생한 오류와 관련된 코드로 표현되고 있다면, 다음절에서 정의하는 parErrorElementIndex 파라메터에 정의된 애트리뷰트의 인덱스 값을 참조하여 해당 오류에 대한 대응 처리 과정을 수행해야 합니다. 만약 parUM3OperationErrorCode 파라메터가 애트리뷰트와 관계없는 오류를 나타내고 있다면 다음 절에서 정의하는 parErrorElementIndex 파라메터의 값은 무시해도 상관 없는 것으로 정의합니다. 이러한 parUM3OperationErrorCode 와 parErrorElementIndex 파라메터의 값 사이의 관계는 본 권고안이 정의하는 모든 OperationResponse 오퍼레이션의 파라메터에 동일하게 적용됩니다.

If the parUM3OperationErrorCode contains the error that occurred during the process of attribute creation or related code, then the process to handle the corresponding error should be executed with the index value of the attribute defined in the parErrorElementIndex parameter, which is defined in the next section. If the parUM3OperationErrorCode parameter contains an error that is not related to specific attribute, then the value of the parErrorElementIndex parameter defined in the next section can be ignored in this definition. This relationship between the values of parUM3OperationErrorCode and parErrorElementIndex parameter applies to the parameters of all OperationResponse operations defined in this recommendation in the same way.

13.18.4.3. parErrorElementIndex

상기 오퍼레이션의 수행이 실패할 경우, 특히 특정 애트리뷰트의 생성과 관련된 오류가 발생할 경우 해당 애트리뷰트가 몇 번째로 생성 작업이 진행된 것인지에 관한 정보가 기록됩니다.

When the above operation fails, especially when there is error related to the creation of a specific attribute, it stores the index information to tell the sequence of the creation of the corresponding attribute.

본 권고안은 오브젝트를 인코딩 함에 있어서 그 애트리뷰트의 순서에 제한을 두고 있지 않지만, 파라메터 값으로 기록되는 시점에는 임의의 순서대로 애트리뷰트가 인코딩 되어 기록되게 됩니다. 따라서, 해당 애트리뷰트의 순서를 나타내는 인덱스 정보를 활용하여 어떤 애트리뷰트의 생성과정에서 오류가 발생했는지를 확인할 수 있습니다.

This recommendation does not put any restriction on the sequence of the attributes in encoding the objects, and attributes are encoded and stored in random order with the time when the parameter values are recorded. Therefore, the occurrence of error while creating a certain attribute can be checked by using the index information that indicates the sequence of the corresponding attribute.

13.18.5. Handling Procedure at the Receiver

본 권고안이 정의하는 UM3 프로토콜의 모든 클래스 타입들은 UM3ClassIdentifier 와 UM3ObjectName 타입의 um3ClassIdentifier, um3ObjectName 애트리뷰트들을 갖고 있습니다. 즉, 해당 애트리뷰트의 값을 확인할 경우 어떤 클래스 타입인지와 해당 오브젝트의 이름을 확인할 수 있습니다. 또한 함께 주어지는 parUM3DnObjectName 파라메터의 값은 해당 오브젝트가 어떤 위치에 생성되어야 할 지를 가리키게 됩니다. CreateObject 오퍼레이션이 실패하는 대부분의 원인은 parUM3DnObjectName 으로 주어지는 DN 의 값이 잘못 주어진 경우입니다. 또한 이미 다른 프로세스가 해당 오브젝트를 포함해야 하는 상위 오브젝트를 삭제한 경우도 있을 수 있습니다. 이러한 비동기 방식의 생성 및 삭제 관련 오퍼레이션은 오류가 발생할 확률이 높으므로 구현측면에서는 주의를 요합니다.

All class types of the UM3 protocol defined in this recommendation have um3ClassIdentifier, um3ObjectName attribute of UM3ClassIdentifier, and UM3ObjectName type, respectively. The class type and name of the corresponding object can be checked by using the value of the corresponding attribute. The value of the parUM3DnObjectName parameter that is provided together indicates where the corresponding object should be created. The major cause of failure of the CreateObject operation is a wrong value for DN provided as parUM3DnObjectName. Another cause would be that the upper level object that should contain the corresponding object is deleted by another process. Caution is required in implementation because asynchronous creation and deletion of related operations is likely to cause errors.

13.19. CreateObjectGroup operation

상기 CreateObject 오퍼레이션이 한 개의 오브젝트를 생성하기 위한 오퍼레이션인 반면 CreateObjectGroup 오퍼레이션은 한 개 이상 복수의 오브젝트를 생성하기 위한 오퍼레이션 입니다.

While the CreateObject operation is used for creating one object, the CreateObjectGroup operation is for creating multiple objects.

13.19.1. Structure of Signature and Return Type

CreateObjectGroup 오퍼레이션의 signature 와 리턴 타입의 구조는 다음과 같습니다.

The structure of Signature and Return Type of the CreateObjectGroup operation are as follows.

표 60-CreateObjectGroup 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of CreateObjectGroup operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3UnsignedInteger16 UM3ObjectListToBeCreated	parNoOfObjectsToBeCreated parUM3ObjectListToBeCreated	
SUCCESS	UM3Boolean	parResult: TRUE	

FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult: FALSE parUM3OperationErrorCode parErrorObjectElementIndex	OVLD 4
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16 UM3UnsignedInteger16	parResult : FALSE parUM3OperationErrorCode parErrorObjectElementIndex parErrorAttributeElementIndex	OVLD 9

위의 signature 정의에서 parNoOfObjectsToBeCreated 는 생성하고자 하는 오브젝트의 개수를 나타내며, parUM3ObjectListToBeCreated 파라메터는 UM3ObjectListToBeCreated 타입으로 정의되어 있습니다.

다음은 위의 CreateObjectGroup 오퍼레이션의 ASN.1 정규표현식으로의 정의입니다.

In this definition of signatures, parNoOfObjectsToBeCreated contains the number of objects to create, and the parUM3ObjectListToBeCreated parameter is UM3ObjectListToBeCreated type.

The CreateObjectGroup operation can be defined by using ASN.1 regular expressions as follows.

```
CreateObjectGroup ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parNoOfObjectsToBeCreated          UM3UnsignedInteger16,
        &parUM3ObjectListToBeCreated        UM3ObjectListToBeCreated
}
```

UM3ObjectListToBeCreated 타입의 정의는 10.39 절에서 정의한 바와 같습니다.

UM3ObjectListToBeCreated type is defined in the section 10.39.

13.19.2. REQUEST signature

CreateObjectGroup 오퍼레이션은 앞서 정의한 CreateObject 오퍼레이션과 동일한 형식을 갖습니다. 다만, CreateObject 오퍼레이션이 한 쌍의 파라메터로 한 개의 오브젝트에 대한 생성을 지시하는 것과는 달리, CreateObjectGroup 오퍼레이션은 여러 개의 DN 과 ANY DEFINED BY UM3 ASN.1 module 타입으로 복수의 오브젝트에 대한 생성을 지시합니다.

The CreateObjectGroup operation is the same type as the CreateObject operation. Unlike the CreateObject operation that instructs the creation of one object with a pair of parameters, the CreateObjectGroup operation instructs the

creation of multiple objects by using multiple DN and ANY DEFINED BY UM3 ASN.1 module type.

13.19.2.1. parNoOfObjectsToBeCreated

생성하고자 하는 오브젝트의 개수를 나타내며, 그 타입은 UM3UnsignedInteger16 으로 정의됩니다. 해당 파라메터는 매니저와 에이전트의 연산의 용이성을 확보하고 오류를 줄이기 위한 목적으로 사용되는 파라메터 입니다. 즉, 매니저로 하여금 자신이 생성하는 오브젝트의 정확한 개수를 계속 확인토록 하는 효과가 있으며, 에이전트는 파라메터에 포함된 생성 대상 오브젝트의 개수를 미리 파악하도록 함으로써 불필요한 연산의 복잡성을 줄이기 위한 목적을 갖고 있습니다.

This contains the number of objects to create, and the type is defined with UM3UnsignedInteger16. This parameter is used for securing the convenience of computation for the manager and agent and for reducing errors. It will ensure that the manager keeps track of the number of objects created by itself accurately and that the agent obtains the information of the number of target objects to create as specified in the parameter to reduce the complexity of the computation.

이 때 앞서 정의한 그룹 대상 오브젝트들에 대한 오퍼레이션에서와 마찬가지로, 개수는 1 이상의 값을 가져야 하며, parUM3ObjectListToBeCreated 파라메터에 저장된 첫 번째 엘리먼트의 번호는 0 으로 지정하여 연산이 이루어져야 합니다.

Like the operations of group objects defined earlier, it should have a number equal to or greater than 1, and the index of the first element stored in the parUM3ObjectListToBeCreated parameter should be set to 0 for computation purposes.

13.19.2.2. parUM3ObjectListToBeCreated

parUM3ObjectListToBeCreated 파라메터는 UM3ObjectListToBeCreated 타입으로 정의된 파라메터 입니다. UM3ObjectListToBeCreated 타입은 그림 9 에 나타낸 것과 같은 구조를 갖습니다. 즉, 생성하고자 하는 오브젝트의 DN 과 해당 오브젝트의 값으로 만들어진 ANY DEFINED BY UM3 ASN.1 module 타입의 오브젝트로 구성되어 있습니다.

The parUM3ObjectListToBeCreated parameter is UM3ObjectListToBeCreated type. UM3ObjectListToBeCreated type has a structure as shown in the 그림 9, containing objects of ANY DEFINED BY UM3 ASN.1 module type with DN of the object to create and the value of the corresponding object.

앞서 정의한 것과 마찬가지로 해당 파라메터를 구성하는 DN 과 ANY DEFINED BY UM3 ASN.1 module 타입의 쌍을 이루는 엘리먼트는 그 시작 번호를 0 으로 시작하여야 합니다. 즉, 상기 parNoOfObjectsToBeCreated 파라메터의 값에서 1 을 뺀 숫자가, 상기 맨 마지막 엘리먼트의 번호와 일

치하여야 합니다.

As defined earlier, the index of the first element of the pair of DN and ANY DEFINED BY UM3 ASN.1 module type configuring the corresponding parameter should start from 0. The index of the last element should be equal to the value of the parNoOfObjectsToBeCreated parameter subtracted by 1.

13.19.3. SUCCESS signature

13.19.3.1. parResult : TRUE

UM3Boolean 타입을 갖는 parResult 파라메터에 TRUE 값이 기록되어 전송됩니다.

The parResult parameter of the UM3Boolean type is set to TRUE, and the result is returned.

13.19.4. FAIL signature OVLD 4

상기 CreateObjectGroup 오퍼레이션의 수행이 실패 할 경우, 에이전트는 OperationResponse 오퍼레이션의 파라메터를 다음과 같이 구성하여 매니저로 전송합니다.

If the CreateObjectGroup operation fails, then the agent configures the parameter of the OperationResponse operation as follows and sends it to the manager.

13.19.4.1. parResult : FALSE

CreateObjectGroup 오퍼레이션이 실패할 경우 UM3Boolean 타입을 갖는 parResult 파라메터에는 FALSE 가 기록되어 전송됩니다.

If the CreateObjectGroup operation fails, then the parResult parameter of UM3Boolean type is set to FALSE and the result is returned.

13.19.4.2. parOperationErrorCode

CreateObjectGroup 오퍼레이션이 실패 했을 경우 parOperationErrorCode 에는 실패의 원인을 나타내는 코드가 기록되어 매니저로 전송됩니다.

If the CreateObjectGroup operation fails, then the code that indicates the cause of failure is stored in the parOperationErrorCode and it is sent to the manager.

13.19.4.3. parErrorObjectElementIndex

CreateObjectGroup 오퍼레이션의 실행결과가 실패일 경우, parErrorObjectElementIndex 에는 오브젝트의 생성 도중 발생한 에러가 몇 번째 오브젝트의 생성과정에서 발생했는지에 관한 정보를 기록합니다. 마찬가지로 기록되는 번호는 2 바이트의 부호화하지 않은 양의 정수이며, 그 숫자는 0 부터 시작되는 해당 엘리먼트의 숫자 혹은 인덱스 번호가 기록되어야 합니다.

If the CreateObjectGroup operation fails, then the index of the objects in order of creation required to find the object that causes error is stored in the parErrorObjectElementIndex. It is an unsigned positive two byte number starting from 0.

만약 오브젝트의 생성과정이 아닌 다른 과정의 연산 도중 오류가 발생할 경우 parErrorObjectElementIndex 파라메터에는 0 이 기록되어 전송되며, 다른 과정에서 발생한 오류를 나타내는 정보는 parOperationErrorCode 에 기록되게 됩니다.

If an error occurs during computation rather than the creation of an object, then the value of parErrorObjectElementIndex parameter is set to 0 and returned, and the information about the errors that occurred in another process is stored in the parOperationErrorCode.

13.19.5. FAIL signature OVLD 9

OperationResponse 오퍼레이션의 실패 signature OVLD 2 는 특정 오브젝트의 특정 애트리뷰트의 생성 과정에서 오류가 발생할 경우 해당 오브젝트와 해당 애트리뷰트의 정보를 매니저로 리턴하기 위해 사용합니다.

FAIL signature OVLD 2 of OperationResponse operation is used to return the information of the specific object and specific attribute to the manager when an error occurs while creating the corresponding attribute of the corresponding object.

13.19.5.1. parResult : FALSE

CreateObjectGroup 오퍼레이션이 실패할 경우 UM3Boolean 타입을 갖는 parResult 파라메터에는 FALSE 가 기록되어 전송됩니다.

If the CreateObjectGroup operation fails, then the parResult parameter of UM3Boolean type is set to FALSE and the result is returned.

13.19.5.2. parOperationErrorCode

CreateObjectGroup 오퍼레이션이 실패 했을 경우 parOperationErrorCode 에는 실패의 원인을 나타내는

코드가 기록되어 매니저로 전송됩니다.

If the CreateObjectGroup operation fails, then the cause of failure is stored in parOperationErrorCode and it is returned to the manager.

13.19.5.3. parErrorObjectElementIndex

CreateObjectGroup 오퍼레이션의 실행결과가 실패일 경우, 특히 특정 오브젝트를 생성하는 과정 중에 수반되는 특정 애트리뷰트의 생성과정에서 오류가 발생할 경우, 해당 오브젝트와 애트리뷰트의 정보를 리턴하기 위해 parErrorObjectElementIndex 와 parErrorAttributeElementIndex 파라메터가 사용됩니다.

If the result of the CreateObjectGroup operation is FAIL, especially when the error occurs while creating specific attribute during the process of creating a specific object, then the parErrorObjectElementIndex and parErrorAttributeElementIndex parameters are used to return the information about the corresponding object and attribute.

즉, parErrorObjectElementIndex 파라메터의 값에는 오류가 발생한 애트리뷰트가 속한 오브젝트가 파라메터를 구성하는 몇 번째의 오브젝트인가를 나타내는 파라메터 입니다.

In other words, the parErrorObjectElementIndex parameter contains the index of the object that caused an error in the parameter that includes the corresponding object as an element.

13.19.5.4. parErrorAttributeElementIndex

CreateObjectGroup 오퍼레이션의 실행 결과가 실패일 경우, 오류가 발생한 오브젝트의 정보를 parErrorObjectElementIndex 파라메터에 기록하고, 해당 오브젝트의 몇 번째 애트리뷰트에서 오류가 발생했는가 하는 정보를 전달하기 위해 parErrorAttributeElementIndex 파라메터에 해당 정보를 기록합니다.

If the result of the CreateObjectGroup operation is FAIL, then the information of the object that caused an error is stored in the parErrorObjectElementIndex parameter and the index of the attribute of the corresponding object is stored in the parErrorAttributeElementIndex parameter.

13.19.5.5. Handling Procedure at the Receiver

수신측의 처리 절차는 상기 파라메터의 정의에서 정의한 것과 동일합니다. 즉, 파라메터로 주어진 오브젝트 리스트를 구성하는 각각의 엘리먼트들에 대해 하나씩 오브젝트를 생성합니다. 생성 과정이 성공적으로 완료되었을 경우에는 parResult 파라메터에 TRUE 값을 기록하여 매니저로 전송합니다. 그러나, 생성 과정 중에 오류가 발생하여 생성과정이 중단 되었을 경우에는 해당 오류의 종류 혹은 그 원인을 parOperationErrorCode 에 기록하고, 해당 오류가 발생한 엘리먼트 즉, 오브젝트가 몇 번째 오브젝트인가를 확인하여 그 값을 parErrorObjectElementIdnex 파라메터에 기록하여 매니저로 송신합니다.

The procedure at the receiver is the same as the one defined in the parameter as above. Objects corresponding to the elements of the object list provided as parameters are created one by one. If the creation process is successful, then the value of parResult parameter is set to TRUE and it is sent to the manager. If the creation process is interrupted due to an error, then the type or cause of error is stored in the parOperationErrorCode and the index of the object that caused an error in the list of elements is stored in the parErrorObjectElementIdnex parameter. They are then sent to the manager.

매니저는 OperationResponse 오퍼레이션을 수신하고 해당 parResult 파라메터의 값이 FALSE 인 경우에는 parOperationErrorCode 값을 확인하여 그 원인을 파악하고, parErrorObjectElementIdnex 파라메터의 값을 읽어 몇 번째 오브젝트의 생성과정에서 오류가 발생했는지를 파악합니다. 혹은 parOperationErrorCode 파라메터의 값과 parErrorElementIndex 파라메터의 값이 오브젝트의 생성을 위한 사전 준비단계 혹은 생성완료 후 정리단계에서 발생한 오류일 경우, 해당 상황에 적절한 내부 연산 혹은 다른 오퍼레이션의 호출 등의 작업을 진행합니다. 상기 과정은 애트리뷰트의 생성과정에 발생한 오류에 대해서도 동일하게 적용됩니다

The manager receives the OperationResponse operation and checks the value of the parResult parameter. If it is FALSE, then it checks the values of parOperationErrorCode to examine the cause and reads the value of the parErrorObjectElementIdnex parameter to find the object that caused the error. If the values of the parOperationErrorCode parameter and parErrorElementIndex parameter indicate that the error occurred in the preparation stage for creating the objects or in a stage after creating all objects, then it performs internal computation of calls other processes required for the situation. This procedure applies to the error that occurred while creating attributes in the same way.

13.20. DeleteObject operation

DeleteObject 오퍼레이션은 에이전트가 관리하고 있는 정보모델의 특정 오브젝트를 삭제하기 위해 사용합니다. 오브젝트를 삭제한다는 것은 구현 측면에서 바라볼 때 메모리에 로드되어 있는 클래스의 인스턴스를 삭제한다는 뜻입니다. 또한 해당 오브젝트가 갖고 있는 애트리뷰트의 인스턴스들도 모두 삭제하며, 해당 오브젝트와 포함관계에 있는 오브젝트들의 오브젝트와의 관계까지 모두 초기화 시킨다는 의미입니다. 해당 오퍼레이션이 실패 했다는 의미는 삭제해야 할 오브젝트의 삭제가 불가능한 경우입니다.

The DeleteObject operation is used to delete a specific object in the information model managed by the agent. Deleting an object means removing the instance of the class from the memory from the perspective of implementation. It also means that instances of all attributes of the class are also deleted, and relationships between objects that are related to the corresponding object should also be initialized. Failure of this operation means that the object to be deleted cannot be deleted.

13.20.1. Structure of Signature and Return Type

DeleteObject 오퍼레이션의 signature 와 리턴 타입의 구조는 다음 표 61-DeleteObject 오퍼레이션의 signature 와 리턴 타입의 구조 과 같습니다. 오브젝를 삭제하기 위해서는 해당 오브젝트의 클래스 아이디와 오브젝트 이름을 사용합니다. 해당 정보 이외의 다른 정보는 필요로 하지 않으며, 오브젝트의 삭제 중 발생하는 오류가 오브젝트 혹은 시스템 등에 기인할 경우 실패 signature OVLD 1 을 이용합니다. 오류발생의 원인이 해당 오브젝트의 애트리뷰트에 있을 경우에는 실패 signature OVLD 2 를 이용합니다.

The structure of signature and return type of the DeleteObject operation are shown in the 표 61-DeleteObject 오퍼레이션의 signature 와 리턴 타입의 구조. The class ID and object name are used to delete specific object. Other information besides them is not required, and FAIL signature OVLD 1 is used for an error that occurs while deleting an object. FAIL signature OVLD 2 is used if the error is caused by an attribute of the corresponding object.

표 61-DeleteObject 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of DeleteObject operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ClassIdentifier UM3ObjectName	parUM3ClassIdentifier parUM3ObjectName	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3
FAIL	UM3Boolean UM3OperationErrorCode UM3ClassIdentifier UM3ObjectName	parResult: FALSE parUM3OperationErrorCode parUM3AttributeClassIdentifier parUM3AttributeObjectName	OVLD 7

The REQUEST signature of the DeleteObject operation can be defined by using ASN.1 regular expressions as follows.

위의 DeleteObject 오퍼레이션의 요청 signature 의 ASN.1 정규표현식으로의 표현은 다음과 같습니다.

```
DeleteObject ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
```

&parUM3ObjectName UM3ObjectName
}

13.20.2. REQUEST signature

13.20.2.1. parUM3ClassIdentifier

This contains the class ID of the object to delete.

삭제하고자 하는 오브젝트의 클래스아이디를 나타냅니다.

13.20.2.2. parUM3ObjectName

This contains the object name of the object to delete.

삭제하고자 하는 오브젝트의 오브젝트 이름을 나타냅니다.

13.20.3. SUCCESS signature

13.20.3.1. parResult : TRUE

If the corresponding operation is a success, then the parameter parResult of UM3Boolean type is set to TRUE.

해당 오퍼레이션이 성공할 경우 UM3Boolean 타입의 파라메터 parResult 에 TRUE 값을 기록합니다.

13.20.4. FAIL signature OVLD 3

FAIL signature OVLD 3 returns the parameter of only the UM3OperationErrorCode type that indicates failure of operation without additional information when operation fails. This FAIL signature is used for both cases of the failure caused by the object or by other reasons besides the object.

실패 signature OVLD 3 은 오퍼레이션이 실패하였을 경우 수행 실패라는 결과와 그 원인을 나타내는 UM3OperationErrorCode 타입의 파라메터 만을 리턴합니다. 이는 해당 오퍼레이션의 실패 원인이 오브젝트가 아닌 다른 이유 때문이거나 혹은 오브젝트에 기인하는 등 2 가지의 경우 사용되는 실패 signature 입니다.

13.20.4.1. parResult : FALSE

This parameter is UM3Boolean type. It is set to the FAIL value upon failure of operation, and it is sent to the manager.

UM3Boolean 타입의 파라메터이며 오퍼레이션의 실패에 따른 FALSE 값을 기록하여 매니저로 전송합

니다.

13.20.4.2. parUM3OperationErrorCode

This parameter is UM3OperationErrorCode type, and it contains the code that indicates the cause of failure.

UM3OperationErrorCode 타입의 파라메터이며 실패의 원인을 나타내는 코드를 기록합니다.

13.20.5. FAIL signature OVLD 7

FAIL signature OVLD 7 signature is used when the failure is caused by an attribute.

실패 signature OVLD 7 는 오퍼레이션의 실패 원인이 애트리뷰트에 있을 경우 사용하는 signature 입니다.

13.20.5.1. parResult : FALSE

UM3Boolean 타입의 파라메터이며 오퍼레이션의 실패에 따른 FALSE 값을 기록하여 매니저로 전송합니다.

This parameter is UM3Boolean type. It is set to the FALSE value upon failure of operation, and it is sent to the manager.

13.20.5.2. parUM3OperationErrorCode

UM3OperationErrorCode 타입의 파라메터이며 실패의 원인을 나타내는 코드를 기록합니다.

This parameter is UM3OperationErrorCode type. This contains the code that indicates the cause of failure.

13.20.5.3. parUM3AttributeClassIdentifier

오퍼레이션이 실패한 원인이 애트리뷰트에 있을 경우 해당 애트리뷰트의 클래스아이디를 나타냅니다.

This contains the class of the attribute that caused the failure.

13.20.5.4. parUM3AttributeObjectName

오퍼레이션 실패의 원인을 제공한 애트리뷰트의 애트리뷰트 오브젝트 이름을 나타냅니다.

This contains the attribute object name of the attribute that caused failure.

13.20.6. Handling Procedure at the Receiver

해당 오퍼레이션을 수신한 에이전트는 주어진 parUM3ClassIdentifier 와 parUM3ObjectName 파라메터를 이용하여 해당 오브젝트를 정보모델 트리에서 찾아냅니다. 해당 오브젝트를 찾은 후 해당 오브젝트가 포함관계 혹은 aggregation relationship 관계로 갖고 있는 하위 오브젝트들이 있는지를 확인합니다. 만약 해당 오브젝트가 하위 오브젝트들을 갖고 있는 상황이라면 해당 오브젝트는 삭제가 불가능합니다. 이럴 경우 오퍼레이션 실패의 원인은 오브젝트에 있으며 실패 signature OVLD 3 이 OperationResponse 오퍼레이션의 signature 로 사용됩니다. 위와는 달리 하위 오브젝트가 없는 것으로 확인된다면 해당 오브젝트를 구성하는 모든 애트리뷰트 오브젝트 들을 삭제한 후, 해당 오브젝트를 삭제합니다. OperationResponse 오퍼레이션은 성공 signature 를 이용해 매니저에 오퍼레이션이 성공적으로 완료되었음을 통보합니다.

When the agent receives this operation, it searches for the corresponding object from the information model tree by using the provided parUM3ClassIdentifier and parUM3ObjectName parameters. When it is found, it checks whether there are lower level objects in the relationship or aggregation relationship with the corresponding object. If it has lower level objects, then it cannot be deleted. In this case, the cause of failure is the object and FAIL signature OVLD 3 is used as the signature of OperationResponse operation. Unlike this case, if there are no lower level objects, then it deletes all attribute objects of the corresponding object and deletes the object itself. The OperationResponse operation notifies the success of the operation to the manager by using the SUCCESS signature.

13.21. DeleteObjectGroup operation

DeleteObjectGroup 오퍼레이션은 1 개 이상 복수의 오브젝트를 삭제할 때 사용합니다. 매니저는 에이전트에게 DeleteObjectGroup 오퍼레이션을 통해 특정 오브젝트 들을 삭제토록 요청할 수 있으나 UM3-GET 서비스의 오퍼레이션처럼 조건문을 지정할 수는 없습니다. 즉, 삭제하고자 하는 오브젝트들을 명시적으로 요청 signature 의 파라메터에 기록하여 전송해야 합니다. 이는 잘못된 삭제 오퍼레이션에 의해 전체 서비스 시스템이 큰 오류를 일으키는 일을 미연에 방지하기 위함입니다.

The DeleteObjectGroup operation is used to delete multiple objects. The manager can request the agent to delete a specific object through the DeleteObjectGroup operation, but it cannot assign the conditional statements like the operations of the UM3-GET service. In other words, objects should be explicitly specified in the parameter of the REQUEST signature and the request has to be sent. The purpose is to prevent a fatal error to the system by an incorrect delete operation.

표 62-DeleteObject 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of DeleteObject operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ObjectList	parUM3ObjectList	UM3ObjectIndicator element
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16	parResult: FALSE parUM3OperationErrorCode parUM3ErrorObjectElementIndex	OVLD 4
FAIL	UM3Boolean UM3OperationErrorCode UM3UnsignedInteger16 UM3ClassIdentifier UM3ObjectName	parResult : FALSE parUM3OperationErrorCode parUM3ErrorObjectElementIndex parUM3AttributeClassIdentifier parUM3AttributeObjectName	OVLD 5

13.21.1. Structure of Signature and Return Type

DeleteObjectGroup 오퍼레이션은 DeleteObject 오퍼레이션과는 달리 복수의 오브젝트를 한 번에 삭제하기 위한 오퍼레이션입니다. DeleteObjectGroup 오퍼레이션의 signature 와 리턴 타입의 구조는 표 62-DeleteObject 오퍼레이션의 signature 와 리턴 타입의 구조 과 같습니다.

이러한 DeleteObjectGroup 오퍼레이션의 ASN.1 정규표현식으로의 표현은 다음과 같습니다.

Unlike the DeleteObject operation, the DeleteObjectGroup operation is used to delete multiple objects in one operation. The structure of Signature and Return Type of DeleteObjectGroup operation is shown in 표 62-DeleteObject 오퍼레이션의 signature 와 리턴 타입의 구조.

The DeleteObjectGroup operation can be defined by using ASN.1 regular expressions as follows.

```
DeleteObjectGroup ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName UNIQUE,
        &parUM3ObjectList                 UM3ObjectList
}
```

13.21.2. REQUEST signature

DeleteObjectGroup 오퍼레이션은 UM3ObjectList 타입의 파라메터를 요청 signature 의 파라메터로 사용합니다.

The DeleteObjectGroup operation uses the UM3ObjectList type parameter as the parameter for the REQUEST signature.

13.21.2.1. parUM3ObjectList

parUM3ObjectList 파라메터는 UM3ObjectList 타입의 파라메터이며 삭제하고자 하는 오브젝트들의 리스트가 기록되어야 합니다. 해당 리스트의 엘리먼트들은 UM3ObjectIndicator 타입이며 삭제대상 오브젝트들의 클래스아이디와 오브젝트 이름들이 기록되어 있습니다.

위에서 정의한 parUM3ObjectList 파라메터의 엘리먼트들은 UM3 SEQUENCE OF 타입으로 정의되어 있으므로 이들의 나열 순서는 의미 있는 순서들입니다. 만약 오퍼레이션이 실패할 경우에는 해당 순서를 기준으로 오류가 발생한 오브젝트의 정보를 리턴하게 됩니다.

The parUM3ObjectList parameter is UM3ObjectList type, and it contains the list of objects to delete. The elements of this list are UM3ObjectIndicator type, and the list contains the class ID and object name of the objects to delete.

The elements of the parUM3ObjectList parameter are UM3 SEQUENCE OF type, and the sequence of the list has significant meaning. If the operation fails, then the index of the object that caused an error is returned.

13.21.3. SUCCESS signature

13.21.3.1. parResult : TRUE

오퍼레이션의 수행이 성공적으로 완료되었음을 나타내는 파라메터이며 UM3Boolean 타입으로 정의됩니다. 그 default 값을 TRUE 로 기록하여 매니저로 전송합니다.

This parameter indicates the successful execution of operation and it is UM3Boolean type. Its default value is set to TRUE and it is returned to the manager.

13.21.4. FAIL signature OVLD 3

실패 signature OVLD 3 은 오퍼레이션의 수행이 실패한 이유가 parUM3ObjectList 파라메터로 주어진 오브젝트나 혹은 해당 오브젝트를 구성하고 있는 애트리뷰트에 있는 것이 아니라, 시스템 혹은 다른 이유로 인해 오퍼레이션이 실패하였음을 나타냅니다.

FAIL signature OVLD 3 indicates that the operation has failed not because of the object or attribute of the object provided as the parUM3ObjectList parameter, but because of the system or some other reasons.

13.21.4.1. parResult : FALSE

UM3Boolean 타입의 파라메터이며 그 값은 FALSE 로 지정되어 전송됩니다.

This parameter is UM3Boolean type, and the value is set to FALSE and the result is returned.

13.21.4.2. parUM3OperationErrorCode

오퍼레이션이 실패한 이유를 UM3OperationErrorCode 타입의 값으로 기록합니다.

The cause of failure is stored in the value of UM3OperationErrorCode type and the result is returned.

13.21.5. FAIL signature OVLD 4

해당 오퍼레이션이 실패한 이유가 특정 오브젝트에 있을 경우 사용하는 signature 입니다. 앞서 정의한 바와 같이 요청 signature 를 구성하고 있는 parUM3ObjectList 파라메터는 UM3 SEQUENCE OF 타입으로 정의되므로 순서대로 주어진 오브젝트들 중 몇 번째 오브젝트에서 오류가 발생했는가를 parUM3ErrorObjectElementIndex 파라메터에 기록하여 전송하게 됩니다.

This signature is used when the cause of failure of the corresponding operation is a specific object. Since the parUM3ObjectList parameter of the REQUEST signature is UM3 SEQUENCE OF type, the index of the object that caused an error is stored in the parUM3ErrorObjectElementIndex parameter and the result is returned.

13.21.5.1. parResult : FALSE

UM3Boolean 타입의 파라메터이며 그 값은 FALSE 로 기록되어 전송됩니다.

The parameter is UM3Boolean type. Its value is set to FALSE, and the result is returned.

13.21.5.2. parUM3OperationErrorCode

UM3OperationErrorCode 타입의 파라메터이며 오퍼레이션이 실패한 원인을 나타내는 코드를 기록하게 됩니다.

This parameter is UM3OperationErrorCode type, and it contains the cause of failure of the operation.

13.21.5.3. parUM3ErrorObjectElementIndex

요청 signature 를 구성하는 parUM3ObjectList 파라메터에 주어진 엘리먼트들 중 몇 번째 엘리먼트의 오브젝트에서 오류가 발생했는가를 나타내는 인덱스 정보입니다. 해당 파라메터의 타입은 UM3UnsignedInteger16 타입입니다.

This is index information that points to the object that caused an error among the elements in the parUM3ObjectList parameter of the REQUEST signature. Its type is UM3UnsignedInteger16.

13.21.6. FAIL signature OVLD 5

실패 signature OVLD 3 은 특정 애트리뷰트가 오퍼레이션 실패의 원인을 제공하였을 경우 사용하는 OperationResponse 오퍼레이션의 signature 입니다.

FAIL signature OVLD 3 of OperationResponse operation is used when a specific attribute causes an error.

13.21.6.1. parResult : FALSE

UM3Boolean 타입의 파라메터이며 그 값은 FALSE 로 기록되어 매니저로 전송됩니다.

This parameter is UM3Boolean type, and its value is set to FALSE and the result is returned.

13.21.6.2. parUM3OperationErrorCode

UM3OperationErrorCode 타입의 파라메터이며 오퍼레이션이 실패한 원인을 나타내는 코드를 기록하게 됩니다.

This parameter is UM3OperationErrorCode type, and it contains the code that indicates the cause of failure of the operation.

13.21.6.3. parUM3ErrorObjectElementIndex

UM3UnsignedInteger16 타입의 파라메터입니다. 요청 signature 를 구성하는 parUM3ObjectList 파라메터에 주어진 엘리먼트들 중 몇 번째 엘리먼트가 가리키는 오브젝트에 속한 애트리뷰트에서 오류가 발생했는가를 나타내는 인덱스 정보입니다.

This parameter is UM3UnsignedInteger16 type. It is index information that points to a specific attribute of the object that caused an error among the elements of the parUM3ObjectList parameter of the REQUEST signature.

13.21.6.4. parUM3AttributeClassIdentifier

오류가 발생한 애트리뷰트의 클래스 아이디를 기록합니다.

This contains the class ID of attribute that caused an error.

13.21.6.5. parUM3AttributeObjectName

오류가 발생한 애트리뷰트의 애트리뷰트 오브젝트 이름을 기록합니다.

This contains the attribute object name of the attribute that caused an error.

13.21.7. Handling Procedure at the Receiver

에이전트는 매니저가 전송한 DeleteObjectGroup 오퍼레이션을 수신한 후 해당 오퍼레이션의 파라메터로 주어진 parUM3ObjectList 파라메터의 엘리먼트로 기록된 UM3ClassIdentifier 와 UM3ObjectName 타입의 오브젝트를 검색합니다. 해당 오브젝트를 찾은 후의 오퍼레이션 수행 절차는 DeleteObject 오퍼레이션과 동일하게 진행됩니다.

When agent receives the DeleteObjectGroup operation, it searches for the object of UM3ClassIdentifier and UM3ObjectName type from the elements of the parUM3ObjectList parameter of the corresponding operation. When it is found, the remaining procedure is the same as the DeleteObject operation.

그림 23-DeleteObjectGroup 오퍼레이션의 parUM3ObjectList 파라메터의 올바른 구성 [은 위의 오퍼레이션 수행 과정과 앞서 정의한 DeleteObject 오퍼레이션의 수행 과정에서 정의한 aggregation relationship 관계에 있는 오브젝트들 간의 삭제에 관한 예를 보여주고 있습니다. 그림 23-DeleteObjectGroup 오퍼레이션의 parUM3ObjectList 파라메터의 올바른 구성 [의 중앙에 표시한 에이전트가 관리하는 정보모델 트리에는 secondParam 오브젝트가 그 하위 오브젝트들로 fifthParam 과 sixthParam 들을 aggregation relationship 관계로 갖고 있음을 보여주고 있습니다. 이러한 aggregation relationship 관계가 형성되어 있을 경우 하위 오브젝트 들에 대한 삭제과정 없이 상위 오브젝트를 삭제하려 할 때 에이전트는 반드시 이를 오류로 처리할 수 있어야 합니다. 그림 23-DeleteObjectGroup 오퍼레이션의 parUM3ObjectList 파라메터의 올바른 구성 [의 오른편에는 올바르게 구성된 parUM3ObjectList 파라메터의 구성을 나타내었

습니다.

그림 23-DeleteObjectGroup 오퍼레이션의 parUM3ObjectList 파라메터의 올바른 구성 [is an example of the delete operation and deletion of objects in the aggregation relationship as defined earlier in the DeleteObject operation. Fig. 23 shows that the secondParam object has the aggregation relationship with the fifthParam and sixthParam in the lower level in the information model tree managed by the agent as shown at the center. When there is aggregation relationship like this, an attempt to delete an upper level object without deleting lower level objects causes an error and the agent should be able to handle it as an exception. The correctly configured parUM3ObjectList parameter is shown on the right side of 그림 23-DeleteObjectGroup 오퍼레이션의 parUM3ObjectList 파라메터의 올바른 구성 [.

13.22. CancelGetObjectValue operation

CancelGetObjectValue 오퍼레이션은 GetObjectValue 오퍼레이션을 요청한 매니저가 에이전트에게 GetObjectValue 오퍼레이션을 취소하기 위해 사용하는 오퍼레이션입니다. 앞서 정의한 connection oriented UM3 session 의 형태로 구현된 시스템의 경우에는 별도의 오퍼레이션 요청을 받아들여 수행중인 오퍼레이션에 대한 수행을 중단하여야 합니다.

The CancelGetObjectValue operation is used for the manager that requested the GetObjectValue operation to request the agent to cancel the GetObjectValue operation. In case of a system that implemented the protocol with a connection oriented UM3 session as defined earlier, a separate request should be received to cancel the currently active operation.

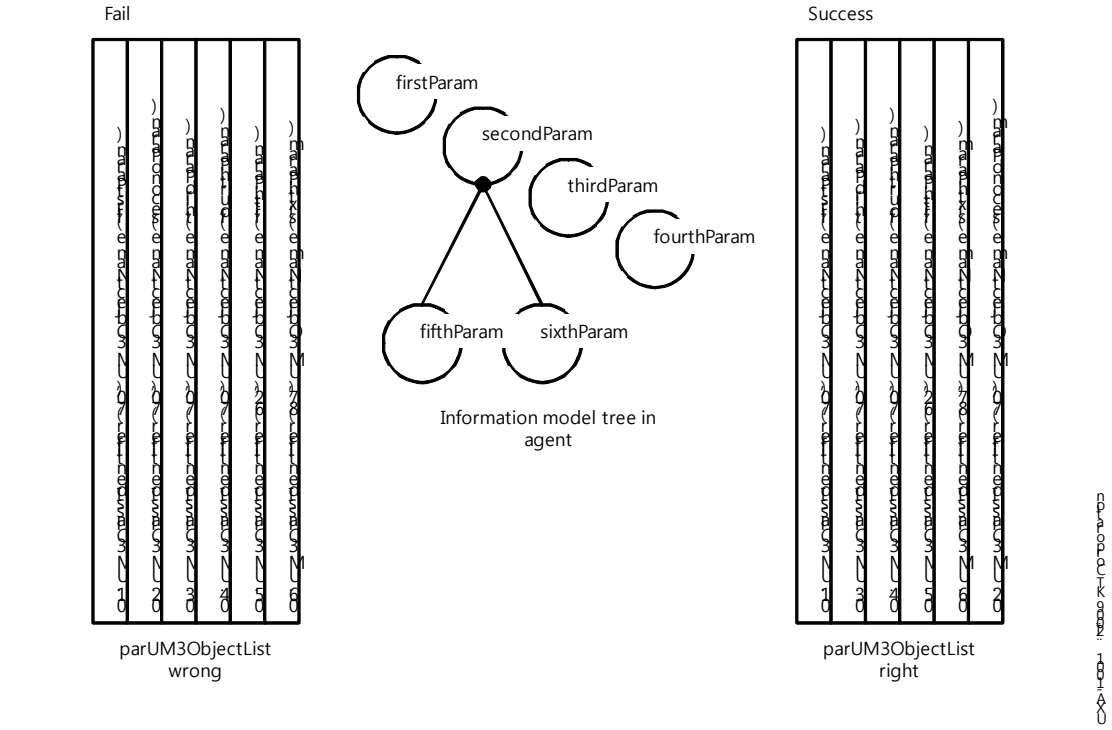


그림 23-DeleteObjectGroup 오퍼레이션의 parUM3ObjectList 파라메터의 올바른 구성
[Correct configuration of parUM3ObjectList parameter of DeleteObjectGroup operation]

13.22.1. Structure of Signature and Return Type

CancelGetObjectValue 오퍼레이션의 signature 와 리턴 타입의 구조는 아래와 같습니다.

The Structure of Signature and Return Type of the CancelGetObjectValue operation are as follows.

표 63-CancelGetObjectValue 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of CancelGetObjectValue operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ObjectName	parUM3OperationToBeCancelled	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean	parResult: FALSE	OVLD 3

UM3OperationErrorCode	parUM3OperationErrorCode

표 63-CancelGetObjectValue 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 요청 signature 는 한 개의 파라메터 만이 주어지며, 성공 signature 또한 한 개의 파라메터 만이 주어집니다. 오퍼레이션이 실패했을 경우에는 해당 실패의 원인을 UM3OperationErrorCode 타입의 파라메터에 기록하여 전송합니다.

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

As shown in Table 62, the REQUEST signature has only one parameter, and the SUCCESS signature also has only one parameter. If the operation fails, then the cause of failure is stored in the parameter of UM3OperationErrorCode type and the result is returned.

The following is the definition of this operation by using ASN.1 regular expressions.

```
CancelGetObjectValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled     UM3ObjectName
}
```

상기 ASN.1 정규표현식에서 parUM3OperationToBeCancelled 파라메터는 취소하고자 하는 UM3-GET 오퍼레이션의 세션 아이디를 나타냅니다.

In the above ASN.1 regular expression, the parUM3OperationToBeCancelled parameter contains the session ID of the UM3-GET operation to cancel.

13.22.2. REQUEST signature

13.22.2.1. parUM3OperationToBeCancelled

CancelGetObjectValue 오퍼레이션이 적용되어야 할 즉, 취소되어야 할 GetObjectValue 오퍼레이션의 세션 아이디가 parUM3OperationToBeCancelled 파라메터의 값으로 주어져야 합니다.

The session ID of the operation to apply CancelGetObjectValue operation, i.e. GetObjectValue operation to cancel, should be stored as the value of the parUM3OperationToBeCancelled parameter.

13.22.3. SUCCESS signature

13.22.3.1. parResult : TRUE

취소 오퍼레이션이 성공하였을 경우 parResult 파라메터에는 TRUE 값이 기록되어 매니저로 전송됩니다.

If cancel operation is a success, then the parResult parameter is set to TRUE and it is returned to the manager.

13.22.4. FAIL signature OVLD 3

13.22.4.1. parResult : FALSE

취소 오퍼레이션이 실패하였을 경우 parResult 파라메터에는 FALSE 값이 기록되어 매니저로 전송됩니다.

If cancel operation fails, then the parResult parameter is set to FALSE and it is returned to the manager.

13.22.4.2. parUM3OperationErrorCode

parUM3OperationErrorCode 파라메터에는 오퍼레이션이 실패한 이유가 기록되어 매니저로 전송됩니다.

The cause of failure of operation is stored in the parUM3OperationErrorCode parameter, and it is returned to the manager.

13.22.5. Handling Procedure at the Receiver

에이전트는 매니저로부터 요청되는 GetObjectValue 오퍼레이션과 더불어 UM3-GET 서비스에 속하는 모든 오퍼레이션들에 대해 해당 오퍼레이션이 종료되는 순간까지 해당 오퍼레이션의 세션 아이디 값을 저장관리할 수 있어야 합니다. 즉, 장시간에 걸쳐 수행되는 UM3-GET 서비스의 오퍼레이션들은 언제라도 매니저에 의해 취소될 수 있어야 하며, 해당 오퍼레이션에 대한 구분은 매니저가 생성하여 에이전트로 전송하는 um3OperationObjectName 값이 저장된 오퍼레이션 APDU 의 N 필드의 값을 기준으로 이루져야 합니다.

The agent should be able to store and manage the session ID of the GetObjectValue operation requested by the manager along with all other operations of the UM3-GET service until the corresponding session is terminated. In other words, the manager should be able to cancel the operations of the UM3-GET service that is active for an extended period of time at any time when required, and the identification of the corresponding operation should be performed by using the value of the N field for the APDU operation that has the um3OperationObjectName value that is created by the manager and sent to agent.

에이전트가 매니저의 요청에 의해 UM3-GET 서비스의 오퍼레이션의 수행을 이미 종료하고 OperationResponse 오퍼레이션을 송신한 상태라면, 매니저의 CancelGetObjectValue 오퍼레이션에 대한 처리 결과는 실패 signature 를 이용하여 응답 APDU 를 보냅니다. 결국 아주 장시간에 걸쳐 수행되는 UM3-GET 서비스 관련 오퍼레이션이 아닌 이상 거의 대부분의 오퍼레이션은 취소를 하기전에 이미 오퍼레이션이 종료된 상태일 확률이 높습니다.

When the agent completes the operation of the UM3-GET service requested by the manager and sends the OperationResponse operation, the result of the CancelGetObjectValue operation should be sent as response APDU by using the FAIL signature. Most of the time, the operation is likely to be completed before cancelling the operation except for UM3-GET service related operations take long periods of time.

오퍼레이션의 취소를 확인하는 방법은 여러가지가 있을 수 있으나 그 중 한가지 방법은 모든 UM3-GET 서비스 오퍼레이션이 OperationResponse 오퍼레이션을 호출하여 매니저에게 응답 APDU 를 보내기 직전에 해당 오퍼레이션에 대한 취소 오퍼레이션이 요청되었는지를 확인하는 방법이 있습니다. 그러나, 이러한 방법은 불필요한 시스템의 구동을 중단할 수 없을 뿐만 아니라, 시스템의 부하를 크게 늘리는 부작용이 있을 수 있음에 유의해야 합니다.

There is more than one way of checking the cancellation of an operation, and one of them is to check the whether the cancellation operation for the corresponding operation was requested before sending the response APDU to the manager by calling the OperationResponse operation. Caution is required for this method because it cannot stop the unnecessary system drive: the side effect is increased system loading.

Connectionless UM3 session 의 형태로 매니저와 에이전트가 통신을 하고 있다면, CancelGetObjectValue 오퍼레이션을 요청한 매니저는 더 이상 자신이 요청한 GetObjectValue 오퍼레이션에 대한 에이전트의 응답을 기다리지 않아야 합니다.

If the manager and agent communicate through a connectionless UM3 session, then the manager requesting the CancelGetObjectValue operation should not wait for the response from the agent.

13.23. CancelGetObjectGroupValue operation

CancelGetObjectGroupValue 오퍼레이션의 signature 와 리턴 타입은 CancelGetObjectValue 오퍼레이션과 동일한 형태로 정의됩니다.

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

The signature and return type of the CancelGetObjectGroupValue operation are the same as the CancelGetObjectValue operation.

This operation can be defined by using ASN.1 regular expressions as follows.

```
CancelGetObjectGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled       UM3ObjectName
}
```

13.24. CancelGetObjectAttributeValue operation

CancelGetObjectAttributeValue 오퍼레이션의 signature 와 리턴 타입은 CancelGetObjectValue 오퍼레이션과 동일한 형태로 정의됩니다.

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

The signature and return type of the CancelGetObjectAttributeValue operation are the same as the CancelGetObjectValue operation.

This operation can be defined by using ASN.1 regular expressions as follows.

```
CancelGetObjectAttributeValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled       UM3ObjectName
}
```

13.25. CancelGetObjectAttributeGroupValue operation

CancelGetObjectAttributeGroupValue 오퍼레이션의 signature 와 리턴 타입은 CancelGetObjectValue 오퍼레이션과 동일한 형태로 정의됩니다.

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

The signature and return type of the CancelGetObjectAttributeGroupValue operation are the same as the CancelGetObjectValue operation.

This operation can be defined by using ASN.1 regular expressions as follows.

```
CancelGetObjectAttributeGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled       UM3ObjectName
}
```

13.26. RequestChangeOfAttributeValueReport operation

RequestChangeOfAttributeValueReport 오퍼레이션은 특정 애트리뷰트의 값이 변화할 때 해당 이벤트를 매니저로 자동으로 보고토록 설정하는 오퍼레이션입니다. 이는 앞의 UM3-GET 서비스의 오퍼레이션들이 갖는 polling 방식의 정보수집 방식과는 달리, event driven 형식의 데이터 수집 방식입니다.

The RequestChangeOfAttributeValueReport operation is used to establish automatic reporting of the event for changes in the value of a specific attribute to the manager. Unlike the polling type information collection method of other operations of the UM3-GET service, it is an event-driven data collection method.

앞서 정의한 바와 같이 해당 오퍼레이션에 대한 응답은 OperationResponse 오퍼레이션이 아닌 ChangeOfAttributeValueReport 오퍼레이션을 통해 이루어집니다. 에이전트는 RequestChangeOfAttributeValueReport 오퍼레이션에 대한 정상수신 여부 혹은 인지 여부를 나타내는 ACK 를 UM3Ack 파라메터를 갖는 OperationResponse 오퍼레이션을 이용해 매니저로 전송합니다. 이는 일반적인 OperationResponse 오퍼레이션의 성공 signature 즉, parResult 파라메터의 값을 TRUE 로 설정하여 전송하는 것과 동일합니다. 이후 에이전트는 지정된 애트리뷰트의 값이 변화하는 경우 ChangeOfAttributeValueReport 오퍼레이션을 이용해 해당 이벤트의 발생을 매니저로 보고하게 됩니다.

As defined earlier, the response to the corresponding operation is performed through the ChangeOfAttributeValueReport operation instead of the OperationResponse operation. The agent sends ACK that indicates the status or acknowledgement of the correct reception for the RequestChangeOfAttributeValueReport operation to the manager by using the OperationResponse operation that has the UM3Ack parameter. This is the same as using the general method of using the SUCCESS signature of the OperationResponse operation i.e. setting the parResult parameter value to TRUE. Afterwards, the agent reports the event to the manager by using the ChangeOfAttributeValueReport operation when the value of the specified attribute changes.

RequestChangeOfAttributeValueReport 오퍼레이션은 위에서 정의한 바와 같이 특정 애트리뷰트에 대해 그 값의 범위 혹은 임계치 (threshold value) 등의 조건에 관계 없이 해당 애트리뷰트의 값이 변화할 경우 무조건 이벤트를 발생시켜 이를 매니저로 통보하기 위한 오퍼레이션입니다. 따라서, 오퍼레이션의 파라메터로 주어지는 UM3ActionBroker 오브젝트의 애트리뷰트인 targetCondition 애트리뷰트의 하위 애트리뷰트 condition 은 항상 COV 를 그 값으로 갖고 있어야 함에 유의해야 합니다. 해당 오퍼레이션을 수신하는 에이전트는 반드시 condition 애트리뷰트의 값이 COV 로 설정되어 있는가를 확인하고 그 값의 정상여부에 따라 다음 단계의 액션을 취해야 합니다.

The RequestChangeOfAttributeValueReport operation is used to generate the event when there is a change in the value of a specific attribute without setting conditions like the range of threshold and notifies the event to the manager. Therefore, caution is required to make sure that the lower level attribute condition of the targetCondition attribute, one of attributes of the UM3ActionBroker object provided as a parameter of the operation, should always have COV as its value. When the agent receives this operation, it should check whether the value of condition

attribute is set to COV and perform the next actions based on this value.

13.26.1. Structure of Signature and Return Type

RequestChangeOfAttributeValueReport 오퍼레이션은 애트리뷰트 값의 변화에 대한 이벤트 리포트를 요청하는 오퍼레이션입니다. 따라서, 대상이 되는 애트리뷰트와 해당 애트리뷰트가 속한 오브젝트의 정보가 필요합니다. 또한 애트리뷰트 값의 변화여부를 측정하기 위한 측정 주기 등의 정보도 필요합니다.

RequestChangeOfAttributeValueReport 오퍼레이션의 signature 와 리턴 타입의 구조는 아래와 같습니다.

The RequestChangeOfAttributeValueReport operation is used to request the event report for the changes in the value of the attribute. Therefore, it requires the information of the attribute and the object that has the corresponding attribute. It also requires other information like the measurement period to measure the change in the value of the attribute.

The structure of signature and return type of the RequestChangeOfAttributeValueReport operation are as follows.

표 64-RequestChangeOfAttributeValueReport 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of RequestChangeOfAttributeValueReport operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ActionBroker	parUM3ActionBroker	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3

표 64-RequestChangeOfAttributeValueReport 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 요청 signature 는 한 개의 파라메터 만이 주어지며, 성공 signature 또한 한 개의 파라메터 만이 주어집니다. 오퍼레이션이 실패했을 경우에는 해당 실패의 원인을 UM3OperationErrorCode 타입의 파라메터에 기록하여 전송합니다.

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

As shown in Table 63, the REQUEST signature has only one parameter and the SUCCESS signature also has only one parameter. If the operation fails, then the cause of failure is stored in the UM3OperationErrorCode type parameter and the result is returned.

This operation can be defined by using ASN.1 regular expressions as follows.

```
RequestChangeOfAttributeValueReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ActionBroker                 UM3ActionBroker
}
```

13.26.2. REQUEST signature

요청 signature 는 UM3ActionBroker 클래스 타입의 parUM3ActionBroker 파라메터가 주어집니다.

The REQUEST signature has a parUM3ActionBroker parameter of UM3ActionBroker class type.

13.26.2.1. parUM3ActionBroker

UM3ActionBroker 클래스 타입의 오브젝트가 파라메터의 형태로 주어집니다. 해당 오브젝트는 모든 애트리뷰트이 값이 채워진 완전한 형태로 주어져야 합니다. 특히 parUM3ActionBroker 파라메터 오브젝트의 애트리뷰트로 정의되어 있는 UM3ObjectValueCondition 타입의 targetCondition 애트리뷰트 오브젝트의 애트리뷰트인 condition 애트리뷰트의 값은 COV 로 지정되어야 합니다. 즉, 타겟 애트리뷰트의 값의 범위에 상관없이 그 값이 변화할 경우 무조건 이벤트를 보고해야 한다는 뜻입니다.

The object of UM3ActionBroker class type is provided as a parameter, and the corresponding object should be the complete form in which all attribute values are filled. The value of the condition attribute that is an attribute of the targetCondition attribute object of UM3ObjectValueCondition type, defined as an attribute of parUM3ActionBroker parameter object, should be set to COV. In other words, when the values changes, its event should be reported independently from the range of values of target attributes.

13.26.3. SUCCESS signature

13.26.3.1. parResult : TRUE

RequestChangeOfAttributeValueResponse 오퍼레이션의 요청을 정상적으로 접수하였다는 UM3Ack 의 signature 입니다. 즉, 해당 오퍼레이션의 요청을 받아 이벤트 감시를 정상적으로 시작하였다는 ACK 신호입니다.

This is a signature of UM3Ack that indicates that the request of the RequestChangeOfAttributeValueResponse operation is received correctly. In other words, it is an ACK signal that indicates that the event monitoring has started normally based on the request of the corresponding operation.

13.26.4. FAIL signature OVLD 3

정상적인 동작이 불가능한 경우 사용하는 signature 입니다. 경우에 따라서는 애트리뷰트의 이벤트 검출을 지원하지 않는 경우가 있을 수 있습니다. 즉, polling 방식의 이벤트 검출만을 지원하는 경우도 있으며 이런경우에는 eventDrivenNotSupport 에러가 기록되어 리턴되게 됩니다.

이와 더불어 매니저가 전송한 오퍼레이션 파라메터의 값이 부적절하게 정의된 경우 즉, 앞서 정의한 UM3ObjectValueCondition . targetCondition . condition 애트리뷰트의 값이 COV 로 지정되어 있지 않은 경우도 실패의 원인으로 분류하고 그 결과를 리턴해야 합니다.

This signature is used when normal operation is not possible. Event detection of the attribute may not be supported depending on the case, and only polling type event detection may be supported. In this case, an eventDrivenNotSupport error is flagged and the result is returned.

If the value of the operation parameter sent by manager is improperly defined i.e. if the value of the UM3ObjectValueCondition.targetCondition.condition attribute is not set to COV, then it is classified as a failure and that result should be returned.

13.26.4.1. parResult : FALSE

오퍼레이션의 요청을 실행할 수 없을 경우 사용하는 signature 입니다. parResult 파라메터에는 FALSE 값이 기록되어 전송됩니다.

This signature is used when the request of the operation cannot be executed. The parResult parameter value should be set to FALSE, and the result is returned.

13.26.4.2. parUM3OperationErrorCode

오퍼레이션이 실패한 원인을 기록합니다.

This contains the cause of failure of operation.

13.26.5. Handling Procedure at the Receiver

해당 오퍼레이션을 수신한 에이전트는 이벤트 검출 및 해당 이벤트의 통지 혹은 보고를 위해 다음과 같은 절차를 수행합니다.

The agent receiving the corresponding operation should perform the following procedure to detect the events and notify or report the corresponding event.

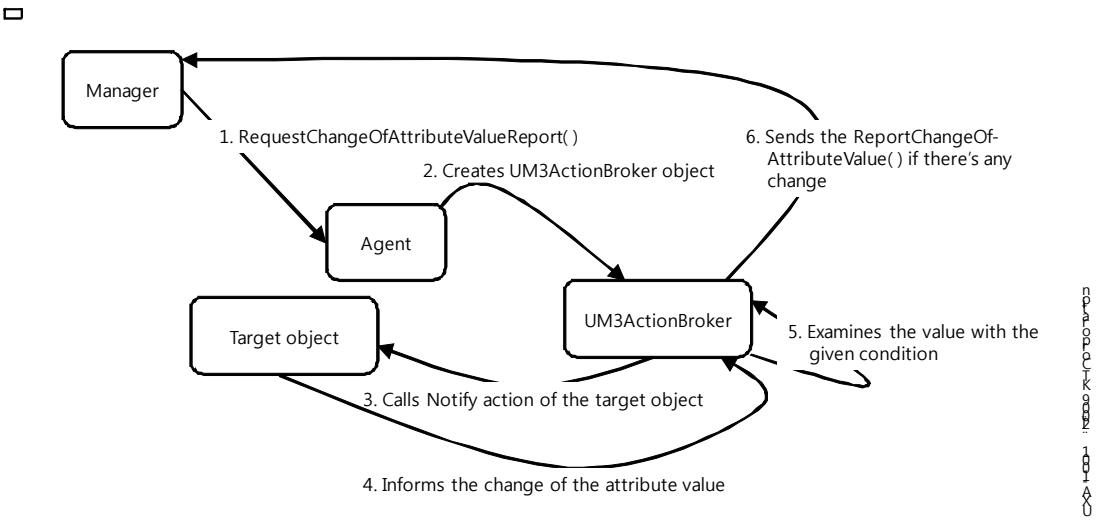


그림 24-RequestChangeOfAttributeValueReport 오퍼레이션의 처리 절차
[Handling procedure of RequestChangeOfAttributeValueReport operation]

우선 RequestChangeOfAttributeValueReport 오퍼레이션의 파라메터로 주어진 UM3ActionBroker 클래스 타입의 오브젝트를 생성합니다. 즉, 파라메터로 주어진 값을 이용하여 오브젝트 인스턴스를 생성하고 해당 오브젝트 인스턴스의 Notify 액션을 기동시킵니다. UM3ActionBroker 클래스는 UM3Base 클래스로부터 상속받은 Notify 액션을 override 하여 UM3Base 클래스와는 다른 형태의 액션을 수행합니다.

Firstly, the object of UM3ActionBroker class type provided as a parameter of the RequestChangeOfAttributeValueReport operation is created. The object instance is then created by using the given values of the parameter, and the Notify action of the corresponding object instance is launched. The Notify action inherited from the UM3Base class in overridden in the UM3ActionBroker class, and the performed action is different from the UM3Base class.

즉, UM3ActionBroker 클래스의 Notify 액션은 타겟으로 지정된 오브젝트의 Notify 액션을 호출하며, 타겟 오브젝트의 Notify 액션은 앞서 UM3Base 클래스에서 정의한 바와 같이 특정 타겟 애트리뷰트의 값이 변화할 경우 이를 UM3ActionBroker 오브젝트로 통보합니다.

In other words, the Notify action of the UM3ActionBroker class calls the Notify action of the specified object, and the Notify action of the target object notifies the changes in the value of target attribute as defined in the UM3Base class earlier to the UM3ActionBroker object.

타겟 애트리뷰트 값의 변화를 통보받은 UM3ActionBroker 오브젝트의 Notify 액션은 해당 애트리뷰트

의 값을 점검하고 조건을 만족할 경우 이를 ChangeOfAttributeValueReport 오퍼레이션을 통해 매니저로 통보하게 됩니다.

When the Notify action of the UM3ActionBroker object receives the notification of the changes of values in target attribute, it checks the value of the corresponding attribute and tests the conditions. If the conditions are satisfied, then it is notified to the manager through the ChangeOfAttributeValueReport operation.

13.27. RequestConditionDetectedReport operation

RequestConditionDetectedReport 오퍼레이션은 앞서 정의한 RequestChangeOfAttributeValueReport 오퍼레이션과 동일한 요청 signature 와 리턴 타입으로 구성됩니다. 단, RequestChangeOfAttributeValueReport 오퍼레이션과는 달리 parUM3ActionBroker.targetCondition.condition 애트리뷰트의 값은 COV 가 아닌 다른 조건 identifier 가 주어져야 합니다. 즉, 주어진 조건에 일치하는 타겟 애트리뷰트 값의 변화가 발생할 때만 매니저로 이벤트를 통보하는 형식입니다.

The RequestConditionDetectedReport operation has the REQUEST signature and the same return type as the RequestChangeOfAttributeValueReport operation that was defined earlier. Unlike the RequestChangeOfAttributeValueReport operation, the value of the parUM3ActionBroker.targetCondition.condition attribute should be set by the condition identifier instead of COV. In other words, it should be configured so that the event is only notified when there are changes in the value of the target attribute that meets the given conditions.

13.27.1. Structure of Signature and Return Type

RequestConditionDetectedReport 오퍼레이션은 애트리뷰트 값의 변화가 발생하고 그 변화한 값이 특정 조건에 부합하는 경우, 이에 대한 이벤트 리포트를 요청하는 오퍼레이션입니다. 따라서, 앞서 정의한 이벤트 리포트 관련 오퍼레이션들과 같이 대상이 되는 애트리뷰트와 해당 애트리뷰트가 속한 오브젝트의 정보가 필요합니다. 또한 이벤트를 감지하기 위한 조건과 값의 변화여부를 측정하기 위한 주기 등의 정보도 필요합니다.

The RequestConditionDetectedReport operation is used to request the event report when there are changes in the value of attribute and the changed values meet specific conditions. Like the operations related to the event report defined earlier, the information of the target attribute and object that has the corresponding attribute is required. Other information like the condition to detect the event and the period to measure the changes is also required.

RequestConditionDetectedReport 오퍼레이션은 UM3ActionBroker 타입의 parUM3ActionBroker 파라메터에 상기 필요한 정보들을 기록하여 에이전트로 전송합니다.

The RequestConditionDetectedReport operation stores the required information in the parUM3ActionBroker parameter of UM3ActionBroker type, and it is sent to the agent.

표 65-RequestConditionDetectedReport 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 요청 signature 는 한 개의 파라메터 만이 주어지며, 성공 signature 또한 한 개의 파라메터 만이 주어집니다. 오퍼레이션이 실패했을 경우에는 해당 실패의 원인을 UM3OperationErrorCode 타입의 파라메터에 기록하여 전송합니다.

As shown in Table 64, the REQUEST signature has only one parameter, and the SUCCESS signature also has only one parameter. If the operation fails, then the cause of failure is stored in the UM3OperationErrorCode type parameter and it is transmitted.

표 65-RequestConditionDetectedReport 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of RequestConditionDetectedReport operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ActionBroker	parUM3ActionBroker	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

This operation can be defined by using ASN.1 regular expressions as follows.

```
RequestConditionDetectedReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName UNIQUE,
        &parUM3ActionBroker               UM3ActionBroker
}
```

13.27.2. REQUEST signature

요청 signature 는 UM3ActionBroker 클래스 타입의 parUM3ActionBroker 파라메터가 주어집니다.

The REQUEST signature has a parUM3ActionBroker parameter of UM3ActionBroker class type.

13.27.2.1. parUM3ActionBroker

UM3ActionBroker 클래스 타입의 오브젝트가 파라메터의 형태로 주어집니다. 해당 오브젝트는 모든 애트리뷰트이 값이 채워진 완전한 형태로 주어져야 합니다.

The object of UM3ActionBroker class type is provided as a parameter. The corresponding object should be provided with all attributes completely filled up.

13.27.3. SUCCESS signature

13.27.3.1. parResult : TRUE

RequestConditionDetectedReport 오퍼레이션의 요청을 정상적으로 접수하였다는 UM3Ack 의 signature 입니다. 즉, 해당 오퍼레이션의 요청을 받아 이벤트 감시를 정상적으로 시작하였다는 ACK 신호입니다.

This is a signature of UM3Ack that indicates normal reception of the RequestConditionDetectedReport operation. In other words, it is an ACK signal that indicates that the event monitoring has normally started upon the request of the corresponding operation.

13.27.4. FAIL signature OVLD 3

정상적인 동작이 불가능한 경우 사용하는 signature 입니다. 경우에 따라서는 애트리뷰트의 이벤트 검출을 지원하지 않는 경우가 있을 수 있습니다. 즉, polling 방식의 이벤트 검출만을 지원하는 경우도 있으며 이런경우에는 eventDrivenNotSupport 에러가 기록되어 리턴되게 됩니다.

This signature is used when normal operation is not possible. Event detection of the attribute may not be supported depending on the case, and only polling type event detection may be supported. In this case, an eventDrivenNotSupport error is flagged and the result is returned.

13.27.4.1. parResult : FALSE

오퍼레이션의 요청을 실행할 수 없을 경우 사용하는 signature 입니다. parResult 파라메터에는 FALSE 값이 기록되어 전송됩니다.

This signature is used when the request of the operation cannot be executed. The parResult parameter value should be set to FALSE, and the result is returned.

13.27.4.2. parUM3OperationErrorCode

오퍼레이션이 실패한 원인을 기록합니다.

This contains the cause of failure of operation.

13.27.5. Handling Procedure at the Receiver

해당 오퍼레이션을 수신한 에이전트는 이벤트 검출 및 해당 이벤트의 통지 혹은 보고를 위해 RequestChangeOfAttributeValueReport 오퍼레이션과 동일한 다음과 같은 절차를 수행합니다.

The agent receiving the corresponding operation should perform the following procedure to detect the events and notify or report the corresponding event in the same way as the RequestChangeOfAttributeValueReport operation.

우선 RequestConditionDetectedReport 오퍼레이션의 파라메터로 주어진 UM3ActionBroker 클래스 타입의 오브젝트를 생성합니다. 즉, 파라메터로 주어진 값을 이용하여 오브젝트 인스턴스를 생성하고 해당 오브젝트 인스턴스의 Notify 액션을 기동시킵니다. UM3ActionBroker 클래스는 UM3Base 클래스로부터 상속받은 Notify 액션을 override 하여 UM3Base 클래스와는 다른 형태의 액션을 수행합니다. 즉, UM3ActionBroker 클래스의 Notify 액션은 타겟으로 지정된 오브젝트의 Notify 액션을 호출하며, 타겟 오브젝트의 Notify 액션은 앞서 UM3Base 클래스에서 정의한 바와 같이 특정 타겟 애트리뷰트의 값이 변화할 경우 이를 UM3ActionBroker 오브젝트로 통보합니다. 타겟 애트리뷰트 값의 변화를 통보받은 UM3ActionBroker 오브젝트의 Notify 액션은 해당 애트리뷰트의 값을 점검하고 조건을 만족할 경우 이를 ConditionDetectedReport 오퍼레이션을 통해 매니저로 통보하게 됩니다.

Firstly, the object of UM3ActionBroker class type provided as a parameter of the RequestChangeOfAttributeValueReport operation is created. Then, the object instance is created by using the given values of the parameter, and the Notify action of the corresponding object instance is launched. The Notify action inherited from the UM3Base class is overridden in the UM3ActionBroker class, and the performed action is different from the UM3Base class. In other words, the Notify action of the UM3ActionBroker class calls the Notify action of the specified object, and the Notify action of the target object notifies the changes in the value of the target attribute as defined earlier in the UM3Base class to the UM3ActionBroker object. When the Notify action of the UM3ActionBroker object receives the notification of the changes of values in the target attribute, it checks the value of the corresponding attribute and tests the conditions. If the conditions are satisfied, then it is notified to the manager through the ConditionDetectedReport operation.

13.28. ChangeOfAttributeValueReport operation

ChangeOfAttributeValueReport 오퍼레이션은 RequestChangeOfAttributeValueReport 오퍼레이션에 대한 응답 오퍼레이션입니다. 즉, 특정 애트리뷰트의 값이 변화할 경우 해당 변화사실을 이벤트로 보고하기 위한 응답 오퍼레이션입니다.

The ChangeOfAttributeValueReport operation is a response operation for the RequestChangeOfAttributeValueReport operation. In other words, this response operation is used to report the changes in the values of a specific attribute as

an event.

13.28.1. Structure of Signature and Return Type

ChangeOfAttributeValueReport 오퍼레이션은 애트리뷰트 값의 변화를 매니저에게 보고하기 위한 오퍼레이션이므로, 이벤트가 발생한 애트리뷰트의 UM3 RDN, 변화한 애트리뷰트의 값, 변화한 일시, RequestChangeOfAttributeValueReport 오퍼레이션의 UM3 session identifier 등의 정보를 매니저로 리턴해주어야 합니다.

The ChangeOfAttributeValueReport operation is used to report the event of an attribute to the manager when there are changes in the values of attribute, and information such as UM3 RDN of the attribute that triggered the event, changed value of attribute, date and time of change, and UM3 session identifier of the RequestChangeOfAttributeValueReport operation should be returned to the manager.

표 66-ChangeOfAttributeValueReport 오퍼레이션의 signature 와 리턴 타입의 구조 [에서 보는 바와 같이 요청 signature 는 한 개의 파라메터 만이 주어지며, 본 권고안이 정의하는다른 오퍼레이션들과는 달리 실패 관련 signature 는 존재하지 않습니다. 즉, 이벤트가 감지되었을 때만 해당 오퍼레이션이 요청 signature를 통해 매니저로 송신되며, 매니저는 해당 오퍼레이션에 대한 UM3Ack 를 에이전트에 송신하여 이벤트 리포트를 접수하였음을 알리고 연결을 끊게 됩니다.

As shown in Table 65, the REQUEST signature has only one parameter, and the SUCCESS signature also has only one parameter. Unlike the other operations defined in this recommendation, there is no signature related to failure. In other words, the corresponding operation is sent to the manager through REQUEST signature only when an event is detected, and the manager sends UM3Ack for the corresponding operation to the agent to indicate the event report is received, and then it closes the connection.

표 66-ChangeOfAttributeValueReport 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of ChangeOfAttributeValueReport operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3EventReport	parUM3EventReport	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

This operation can be defined by using ASN.1 regular expressions as follows.

```
ChangeOfAttributeValueReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName DEFAULT received-session-identifier,
        &parUM3EventReport                UM3EventReport
}
```

13.28.2. REQUEST signature

요청 signature 는 UM3EventReport 클래스 타입의 parUM3EventReport 파라메터가 주어집니다. 단, UM3OperationResponse 오퍼레이션과 마찬가지로 ChangeOfAttributeValueReport 오퍼레이션의 um3OperationObjectName 애트리뷰트의 값은 해당 ChangeOfAttributeValueReport 오퍼레이션을 기동시킨 RequestChangeOfAttributeValueReport 오퍼레이션의 세션 아이디의 값을 갖고 있어야 합니다.

The REQUEST signature has a parUM3EventReport parameter of UM3EventReport class type. Like the UM3OperationResponse operation, the value of the um3OperationObjectName attribute of the ChangeOfAttributeValueReport operation should have the session ID of the RequestChangeOfAttributeValueReport operation that triggered the corresponding ChangeOfAttributeValueReport operation.

13.28.2.1. parUM3EventReport

UM3EventReport 클래스 타입의 오브젝트가 파라메터의 형태로 주어집니다. 해당 오브젝트는 모든 애트리뷰트이 값이 채워진 완전한 형태로 주어져야 합니다.

The UM3EventReport class type object is provided as a parameter. The corresponding object should be provided with all attributes completely filled up.

13.28.3. SUCCESS signature

13.28.3.1. parResult : TRUE

ChangeOfAttributeValueReport 오퍼레이션의 요청 즉, 이벤트 발생 사실과 이벤트의 내용에 대한 보고를 정상적으로 접수하였다는 매니저 측의 UM3Ack APDU 입니다

This is UM3Ack APDU from the manager that indicates that request of the ChangeOfAttributeValueReport operation i.e. report about the event status and that contents of the event are received normally.

13.28.4. FAIL signature OVLD 3

13.28.4.1. parResult : FALSE

ChangeOfAttributeValueReport 오퍼레이션의 요청 즉, 이벤트 발생 사실과 이벤트의 내용에 대한 보고를 정상적으로 접수하지 못하였다는 매니저 측의 실패 APDU 입니다. 실패하는 원인은 거의 대부분 해당 APDU 의 세션 아이디가 존재하지 않는 경우입니다.

This is FAIL APDU from the manager that indicates that the request of ChangeOfAttributeValueReport operation i.e. report about the event status and that contents of the event are not received normally. In most cases, the cause of failure is that the session ID of the corresponding APDU does not exist.

13.28.4.2. parUM3OperationErrorCode

ChangeOfAttributeValueReport 오퍼레이션의 요청 즉, 이벤트 발생 사실과 이벤트의 내용에 대한 보고를 접수하는 과정에서 에러가 발생하였을 경우 해당 에러에 대한 원인을 나타냅니다.

This contains the cause of error if an error occurs while receiving the request of the ChangeOfAttributeValueReport operation i.e. the report about event status and contents of event.

13.28.5. Handling Procedure at the Receiver

해당 오퍼레이션을 수신한 매니저는 ChangeOfAttributeValueReport 오퍼레이션의 APDU 에 포함되어 전송된 um3OperationObjectName 의 값 즉, N 필드의 값을 확인하여 어떤 이벤트 요청에 대한 이벤트 보고 인지를 확인해야 합니다. 이는 RequestChangeOfAttributeValueReport 오퍼레이션에 의한 이벤트 보고 요청시 해당 APDU 의 N 필드에 기록되어 있는 UM3 session identifier 를 이용하는 방법입니다. 앞서 정의한 바와 같이 본 권고안이 정의하는 UM3 프로토콜은 오퍼레이션의 인코딩 시 N 필드에 세션 아이디를 기록하고, 해당 오퍼레이션에 대한 응답시에 매니저로부터 수신한 세션 아이디를 그대로 복사하여 사용합니다. 따라서, 매니저는 자신이 에이전트로 발송하는 세션 아이디 값을 기억하고 있어야 하며 해당 세션아이디가 어떤 애트리뷰트와 어떤 조건으로 이벤트를 감지하도록 설정되어 있는지를 기억하고 관리해야 합니다.

When the manager receives the corresponding operation, it checks the value of um3OperationObjectName sent with APDU of the ChangeOfAttributeValueReport operation i.e. N field, and it should check what request the event report is for. The UM3 session identifier stored in N field of the corresponding APDU is used for the event report request by the RequestChangeOfAttributeValueReport operation in this method. As defined earlier, the session ID is stored in the N field while encoding the operation of the UM3 protocol defined in this recommendation. The session ID received from the manager as a response to the corresponding operation should be copied and used without modification. Therefore, the manager should remember the session ID sent to the agent, and it also should remember

and manage the settings of the event conditions including the attributes and conditions for each corresponding session ID.

이러한 event driven 방식은 결과적으로 매니저의 처리과정이 복잡해지는 단점이 있습니다. 이에 반해 polling 방식은 매니저가 설치된 서버측에 더 많은 부하가 걸리게되는 단점이 있으나 에이전트가 설치된 게이트웨이 혹은 컴퓨터에 부담을 적게주는 장점도 있습니다.

One disadvantage of this event driven method is the complexity of the process for the manager as a result. On the other hand, the polling method's disadvantage is higher server load where the manager is installed, but it has the advantage of lower loading for the computer or gateway where agent is installed.

에이전트가 검출한 이벤트의 통지 혹은 보고를 받은 매니저는 해당 오퍼레이션에 대해 UM3Ack APDU를 송신하고 그 연결을 끊는 과정을 모든 이벤트의 통지 혹은 보고에 대해 반복합니다.

When the manager receives the notification or report on the event detected by the agent, the manager should send UM3Ack to the corresponding operation and close the connection, and it should repeat this procedure for all event notifications or reports.

13.29. ConditionDetectedReport operation

ConditionDetectedReport 오퍼레이션은 RequestConditionDetectedReport 오퍼레이션에 대한 응답 오퍼레이션입니다. 즉, 특정 애트리뷰트의 값이 주어진 조건에 부합하는 값으로 변화할 경우 해당 변화사실을 이벤트로 보고하기 위한 응답 오퍼레이션입니다.

The ConditionDetectedReport operation is a response operation for the RequestConditionDetectedReport operation. In other words, when the charges of values of specific attributes meet the specified conditions, this response operation is used to report the corresponding changes as an event.

13.29.1. Structure of Signature and Return Type

ConditionDetectedReport 오퍼레이션은 애트리뷰트 값이 미리설정해둔 조건에 부합하는 형태로 변화한 이벤트를 매니저에게 보고하기 위한 오퍼레이션입니다. 따라서, 앞서 정의한 ChangeOfAttributeValueReport 오퍼레이션에서와 같이 이벤트가 발생한 애트리뷰트의 UM3 RDN, 변화한 애트리뷰트의 값, 변화한 일시, RequestConditionDetectedReport 오퍼레이션의 UM3 session identifier 등의 정보를 매니저로 리턴해주어야 합니다.

The ConditionDetectedReport operation is used to report the events that are triggered when the values of the attributes are changed to the form that meets the predetermined conditions. Therefore, information like UM3 RND of the attribute that is associated with the event, changed value of attribute, date of change, and UM3 session identifier should be returned to the manager.

표 67-ConditionDetectedReport 오퍼레이션의 signature 와 리턴 타입의 구조 에서 보는 바와 같이 요청 signature 및 그에 대한 UM3Ack APDU 의 송신은 앞서 정의한 ChangeOfAttributeValueReport 오퍼레이션과 동일합니다.

As shown in Table 66, the REQUEST signature and transmission of the UM3Ack APDU are the same as the previously defined ChangeOfAttributeValueReport operation.

표 67-ConditionDetectedReport 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of ConditionDetectedReport operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3EventReport	parUM3EventReport	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

This operation can be defined by using ASN.1 regular expressions as follows.

```
ConditionDetectedReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parUM3EventReport              UM3EventReport
}
```

13.29.2. REQUEST signature

요청 signature 는 UM3EventReport 클래스 타입의 parUM3EventReport 파라메터가 주어집니다.

The REQUEST signature has the parUM3EventReport parameter of UM3EventReport class type.

13.29.2.1. parUM3EventReport

UM3EventReport 클래스 타입의 오브젝트가 파라메터의 형태로 주어집니다. 해당 오브젝트는 모든 애트리뷰트이 값이 채워진 완전한 형태로 주어져야 합니다.

The object of UM3EventReport class type is provided as a parameter. The corresponding object should be provided

with all attributes completely filled up.

13.29.3. SUCCESS signature

13.29.3.1. parResult : TRUE

ConditionDetectedReport 오퍼레이션의 요청 즉, 이벤트 발생 사실과 이벤트의 내용에 대한 보고를 정상적으로 접수하였다는 매니저 측의 UM3Ack APDU 입니다

It is UM3Ack APDU from the manager that indicates that request of the ConditionDetectedReport operation i.e. report about the event status and that contents of the event are received normally.

13.29.4. FAIL signature OVLD 3

13.29.4.1. parResult : FALSE

ChangeOfAttributeValueReport 오퍼레이션의 요청 즉, 이벤트 발생 사실과 이벤트의 내용에 대한 보고를 정상적으로 접수하지 못했다는 매니저 측의 실패 APDU 입니다. 실패하는 원인은 거의 대부분 해당 APDU의 세션 아이디가 존재하지 않는 경우입니다.

This is the FAIL APDU from the manager that indicates that the request of ChangeOfAttributeValueReport operation i.e. report about the event status and that contents of the event are not received normally. In most cases, the cause of failure is that the session ID of the corresponding APDU does not exist.

13.29.4.2. parUM3OperationErrorCode

ChangeOfAttributeValueReport 오퍼레이션의 요청 즉, 이벤트 발생 사실과 이벤트의 내용에 대한 보고를 접수하는 과정에서 에러가 발생하였을 경우 해당 에러에 대한 원인을 나타냅니다.

This contains the cause of error if an error occurs while receiving the request of the ChangeOfAttributeValueReport operation i.e. the report about event status and contents of event.

13.29.5. Handling Procedure at the Receiver

해당 오퍼레이션을 수신한 매니저는 ConditionDetectedReport 오퍼레이션의 APDU 에 포함되어 전송된 um3OperationObjectName 의 값 즉, N 필드의 값을 확인하여 어떤 이벤트 요청에 대한 이벤트 보고 인지를 확인해야 합니다. 이는 RequestConditionDetectedReport 오퍼레이션에 의한 이벤트 보고 요청시 해당 APDU 의 N 필드에 기록되어 있는 UM3 session identifier 를 이용하는 방법으로서 앞서 정의한 ChangeOfAttributeValueReport 오퍼레이션과 동일한 절차와 방식으로 해당 오퍼레이션요청을 처리합니

다.

When the manager receives the corresponding operation, it checks the value of um3OperationObjectName sent with APDU of the ConditionDetectedReport operation i.e. N field, and it should check what request the event report is for. UM3 session identifier stored in N field of the corresponding APDU is used for the event report request by the RequestConditionDetectedReport operation in this method. The request of the corresponding operation is processed in the same way as the procedure for the ChangeOfAttributeValueReport that was defined earlier.

13.30. CancelEventReport operation

CancelEventReport 오퍼레이션은 RequestChangeOfAttributeValueReport 오퍼레이션과 RequestConditionDetectedReport 오퍼레이션에 대한 해제 혹은 취소를 명령하는 오퍼레이션입니다.

The CancelEventReport operation is the command to cancel the RequestChangeOfAttributeValueReport operation and RequestConditionDetectedReport operation.

13.30.1. Structure of Signature and Return Type

에이전트는 RequestChangeOfAttributeValueReport 오퍼레이션과 RequestConditionDetectedReport 오퍼레이션 APDU 의 N 필드에 기록되어 전달된 UM3 session identifier 를 기억하고 있어야 합니다. 그리고 해당 UM3 session identifier 를 파라메터 값으로 하는 CancelEventReport 오퍼레이션이 전달될 경우 해당 UM3 session identifier 로 구분되는 이벤트 검출 과정을 종료할 수 있어야 합니다.

The agent should remember the UM3 session identifier stored in the N field of the RequestChangeOfAttributeValueReport operation and RequestConditionDetectedReport operation APDU. It should terminate the event detection process that is identified by the UM3 session identifier stored in the parameter of the CancelEventReport operation when this operation is requested.

표 68-CancelEventReport 오퍼레이션의 signature 와 리턴 타입의 구조
[Structure of Signature and Return Type of CancelEventReport operation]

Classification	Parameter Type	Name and Value of Parameter	Characteristics
REQUEST	UM3ObjectName	parUM3SessionIdentifier	
SUCCESS	UM3Boolean	parResult: TRUE	
FAIL	UM3Boolean UM3OperationErrorCode	parResult: FALSE parUM3OperationErrorCode	OVLD 3

다음은 ASN.1 정규표현식에 의한 오퍼레이션의 표현입니다.

This operation can be defined by using ASN.1 regular expressions as follows.

```
CancelEventReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier      UM3ClassIdentifier,
        &um3OperationObjectName           UM3ObjectName UNIQUE,
        &parUM3SessionIdentifier          UM3ObjectName
}
```

13.30.2. REQUEST signature

요청 signature 는 UM3CharacterString 클래스 타입의 parUM3SessionIdentifier 파라메터가 주어집니다. 해당 parUM3SessionIdentifier 파라메터의 값은 RequestChangeOfAttributeValueReport 오퍼레이션과 RequestConditionDetectedReport 오퍼레이션이 요청될 때 주어진 N 필드의 UM3 session identifier 값들 중 하나와 동일해야 합니다.

The REQUEST signature has the parUM3SessionIdentifier parameter of UM3CharacterString class type. The value of the corresponding parUM3SessionIdentifier parameter should match one of the UM3 identifier values of the N field provided along with the request of the RequestChangeOfAttributeValueReport operation and RequestConditionDetectedReport operation.

13.30.2.1. parUM3SessionIdentifier

RequestChangeOfAttributeValueReport 오퍼레이션과 RequestConditionDetectedReport 오퍼레이션에 의해 동작하고 있는 이벤트 검출 과정에 대한 일련의 과정에 부여된 고유 아이디 입니다. 해당 고유 아이디는 RequestChangeOfAttributeValueReport 오퍼레이션과 RequestConditionDetectedReport 오퍼레이션이 요청될 때 해당 오퍼레이션 APDU 의 N 필드의 값으로 주어진 UM3 session identifier 값과 동일한 값이어야 합니다.

This contains the unique ID that is assigned to the series of tasks related to the event detection process that is controlled by the RequestChangeOfAttributeValueReport operation and RequestConditionDetectedReport operation. These unique IDs should be equal to the value of the UM3 session identifier provided as the value of the N field of the corresponding operation APDU when the RequestChangeOfAttributeValueReport operation and RequestConditionDetectedReport operation are requested.

13.30.3. SUCCESS signature

13.30.3.1. parResult : TRUE

CancelEventReport 오퍼레이션의 요청 즉, 이벤트 검출을 위한 일련의 과정에 대한 동작을 중단하였다는, 에이전트가 보내는 일종의 UM3Ack APDU 입니다

This is a kind of UM3Ack APDU that is sent by the agent to the request of the CancelEventReport operation, indicating that the series of activities to detect the event are cancelled.

13.30.4. FAIL signature OVLD 3

CancelEventReport 오퍼레이션의 요청 즉, 이벤트 검출을 위한 일련의 과정에 대한 동작을 중단 할 수 없는 오류상황임을 나타냅니다.

This indicates an error that the request of CancelEventReport operation, i.e. to cancel the series of activates for event detection, cannot be cancelled.

13.30.4.1. parResult : FALSE

오퍼레이션의 요청에 따른 수행 결과가 실패임을 나타냅니다.

This indicates that the result of the execution according to the request of operation has failed.

13.30.4.2. parUM3OperationErrorCode

오퍼레이션이 실패한 이유를 나타내는 파라메터 입니다. 주로 잘못된 UM3 session identifier 가 주어졌음을 나타내는 경우가 많으며 이럴 경우 parUM3OperationErrorCode 의 값은 noSuchSession 으로 지정되어야 합니다.

This parameter contains the cause of failure of operation. In most cases, it is caused when the provided UM3 session identifier is incorrect. In this case, the value of parUM3OperationErrorCode should be set to noSuchSession.

13.30.5. Handling Procedure at the Receiver

매너저로부터 해당 오퍼레이션을 수신한 에이전트는 오퍼레이션 APDU 의 파라메터에 기록되어 전달된 UM3 session identifier 를 확인하고 이와 동일한 UM3 session identifier 로 명명된 이벤트 검출 프로세스를 검색합니다. 해당 프로세스가 검색되었을 경우 해당 프로세스의 동작을 해제하고 그 결과를 parResult 파라메터에 TRUE 값을 기록하여 매니저로 전송하고 세션을 종료합니다. 만약 해당 이벤트 검출 과정을 찾을 수 없을 경우 혹은 다른 형태의 에러가 발생할 경우에는 해당 오류의 종류를

UM3OperationErrorCode 타입의 파라메터에 기록하여 매니저로 전송합니다.

When the agent receives the corresponding operation from the manager, it checks the UM3 session identifier in the parameter of the operation APDU and searches for the event detection process that is associated with the corresponding UM3 session identifier. If it is found, then it cancels the operation of the corresponding process and sets the value of parResult parameter to TRUE. It then sends the result to the manager and terminates the session. If the corresponding event detection process cannot be found or an error is caused by other reasons, then the type of error is stored in the UM3OperationErrorCode type parameter and it is returned to the manager.

14. UM3 ASN.1 module 의 정의

이 장에서는 본 권고안이 정의하는 모든 타입 혹은 클래스들을 ASN.1 정규표현식으로 기술합니다. 순서는 영문 알파벳 순서로 나열됩니다. 또한 UM3 오퍼레이션 클래스들에 대한 ASN.1 정규표현식도 함께 기술합니다.

14.1. UM3 서비스관리 정보 모델에 대한 ASN.1 module 의 정의

```
UM3 Service Management Information Module DEFINITIONS ::=
BEGIN


AccumulatorSensor ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentSensorStatus            DeviceOperationStatus,
        &serialNumber                   UM3CharacterString,
        &presentValue                   PresentAccumulatorSensorValue
}


AnalogControl ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
```

```
        &manufacturerInfo                   CompanyInfo OPTIONAL,
        &vendorInfo                         CompanyInfo OPTIONAL,
        &maintenancePersonel                PersonInfo OPTIONAL,
        &operationalCondition               DeviceOperationalCondition OPTIONAL,
        &hasBattery                         UM3Boolean,
        &batteryInfo                        InstalledBattery OPTIONAL,
        &hasBackupBattery                   UM3Boolean,
        &backupBatteryInfo                  InstalledBattery OPTIONAL,
        &presentBatteryValueInfo            PresentBatteryValue,
        &presentBackupBatteryValueInfo      PresentBatteryValue OPTIONAL,
        &installedEnvironment               InstalledEnvironment OPTIONAL,
        &powerConsumption                   UM3Real,
        &levelInAConfigurationTree          UM3UnsignedInteger16,
        &upperGatewayAddress                UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo                DeviceMaintenanceScheduleInfo,
        &presentControlStatus               DeviceOperationStatus,
        &serialNumber                       UM3CharacterString,
        &presentValue                       PresentAnalogControlValue
}

AnalogSensor ::= UM3 CLASS {
        &um3ClassIdentifier                 UM3ClassIdentifier,
        &um3ObjectName                      UM3ObjectName,
        &um3ObjectNameAlias                 UM3CharacterString OPTIONAL,
        &um3ObjectDescription               UM3CharacterString OPTIONAL,
        &um3AttributeList                   UM3ObjectList,
        &manufacturerInfo                   CompanyInfo OPTIONAL,
        &vendorInfo                         CompanyInfo OPTIONAL,
        &maintenancePersonel                PersonInfo OPTIONAL,
        &operationalCondition               DeviceOperationalCondition OPTIONAL,
        &hasBattery                         UM3Boolean,
        &batteryInfo                        InstalledBattery OPTIONAL,
        &hasBackupBattery                   UM3Boolean,
        &backupBatteryInfo                  InstalledBattery OPTIONAL,
        &presentBatteryValueInfo            PresentBatteryValue,
        &presentBackupBatteryValueInfo      PresentBatteryValue OPTIONAL,
        &installedEnvironment               InstalledEnvironment OPTIONAL,
        &powerConsumption                   UM3Real,
        &levelInAConfigurationTree          UM3UnsignedInteger16,
        &upperGatewayAddress                UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo                DeviceMaintenanceScheduleInfo,
        &presentSensorStatus                DeviceOperationStatus,
        &serialNumber                       UM3CharacterString,
        &location                           UM3CharacterString,
        &presentValue                       PresentAnalogSensorValue
}
```

```
BinaryControl ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentControlStatus           DeviceOperationStatus,
        &serialNumber                   UM3CharacterString,
        &presentValue                   PresentBinaryControlValue
}

BinarySensor ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentSensorStatus            DeviceOperationStatus,
        &serialNumber                   UM3CharacterString,
```

```
        &presentValue                   PresentBinarySensorValue
}

CompanyInfo ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &name                           UM3CharacterString,
        &phoneNumber                    UM3CharacterString,
        &faxNumber                      UM3CharacterString OPTIONAL,
        &email                          UM3CharacterString,
        &address                        UM3CharacterString OPTIONAL
}

ControlBase ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentControlStatus           DeviceOperationStatus,
        &serialNumber                   UM3CharacterString
}

DeviceMaintenanceScheduleInfo ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
```

```
        &um3AttributeList           UM3ObjectList,
        &inServiceDateTime          UM3DateTime,
        &latestMaintenanceDateTime  UM3DateTime OPTIONAL,
        &nextMaintenanceDateTime    UM3DateTime OPTIONAL,
        &durablePeriod              UM3DateTime,
        &maintenancePeriod          UM3DateTime OPTIONAL,
        &numberOfRebooting          UM3UnsignedInteger16 OPTIONAL,
        &shutdownScheduleInfo       UM3DateTime OPTIONAL
}

DeviceOperationalCondition ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &temperatureMax             UM3Real,
        &temperatureMin             UM3Real,
        &ratedVoltage               UM3UnsignedInteger32,
        &powerConsumption           UM3UnsignedInteger32,
        &humidityMax                UM3UnsignedInteger16
}

DeviceOperationStatus ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &presentStatus              UM3Enumerated {
                                        outOfService        (0),
                                        inService           (1),
                                        inMaintenance       (2),
                                        inFault             (3),
                                        communicationFailure (4),
                                        unknownCause        (5),
                                        detached            (6)
                                    },
        &statusTimestamp            UM3DateTime,
        &resumeDateTime             UM3DateTime
}

Gateway ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
```

&um3ObjectDescription	**UM3CharacterString OPTIONAL,**
&um3AttributeList	**UM3ObjectList,**
&manufacturerInfo	**CompanyInfo OPTIONAL,**
&vendorInfo	**CompanyInfo OPTIONAL,**
&maintenancePersonel	**PersonInfo OPTIONAL,**
&operationalCondition	**DeviceOperationalCondition OPTIONAL,**
&hasBattery	**UM3Boolean,**
&batteryInfo	**InstalledBattery OPTIONAL,**
&hasBackupBattery	**UM3Boolean,**
&backupBatteryInfo	**InstalledBattery OPTIONAL,**
&installedEnvironment	**InstalledEnvironment OPTIONAL,**
&hasCoolingFan	**UM3Boolean OPTIONAL,**
&cpuInfo	**InstalledCPU,**
&memoryInfo	**InstalledMemory,**
&um3ProtocolVersion	**UM3CharacterString,**
&powerConsumption	**UM3Real,**
&presentBatteryValueInfo	**PresentBatteryValue,**
&presentBackupBatteryValueInfo	**PresentBatteryValue OPTIONAL,**
&numberOfAnalogInputPort	**UM3UnsignedInteger16,**
&numberOfAnalogOutputPort	**UM3UnsignedInteger16,**
&numberOfBinaryInputPort	**UM3UnsignedInteger16,**
&numberOfBinaryOutputPort	**UM3UnsignedInteger16,**
&numberOfRS232Port	**UM3UnsignedInteger16,**
&numberOfRS485Port	**UM3UnsignedInteger16,**
&numberOfWirelessPort	**UM3UnsignedInteger16,**
&numberOfEthernetPort	**UM3UnsignedInteger16,**
&numberOfIRPort	**UM3UnsignedInteger16,**
&numberOfOpticalPort	**UM3UnsignedInteger16,**
&doesSupportExternalMemoryCard	**UM3Boolean,**
&externalMemoryCardType	**UM3CharacterString OPTIONAL,**
&analogInputPortDescription	**UM3CharacterString OPTIONAL,**
&analogOutputPortDescription	**UM3CharacterString OPTIONAL,**
&binaryInputPortDescription	**UM3CharacterString OPTIONAL,**
&binaryOutputPortDescription	**UM3CharacterString OPTIONAL,**
&rs232PortDescription	**UM3CharacterString OPTIONAL,**
&rs485PortDescription	**UM3CharacterString OPTIONAL,**
&wirelessPortDescription	**UM3CharacterString OPTIONAL,**
&ethernetPortDescription	**UM3CharacterString OPTIONAL,**
&irPortDescription	**UM3CharacterString OPTIONAL,**
&opticalPortDescription	**UM3CharacterString OPTIONAL,**
&softwareInfo	**InstalledSoftware,**
&hasHardDrive	**UM3Boolean,**
&hardDriveInfo	**InstalledHardDrive OPTIONAL,**
&isDedicatedForOneNode	**UM3Boolean OPTIONAL,**
&levelInAConfigurationTree	**UM3UnsignedInteger16,**
&numberOfLowerGateway	**UM3UngisnedInteger32,**
&numberOfLowerNode	**UM3UnsignedInteger32,**
&upperNodeAddress	**UM3TcpIpAddress,**
&address	**UM3TcpIpAddress,**
&doesSupportTelnet	**UM3Boolean,**
&doesSupportFtp	**UM3Boolean,**

```
        &analogSensorList               UM3 SEQUENCE OF AnalogSensor,
        &binarySensorList               UM3 SEQUENCE OF BinarySensor,
        &accumulatedAnalogSensorList    UM3 SEQUENCE OF AccumulatedAnalogSensorList,
        &analogControlList              UM3 SEQUENCE OF AnalogControl,
        &binaryControlList              UM3 SEQUENCE OF BinaryControl,
        &multiStateSensorList           UM3 SEQUENCE OF MultiStateSensor,
        &multiStateControlList          UM3 SEQUENCE OF MultiStateControl,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentValue                   PresentGatewayValue,
        &presentGatewayStatus           DeviceOperationStatus OPTIONAL,
        &maxAPDU                        UM3UnsignedInteger32,
        &sereialNumber                  UM3CharacterString
}


InstalledBattery ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &temperatureMax                 UM3Real OPTIONAL,
        &temperatureMin                 UM3Real OPTIONAL,
        &isBackupBattery                UM3Boolean OPTIONAL,
        &isChargable                    UM3Boolean OPTIONAL,
        &ratedVoltage                   UM3UnsignedInteger32,
        &ratedCurrent                   UM3Real,
        &type                           UM3CharacterString OPTIONAL,
        &size                           UM3CharacterString,
        &weight                         UM3Real OPTIONAL,
        &presentVoltage                 UM3Real,
        &remainingBatteryTime           UM3DateTime OPTIONAL
}


InstalledCPU ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &numberOfCPU                    UM3UnsignedInteger16,
        &manufacturer                   UM3CharacterString,
        &model                          UM3CharacterString,
        &speed                          UM3CharacterString,
        &busWidth                       UM3UnsignedInteger16
}
```

```
InstalledEnvironment ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &isIndoor                   UM3Boolean,
        &hasEnclosure               UM3Boolean,
        &isWaterproof               UM3Boolean,
        &altitude                   UM3Integer16 OPTIONAL,
        &isRackMounted              UM3Boolean OPTIONAL,
        &size                       UM3CharacterString OPTIONAL,
        &pointDescription           UM3CharacterString OPTIONAL,
        &installedDate              UM3DateTime
}

InstalledHardDrive ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &diskSize                   UM3UnsignedInteger32,
        &speed                      UM3CharacterString OPTIONAL,
        &manufacturer               UM3CharacterString OPTIONAL,
        &diskDiameter               UM3Real,
        &hasReplaced                UM3Boolean,
        &replacedDateTime           UM3DateTime,
        &serialNumber               UM3CharacterString
}

InstalledMemory ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &installedMemorySize        UM3UnsignedInteger32,
        &maximumExpandableSize      UM3UnsignedInteger32,
        &accessTime                 UM3Real OPTIONAL,
        &manufacturer               UM3CharacterString OPTIONAL,
        &numberOfMemorySlot         UM3UnsignedInteger16 OPTIONAL,
        &currentMemorySizePerSlot   UM3UnsignedInteger32 OPTIONAL
}
```

```
InstalledSoftware ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &isOS                           UM3Boolean,
        &nameOfSoftware                 UM3CharacterString,
        &version                        UM3CharacterString,
        &updateDateTime                 UM3DateTime,
        &numberOfUpdateUpToNow          UM3UnsignedInteger16 OPTIONAL,
        &manufacturer                   UM3CharactgerString OPTIONAL
}

MultiStateControl ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &manufacturerInfo               CompanyInfo OPTIONAL,
        &vendorInfo                     CompanyInfo OPTIONAL,
        &maintenancePersonel            PersonInfo OPTIONAL,
        &operationalCondition           DeviceOperationalCondition OPTIONAL,
        &hasBattery                     UM3Boolean,
        &batteryInfo                    InstalledBattery OPTIONAL,
        &hasBackupBattery               UM3Boolean,
        &backupBatteryInfo              InstalledBattery OPTIONAL,
        &presentBatteryValueInfo        PresentBatteryValue,
        &presentBackupBatteryValueInfo  PresentBatteryValue OPTIONAL,
        &installedEnvironment           InstalledEnvironment OPTIONAL,
        &powerConsumption               UM3Real,
        &levelInAConfigurationTree      UM3UnsignedInteger16,
        &upperGatewayAddress            UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo            DeviceMaintenanceScheduleInfo,
        &presentSensorStatus            DeviceOperationStatus,
        &serialNumber                   UM3CharacterString,
        &presentValue                   PresentMultiStateControlValue
}

MultiStateSensor ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
```

```
        &manufacturerInfo                CompanyInfo OPTIONAL,
        &vendorInfo                      CompanyInfo OPTIONAL,
        &maintenancePersonel             PersonInfo OPTIONAL,
        &operationalCondition            DeviceOperationalCondition OPTIONAL,
        &hasBattery                      UM3Boolean,
        &batteryInfo                     InstalledBattery OPTIONAL,
        &hasBackupBattery                UM3Boolean,
        &backupBatteryInfo               InstalledBattery OPTIONAL,
        &presentBatteryValueInfo         PresentBatteryValue,
        &presentBackupBatteryValueInfo   PresentBatteryValue OPTIONAL,
        &installedEnvironment            InstalledEnvironment OPTIONAL,
        &powerConsumption                UM3Real,
        &levelInAConfigurationTree       UM3UnsignedInteger16,
        &upperGatewayAddress             UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo             DeviceMaintenanceScheduleInfo,
        &presentSensorStatus             DeviceOperationStatus,
        &serialNumber                    UM3CharacterString,
        &presentValue                    PresentMultiStateSensorValue
}

PersonInfo ::= UM3 CLASS {
        &um3ClassIdentifier              UM3ClassIdentifier,
        &um3ObjectName                   UM3ObjectName,
        &um3ObjectNameAlias              UM3CharacterString OPTIONAL,
        &um3ObjectDescription            UM3CharacterString OPTIONAL,
        &um3AttributeList                UM3ObjectList,
        &name                            UM3CharacterString,
        &phoneNumber                     UM3CharacterString OPTIONAL,
        &cellPhoneNumber                 UM3CharacterString,
        &email                           UM3CharacterString,
        &address                         UM3CharacterString OPTIONAL,
        &companyName                     UM3CharacterString OPTIONAL
}

PresentAccumulatorSensorValue ::= UM3 CLASS {
        &um3ClassIdentifier              UM3ClassIdentifier,
        &um3ObjectName                   UM3ObjectName,
        &um3ObjectNameAlias              UM3CharacterString OPTIONAL,
        &um3ObjectDescription            UM3CharacterString OPTIONAL,
        &um3AttributeList                UM3ObjectList,
        &presentValue                    UM3Real,
        &presentValueTimestamp           UM3DateTime,
        &previousValue                   UM3Real,
        &previousValueTimestamp          UM3DateTime,
        &minPresentValue                 UM3Real,
        &minPresentValueTimestamp        UM3DateTime,
        &units                           UM3CharacterString,
```

```
        &updatePeriod                   UM3DateTime,
        &acceptableMaxPresentValue      UM3Real
}


PresentAnalogControlValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3Real,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3Real,
        &previousValueTimestamp         UM3DateTime,
        &maxPresentValue                UM3Real,
        &minPresentValue                UM3Real,
        &resolution                     UM3Real,
        &units                          UM3CharacterString,
        &acceptableMaxPresentValue      UM3Real,
        &acceptableMinPresentValue      UM3Real
}


PresentAnalogSensorValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3Real,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3Real,
        &previousValueTimestamp         UM3DateTime,
        &maxPresentValue                UM3Real,
        &minPresentValue                UM3Real,
        &resolution                     UM3Real,
        &units                          UM3CharacterString,
        &updatePeriod                   UM3DateTime,
        &acceptableMaxPresentValue      UM3Real,
        &acceptableMinPresentValue      UM3Real
}


PresentBatteryValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
```

```
        &um3AttributeList               UM3ObjectList,
        &presentVoltage                 UM3Real,
        &previousVoltage                UM3Real,
        &remainingBatteryTime           UM3DateTime
}

PresentBinaryControlValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3Boolean,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3Boolean,
        &previousValueTimestamp         UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}

PresentBinarySensorValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3Boolean,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3Boolean,
        &previousValueTimestamp         UM3DateTime,
        &updatePeriod                   UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}

PresentGatewayValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &kbytesMemoryInUse              UM3Real,
        &kbytesMemoryInUseTimestamp     UM3DateTime,
        &mbytesHarddiskInUse            UM3Real OPTIONAL,
        &mbytesHarddiskInUseTimestamp   UM3DateTime OPTIONAL,
        &percentCpuTime                 UM3Real,
```

```
        &percentCpuTimeTimestamp        UM3DateTime,
        &bytesSentPerSecond             UM3UnsignedInteger32,
        &bytesReceivedPerSecond         UM3UnsignedInteger32,
        &bytesPerSecondTimestamp        UM3DateTime
}

PresentMultiStateControlValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3UnsignedInteger16,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3UnsignedInteger16,
        &previousValueTimestamp         UM3DateTime,
        &numberOfStates                 UM3UnsignedInteger16,
        &stateNames                     UM3 SEQUENCE OF UM3CharacterString,
        &updatePeriod                   UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}

PresentMultiStateSensorValue ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &presentValue                   UM3UnsignedInteger16,
        &presentValueTimestamp          UM3DateTime,
        &previousValue                  UM3UnsignedInteger16,
        &previousValueTimestamp         UM3DateTime,
        &numberOfStates                 UM3UnsignedInteger16,
        &stateNames                     UM3 SEQUENCE OF UM3CharacterString,
        &updatePeriod                   UM3DateTime,
        &stateChangedCount              UM3UnsignedInteger16,
        &stateCountStartedDateTime      UM3DateTime
}

SensorBase ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
```

```
        &manufacturerInfo                CompanyInfo OPTIONAL,
        &vendorInfo                      CompanyInfo OPTIONAL,
        &maintenancePersonel             PersonInfo OPTIONAL,
        &operationalCondition            DeviceOperationalCondition OPTIONAL,
        &hasBattery                      UM3Boolean,
        &batteryInfo                     InstalledBattery OPTIONAL,
        &hasBackupBattery                UM3Boolean,
        &backupBatteryInfo               InstalledBattery OPTIONAL,
        &presentBatteryValueInfo         PresentBatteryValue,
        &presentBackupBatteryValueInfo   PresentBatteryValue OPTIONAL,
        &installedEnvironment            InstalledEnvironment OPTIONAL,
        &powerConsumption                UM3Real,
        &levelInAConfigurationTree       UM3UnsignedInteger16,
        &upperGatewayAddress             UM3TcpIpAddress OPTIONAL,
        &operationalTimeInfo             DeviceMaintenanceScheduleInfo,
        &presentSensorStatus             DeviceOperationStatus,
        &serialNumber                    UM3CharacterString
}


SensorMaintenanceScheduleInfo ::= UM3 CLASS {
        &um3ClassIdentifier              UM3ClassIdentifier,
        &um3ObjectName                   UM3ObjectName,
        &um3ObjectNameAlias              UM3CharacterString OPTIONAL,
        &um3ObjectDescription            UM3CharacterString OPTIONAL,
        &um3AttributeList                UM3ObjectList,
        &inServiceDateTime               UM3DateTime,
        &latestMaintenanceDateTime       UM3DateTime OPTIONAL,
        &nextMaintenanceDateTime         UM3DateTime OPTIONAL,
        &durablePeriod                   UM3DateTime,
        &maintenancePeriod               UM3DateTime OPTIONAL,
        &numberOfRewired                 UM3UnsignedInteger16 OPTIONAL,
        &detachScheduleInfo              UM3DateTime OPTIONAL
}


UM3 SEQUENCE ::= UM3 SEQUENCE “{“ “}” |
        UM3 SEQUENCE “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::= “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::=
        UM3ComplexType |
        UM3ComplexTypeList “,” UM3ComplexType

UM3ComplexType ::=
        UM3NamedType |
        UM3NamedType OPTIONAL |
        UM3NamedType DEFAULT VALUE |
        COMPONENT OF UM3Type
```

```
UM3NamedType ::=
        identifier UM3Type |
        UM3Type

UM3Type ::=
        UM3 Primitive type |
        UM3 Complex type |
        UM3 InformationModel Class type


UM3 SEQUENCE OF ::= UM3 SEQUENCE OF UM3Type


UM3 SET ::= UM3 SET “{“ “}” |
        UM3 SET “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::= “{“ UM3ComplexTypeList “}”

UM3ComplexTypeList ::=
        UM3ComplexType |
        UM3ComplexTypeList “,” UM3ComplexType

UM3ComplexType ::=
        UM3NamedType |
        UM3NamedType OPTIONAL |
        UM3NamedType DEFAULT VALUE |
        COMPONENT OF UM3Type

UM3NamedType ::=
        identifier UM3Type |
        UM3Type

UM3Type ::=
        UM3 Primitive type |
        UM3 Complex type |
        UM3 InformationModel Class type


UM3 SET OF ::= UM3 SET OF UM3Type


UM3ActionBroker ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &targetObjectClassIdentifier    UM3ClassIdentifier,
```

```
        &targetObjectObjectName         UM3ObjectName,
        &targetAttributeClassIdentifier UM3ClassIdentifier,
        &targetAttributeObjectName      UM3ObjectName,
        &targetPeriod                   UM3DateTime,
        &isOnlyOnceAction               UM3Boolean,
        &reservedActionTimestamp        UM3DateTime,
        &receipientAddress              UM3 SEQUENCE OF UM3CharacterString,
        &targetCondition                UM3ObjectValueCondition
}

UM3ActionInvoker ::= UM3UnsignedInteger16

UM3Base ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList
}

UM3BitString ::= BIT STRING

UM3Boolean ::= BOOLEAN

um3BooleanValue UM3Boolean ::= True | False

UM3CharacterString ::= CHARACTER STRING

UM3ClassIdentifier ::= UNSIGNED INTEGER

UM3DateTime ::= INTEGER

UM3Enumerated ::= ENUMERATED "{"Enumerations"}"

Enumerations ::=
        RootEnumeration |
        RootEnumeration "," "…" |
        RootEnumeration "," "…" "," AdditionalEnumeration

RootEnumeration ::= Enumeration
```

```
AdditionalEnumeration ::= Enumeration

Enumeration ::=
        EnumerationItem |
        EnumerationItem “,” Enumeration

EnumerationItem ::=
        identifier |
        NamedNumber

NamedNumber ::=
        identifier “(“ SignedNumber”)”

SignedNumber ::=
        number | “-“ number


UM3EventReport ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &targetObjectClassIdentifier    UM3ClassIdentifier,
        &targetObjectObjectName         UM3ObjectName,
        &targetAttributeClassIdentifier UM3ClassIdentifier,
        &targetAttributeObjectName      UM3ObjectName,
        &targetAttributeValue           ANY DEFINED BY UM3 ASN.1 module,
        &eventTimestamp                 UM3DateTime
}


UM3Integer16 ::= SHORT INTEGER


UM3Integer32 ::= INTEGER


UM3Integer64 ::= LONG INTEGER


UM3Null ::= NULL


UM3ObjectIndicator ::= UM3 SEQUENCE {
        um3ClassIdentifierIndicator     UM3ClassIdentifier,
        um3ObjectNameIndicator          UM3ObjectName
}
```

```
UM3ObjectList ::= UM3 SEQUENCE OF ANY DEFINED BY UM3 ASN.1 module


UM3ObjectListToBeCreated ::= UM3 SEQUENCE OF UM3 SEQUENCE {
        dn                              UM3ObjectName,
        anyTypeValue                    ANY DEFINED BY UM3 ASN.1 module
}


UM3ObjectName ::= CHARACTER STRING


UM3ObjectValueCondition ::= UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &upperEnd                       &EndLimitType,
        &lowerEnd                       &EndLimitType,
        &condition                      UM3Enumerated {
                                                LL              (0),
                                                GLLU            (1),
                                                GU              (2),
                                                LEL             (3),
                                                GELaLEU         (4),
                                                GEU             (5),
                                                GELaLU          (6),
                                                GLaLEU          (7),
                                                EU              (8),
                                                EL              (9),
                                                LLoGU           (10),
                                                LELoGU          (11),
                                                LELoGEU         (12),
                                                LLoGEU          (13),
                                                COV             (14)
                                        },
        &targetAttributeClassIdentifier UM3ClassIdentifier,
        &targetAttributeObjectName      UM3ObjectName
}


EndLimitType ::=
        UM3Integer16 |
        UM3Integer32 |
        UM3Integer64 |
        UM3UnsignedInteger16 |
```

```
        UM3UnsignedInteger32 |
        UM3UnsignedInteger64 |
        UM3Real |
        UM3CharacterString |
        UM3Boolean |
        UM3DateTime


UM3OctetString ::= OCTET STRING


UM3OperationErrorCode ::= UM3Enumerated {
        notFound                    (0),
        accessDenied                (1),
        noSuchClass                 (2),
        noSuchObject                (3),
        noSuchAttribute             (4),
        invalidAttributeClassType   (5),
        processingFailure           (6),
        unrecognizedOperation       (7),
        unknownSignature            (8),
        eventDrivenNotSupported     (9),
        noSuchSession               (10)
}


UM3ProtocolSupportDescription ::=UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &version                    UM3CharacterString,
        &level                      UM3CharacterString
}


UM3Real ::= REAL


UM3TcpIpAddress ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &um3ObjectDescription       UM3CharacterString OPTIONAL,
        &um3AttributeList           UM3ObjectList,
        &ipAddressV4                UM3CharacterString,
        &ipAddressV6                UM3CharacterString OPTIONAL,
        &portNumber                 UM3UnsignedInteger16,
```

```
        &tcpOrUdp                       UM3Boolean
}

UM3UnsignedInteger16 ::= UNSIGNED SHORT INTEGER

UM3UnsignedInteger32 ::= UNSIGNED INTEGER

UM3UnsignedInteger64 ::= UNSIGNED LONG INTEGER

UM3UnstructuredObject ::=UM3 CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &um3ObjectDescription           UM3CharacterString OPTIONAL,
        &um3AttributeList               UM3ObjectList,
        &UM3OpenType
}

END
```

14.2. UM3 통신서비스 모델에 대한 ASN.1 module 의 정의

```
UM3 Operation Service Module DEFINITIONS ::=
BEGIN

CancelGetObjectAttributeGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled   UM3ObjectName
}

CancelGetObjectAttributeValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled   UM3ObjectName
}
```

```
CancelGetObjectGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled       UM3ObjectName
}


CancelGetObjectValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3OperationToBeCancelled       UM3ObjectName
}


CancelEventReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3SessionIdentifier            UM3ObjectName
}


ChangeOfAttributeValueReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName DEFAULT received-session-identifier,
        &parUM3EventReport                  UM3EventReport
}


ConditionDetectedReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName DEFAULT received-session-identifier,
        &parUM3EventReport                  UM3EventReport
}


CreateObject ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3DnObjectName                 UM3CharacterString,
        &parAnyTypeValue                    ANY DEFINED BY UM3 ASN.1 module
}


CreateObjectGroup ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parNoOfObjectsToBeCreated          UM3UnsignedInteger16,
        &parUM3ObjectListToBeCreated        UM3ObjectListToBeCreated
}
```

```
DeleteObject ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier              UM3ClassIdentifier,
        &parUM3ObjectName                   UM3ObjectName
}

DeleteObjectGroup ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectList                   UM3ObjectList
}

GetObjectAttributeGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier              UM3ClassIdentifier,
        &parUM3ObjectName                   UM3ObjectName,
        &parUM3AttributeObjectList          UM3ObjectList
}

GetObjectAttributeValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier              UM3ClassIdentifier,
        &parUM3ObjectName                   UM3ObjectName,
        &parUM3AttributeObjectIdentifier    UM3ClassIdentifier,
        &parUM3AttributeObjectName          UM3ObjectName
}

GetObjectGroupValue ::=
        GetObjectGroupValueOvld1 |
        GetObjectGroupValueOvld2 |
        GetObjectGroupValueOvld3

GetObjectGroupValueOvld1 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectList                   UM3ObjectList
}

GetObjectGroupValueOvld2 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
```

```
        &um3OperationObjectName          UM3ObjectName UNIQUE,
        &parUM3ObjectValueCondition      UM3ObjectValueCondition
}

GetObjectGroupValueOvld3 ::= UM3-OPERATION {
        &um3OperationClassIdentifier     UM3ClassIdentifier,
        &um3OperationObjectName          UM3ObjectName UNIQUE,
        &parUM3ObjectList                UM3ObjectList,
        &parUM3ObjectValueCondition      UM3ObjectValueCondition
}


GetObjectValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier     UM3ClassIdentifier,
        &um3OperationObjectName          UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier           UM3ClassIdentifier,
        &parUM3ObjectName                UM3ObjectName
}


OperationResponse ::=
        OperationResponseSuccessAck |
        OperationResponseSuccessOvld1 |
        OperationResponseSuccessOvld2 |
        OperationResponseFailureOvld3 |
        OperationResponseFailureOvld4 |
        OperationResponseFailureOvld5 |
        OperationResponseFailureOvld6 |
        OperationResponseFailureOvld7 |
        OperationResponseFailureOvld8 |
        OperationResponseFailureOvld9

OperationResponseSuccessAck ::= UM3-OPERATION {
        &um3OperationClassIdentifier     UM3ClassIdentifier,
        &um3OperationObjectName          UM3ObjectName DEFAULT received-session-identifier,
        &parResult                       UM3Boolean DEFAULT TRUE
}

OperationResponseSuccessOvld1 ::= UM3-OPERATION {
        &um3OperationClassIdentifier     UM3ClassIdentifier,
        &um3OperationObjectName          UM3ObjectName DEFAULT received-session-identifier,
        &parResult                       UM3Boolean DEFAULT TRUE,
        &parAnyTypeValue                 ANY DEFINED BY UM3 ASN.1 module
}

OperationResponseSuccessOvld2 ::= UM3-OPERATION {
        &um3OperationClassIdentifier     UM3ClassIdentifier,
        &um3OperationObjectName          UM3ObjectName DEFAULT received-session-identifier,
        &parResult                       UM3Boolean DEFAULT TRUE,
        &parUM3ObjectList                UM3ObjectList
```

```
}

OperationResponseFailureOvld3 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode
}

OperationResponseFailureOvld4 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parErrorElementIndex           UM3UnsignedInteger16
}

OperationResponseFailureOvld5 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parErrorElementIndex           UM3UnsignedInteger16,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
        &parUM3ObjectName               UM3ObjectName
}

OperationResponseFailureOvld6 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parUM3ObjectList               UM3ObjectList
}

OperationResponseFailureOvld7 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
        &parUM3ObjectName               UM3ObjectName
}

OperationResponseFailureOvld8 ::= UM3-OPERATION {
        &um3OperationClassIdentifier    UM3ClassIdentifier,
        &um3OperationObjectName         UM3ObjectName DEFAULT received-session-identifier,
        &parResult                      UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode       UM3OperationErrorCode,
        &parUM3ClassIdentifier          UM3ClassIdentifier,
        &parUM3ObjectName               UM3ObjectName
```

```
        &parUM3AttributeClassIdentifier    UM3ClassIdentifier,
        &parUM3AttributeObjectName         UM3ObjectName
}

OperationResponseFailureOvld9 ::= UM3-OPERATION {
        &um3OperationClassIdentifier       UM3ClassIdentifier,
        &um3OperationObjectName            UM3ObjectName DEFAULT received-session-identifier,
        &parResult                         UM3Boolean DEFAULT FALSE,
        &parUM3OperationErrorCode          UM3OperationErrorCode,
        &parErrorObjectElementIndex        UM3UnsignedInteger16,
        &parErrorAttributeElementIndex     UM3UnsignedInteger16
}


RequestChangeOfAttributeValueReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier       UM3ClassIdentifier,
        &um3OperationObjectName            UM3ObjectName UNIQUE,
        &parUM3ActionBroker                UM3ActionBroker
}


RequestConditionDetectedReport ::= UM3-OPERATION {
        &um3OperationClassIdentifier       UM3ClassIdentifier,
        &um3OperationObjectName            UM3ObjectName UNIQUE,
        &parUM3ActionBroker                UM3ActionBroker
}


SetObjectAttributeGroupValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier       UM3ClassIdentifier,
        &um3OperationObjectName            UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier             UM3ClassIdentifier,
        &parUM3ObjectName                  UM3ObjectName,
        &parUM3ObjectList                  UM3ObjectList
}


SetObjectAttributeValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier       UM3ClassIdentifier,
        &um3OperationObjectName            UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier             UM3ClassIdentifier,
        &parUM3ObjectName                  UM3ObjectName,
        &parUM3AttributeClassIdentifier    UM3ClassIdentifier,
        &parUM3AttributeObjectName         UM3ObjectName,
        &parAnyTypeValue                   ANY DEFINED BY UM3 ASN.1 module
}


SetObjectGroupValue ::=
```

```
        SetObjectGroupValueOvld1 |
        SetObjectGroupValueOvld2 |
        SetObjectGroupValueOvld3

SetObjectGroupValueOvld1 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectDestinationList        UM3ObjectList,
        &parUM3ObjectSourceList             UM3ObjectList
}

SetObjectGroupValueOvld2 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectValueCondition         UM3ObjectValueCondition,
        &parUM3ObjectSourceList             UM3ObjectList
}

SetObjectGroupValueOvld3 ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ObjectDestinationList        UM3ObjectList,
        &parUM3ObjectValueCondition         UM3ObjectValueCondition,
        &parUM3ObjectSourceList             UM3ObjectList
}


SetObjectValue ::= UM3-OPERATION {
        &um3OperationClassIdentifier        UM3ClassIdentifier,
        &um3OperationObjectName             UM3ObjectName UNIQUE,
        &parUM3ClassIdentifier              UM3ClassIdentifier,
        &parUM3ObjectName                   UM3ObjectName,
        &parAnyTypeValue                    ANY DEFINED BY UM3 ASN.1 module
}


 END
```

15. UM3 SER

15.1. Background of UM3 SER Development

UM3 프로토콜의 데이터 송수신을 위한 데이터의 종류는 앞서 기술한 서비스관리 정보모델과 응용서비스 정보모델에서 상세하게 정의하였습니다. 또한 상기 정보모델의 오브젝트들을 주고 받기 위한 오퍼레이션 들에 대한 정의를 통신서비스모델에서 권고하고 있습니다.

The data types of UM3 protocol for data exchange are defined in the details in the service management information model and application service information model that was defined earlier. The recommended operation for exchanging objects of this information model is defined in the communication service model.

이러한 정보모델의 오브젝트들은 통신서비스모델의 오퍼레이션들을 통해 송신되거나 수신될 때 일정한 규칙에 따라 그 값들을 배열하게 됩니다. 이러한 0과 1의 비트들의 배열 규칙을 인코딩 룰 (Encoding Rule)이라 부릅니다. UM3 SER (Simple Encoding Rule) 은 본 권고안이 정의하는 UM3 프로토콜을 위한 인코딩 룰입니다.

Values of objects in this information model are arranged based on certain rules when they are sent or received through the operations of the communication service model. The rule of arranging bits of 0s and 1s is called the encoding rule. UM3 SER (Simple Encoding Rule) is the encoding rule for the UM3 protocol defined in this recommendation.

국제표준화 기구인 ITU-T, CCITT 는 상기 인코딩 룰을 X.209, X.690 등의 권고안에서 정의하고 있으며, 이러한 인코딩 룰에는 BER (Basic Encoding Rule), CER (Canonical Encoding Rule) 등 여러 가지 많은 인코딩 룰들이 있습니다.

International standard bodies like ITU-T and CCITT define the encoding rules in recommendations like X.209 and X.690, including various encoding rules such as BER (Basic Encoding Rule) and CER (Canonical Encoding Rule).

또한 최근의 TCP/IP 응용계층 (Application layer) 의 데이터 송수신을 위한 프로토콜에는 HTTP (Hypertext transport protocol), SOAP (Simple object access protocol) 등이 존재합니다. 특히 SOAP 등은 XML (extensible markup language) 과 HTTP 프로토콜을 이용한 프로토콜로, 웹 서비스 등에 많이 활용되고 있는 상황입니다. 상기 HTTP 와 SOAP 등은 모두 일반적인 텍스트를 기반으로 데이터를 송수신하는 방법으로, 해당 프로토콜을 사용하는 소프트웨어나 혹은 하드웨어를 기획하고 개발하는 사람들에게는 매우 친숙하고 쉬운 인코딩 방식으로 인지되고 있습니다.

The protocols for transfer in the latest TCP/IP Application layer includes HTTP (Hypertext transport protocol), SOAP (Simple object access protocol), and so on. SOAP is especially widely used for protocols using XML (extensible

markup language) and HTTP protocol and web services. All of HTTP and SOAP are text based data exchange methods, and they are considered as convenient and easy encoding methods for people who plan and develop software or hardware systems using those protocols.

위의 텍스트 기반의 인코딩 룰과는 달리 BER, CER, DER 등의 인코딩 룰은 사용자들의 입장에서 볼 때 복잡도가 높으며 친숙도가 매우 떨어지는 형태로 볼 수 있습니다.

Unlike these text based encoding rules, encoding rules like BER, CER, and DER are highly complex and not user friendly from the perspective of users.

그러나 산업현장의 모든 장치들 즉, 센서와 제어장치 들이 모두 텍스트 기반의 프로토콜을 지원하기에는 어려운 점이 있습니다. 또한 이들을 연결하고 처리하는 게이트웨이 장치들도 컴퓨터의 형상을 갖추고는 있으나 그 사양이 현저하게 낮은 경우가 많습니다. 이는 서비스 원가의 절감을 위한 노력의 산물이며, 또한 산업현장의 장치들을 개발하는 사업자의 입장에서도 여전히 텍스트기반의 프로토콜이 큰 매력으로 다가오지 않고 있다는 점을 의미하기도 합니다. 다만, 산업현장의 장치들도 그 컴퓨팅 성능이 날로 커지고 있으며 결국 소프트웨어의 데이터 구조와 그 데이터 구조에 할당된 값을 송수신하기 위한 응용 계층의 프로토콜들은 상당 부분이 XML 과 같은 텍스트 기반의 프로토콜로 그 형태가 변화할 것이 분명해 보입니다. 따라서, UM3 Protocol 또한 바이너리 형식의 인코딩을 정의한 UM3 SER 과 더불어 텍스트 기반의 프로토콜인 XML 형식을 지원하는 형태로 그 구성이 이루어져 있습니다. 이 장에서는 위의 두 가지 방식들 중 첫 번째 방식인 바이너리 형식의 데이터 포맷을 다루고 있는 UM3 SER 을 정의하고 설명합니다.

However, it is difficult for general equipment in the industrial field such as sensors and controllers to support text based protocols. The gateway devices that connect them and process the data are computer like systems, but their performance is quite low more often than not. This is a result of efforts of the cost reduction, and text based protocols are still not attractive to the companies who develop the industrial equipment. The performance of the computers in the industrial equipment, however, is getting more powerful. It is obvious that a significant part of the application layer protocols for transferring data structures of software and the values assigned to those data structures will be changed to text based protocols like XML. Therefore, the Ume protocol can support XML, text based protocol, as well as UM3 SER that supports binary type encoding. In this chapter, UM3 SER, the first method out of the two methods described above, and the way it handles binary data format is defined and explained.

15.2. Major Features of UM3 SER

본 권고안은 UM3 SER 인코딩 룰을 정의함에 있어서 다음과 같은 원칙하에 해당 클래스 타입들에 대한 인코딩 방법을 정의하였습니다. 데이터를 생성하거나 처리하는 컴퓨터의 메모리가 해당 데이터 타입을 어떻게 처리하는가에 중점을 두어 각 클래스 타입들에 대한 인코딩 룰을 정의합니다. 즉, 본 권고안이 정의하는 각각의 타입들이 메모리에 로드 되어 특정한 메모리의 주소로 접근 가능한 공간을

차지하고, 해당 공간에 0과 1의 조합으로 특정한 값이 저장되어 있다면, 해당 메모리의 형상을 그대로 네트워크 인터페이스를 이용하여 다른 컴퓨터로 전송하고, 그 데이터를 수신한 컴퓨터도 해당 데이터를 그대로 메모리에 복사함으로써 정상적인 데이터의 수신과 변수 값의 지정을 완료할 수 있는 형태로 인코딩 룰을 지정하였습니다. 이는 인코딩 룰을 적용하기 위해 2중 3중의 불필요한 인스트럭션 (instruction on CPU) 이 수행되는 낭비요소를 줄이기 위한 목적을 갖고 있습니다.

The encoding methods for the following class types are defined with the following principles for the UM3 SER encoding rule defined in this recommendation. The encoding rule is defined for various class types with the focus on how the computer memory that creates or processes the data handles the corresponding data types. When each type defined in this recommendation is loaded in the memory, occupying a certain space that can be accessed by the memory address and specific values comprised of 1s and 0s are stored in the corresponding space, the encoding rule is defined in such a way that the shape of the memory for the corresponding data is transferred to another computer through the network interface. The computer receiving the data just copies the corresponding data into the memory, which completes the normal data transfer and assignment of variables. The purpose of this concept is to reduce wasted resources by avoiding multiple instructions to the CPU to apply the encoding rule.

또한 본 권고안이 정의하는 UM3CharacterString 등 문자열의 인코딩은, 현재 산업현장에서 사용하는 게이트웨이 등이 대부분 ASCII 코드 체계를 사용하고 있으므로 이를 충실하게 반영하고자 노력하였습니다. 또한 본 권고안이 적용되는 서비스의 범위가 1차적으로는 본 권고안을 제정한 ㈜케이티의 서비스이므로, ㈜케이티와의 협력관계에 있는 모든 협력사들의 운용환경 등에 대한 광범위한 조사 및 분석 후 결정된 사항입니다.

In case of character array encoding like UM3CharacterString defined in this recommendation, the ASCII code system is widely used in most systems including gateway devices used in the industrial field, and efforts have been made to reflect it as much as possible. Since the primary scope of the services to which this recommendation applies are services of KT that are established this recommendation, this was determined after a wide range of investigation and analysis about the operation environment of all partner companies in collaboration with KT.

본 권고안이 정의하는 UM3 SER은 ITU-T X.690 의 BER 등과 비교할 경우 다음과 같은 차이점을 갖고 있습니다. UM3 SER 은 태그 (Tag) 라는 요소를 사용하되 UM3ObjectName 타입의 애트리뷰트 값을 이용해 하나의 오브젝트 혹은 여러 개의 오브젝트 애트리뷰트 들로부터 해당 애트리뷰트를 구분하는 두 가지 방법을 병행해서 사용할 수 있습니다. 위의 두 번째 방법은 특히 적정 수준 이상의 메모리와 처리속도를 갖춘 컴퓨터에서 프로토콜의 구현을 매우 쉽게 할 수 있게하는 장점을 갖고 있습니다. 다만 UM3 SER 은 X.208, X.209, X.680, X.690 등이 정의하고 있는 ‘UNIVERSAL’, ‘APPLICATION’ 등의 TAG CLASS 를 구분하지 않으며, 각 오브젝트를 구성하고 있는 애트리뷰트의 이름을 이용하여 해당 애트리뷰트를 구분합니다. 만약 굳이 애트리뷰트의 이름을 이용하여 구분하는 것이 불필요할 경우 이름을 저장하는 필드를 생략하는 방법을 사용합니다.

UM3 SER defined in this recommendation has the following differences when it is compared with BER like ITU-T X.690. UM3 SER uses the Tag element, but two methods of identifying the target attribute from one object or one or more objects attributes by using the attribute values of UM3ObjectName type can be used in parallel. The second method has the advantage in terms of ease of implementation of protocol the computers that have more than a certain amount of memory and processing speed. UM3SER, however, does not classify the TAG CLASS such as 'UNIVERSAL' and 'APPLICATION' that are defined in the X.208, X.209, X.680, and X.690, and the corresponding attribute is identified by using the name of the attribute of each object. If it is not necessary to identify the attribute by using a name, then the field for storing this name can be removed.

15.3. Configuration of UM3 APDU

본 권고안이 정의하는 UM3 프로토콜은 T-NL-N-L-V 라는 형식의 독특한 APDU 구조를 갖고 있습니다. 이 때 T-NL-N-L 필드를 APDU의 헤더 (Header) 로 정의하며 V 필드를 APDU의 바디 (Body) 로 정의합니다. 그림 25-UM3 프로토콜의 APDU 구성 [는 UM3 프로토콜의 APDU의 구성을 나타내고 있습니다.

The UM3 protocol defined in this recommendation has a unique APDU structure called T-NL-N-L-V type, where the T-NL-N-L field is defined as the header of APDU, and the V field is defined as the body of APDU. Fig. 25 shows the configuration of the UM3 protocol.

T 필드는 타입 혹은 UM3 클래스 타입을 나타내는 필드입니다. T 필드에는 정보모델을 구성하고 있는 클래스 타입들의 클래스 아이디가 기록됩니다. 또한 통신서비스모델을 구성하고 있는 오퍼레이션 클래스 타입들의 클래스 아이디가 기록됩니다. 정보모델 혹은 통신서비스모델의 클래스아이디의 타입은 UNSIGNED INTEGER 타입이며 모두 4 바이트의 길이를 갖게 됩니다.

The T field represents the type or UM3 class type. The class ID of the class type configuring the information mode is stored in the T field. The class ID of operation class type configuring the communication service model is also stored. The type of class ID of the information model or communication service model is UNSIGNED INTEGER with four bytes of data.

NL 필드는 NL 필드 다음에 위치하는 N 필드의 길이정보를 기록합니다. NL 필드는 UNSIGNED SHORT INTEGER 타입으로 총 2 바이트의 길이를 갖게 됩니다.

The NL field contains the length of the N field that is next to the NL field. The NL field is UNSIGNED SHORT INTEGER type with two bytes of data.

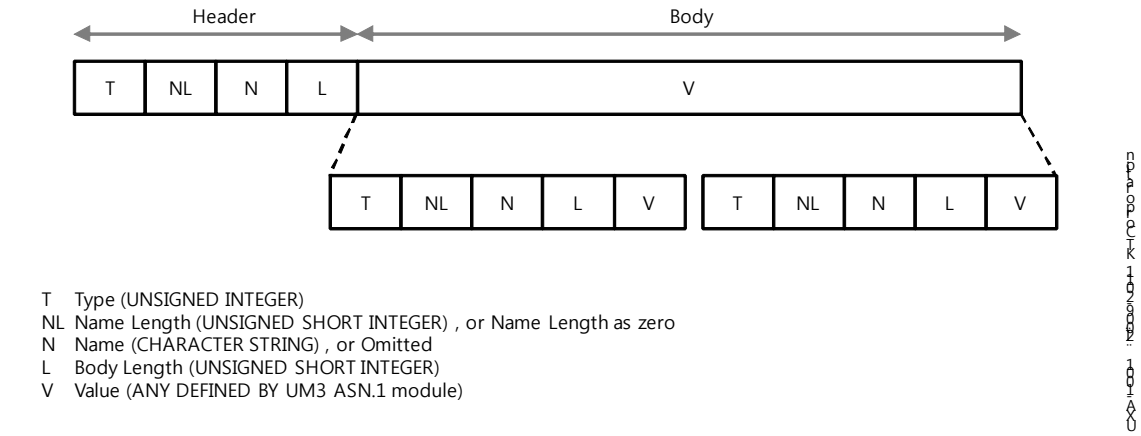


그림 25-UM3 프로토콜의 APDU 구성 [Configuration of UM3 protocol]

N 필드는 위에서 설명한 바와 같이 해당 APDU가 나타내는 오브젝트의 이름을 기록합니다. 즉, T 필드의 값은 클래스의 종류를 구분하기 위한 목적으로 사용되며, N 필드는 해당 클래스에 속하는 오브젝트들을 그 이름으로 구분하기 위해 사용됩니다. 만약, 해당 N 필드가 아래에서 설명하는 V 필드 내부에 존재하는 APDU의 N 필드라면, 이는 특정 오브젝트의 애트리뷰트의 이름을 나타내게 됩니다.

The N field contains the name of object that is pointed by the corresponding APDU as explained above. In other words, the value of the T field is used for the purpose of identification of the class type, and that of the N field is used to identify the object under the class by name. If the corresponding N field is the N field of APDU in the V field that is explained below, then this represents the name of the attribute of a specific object.

L 필드는 그 다음에 위치하는 V 필드의 길이정보를 기록합니다. 그 크기는 2 바이트 크기이며 NL 필드와 마찬가지로 UNSIGNED SHORT INTEGER 타입으로 기록합니다. 만약 해당 APDU의 V 필드가 통신서비스모델의 V 필드일 경우, 이는 해당 오퍼레이션의 파라메터가 기록되어 있음을 뜻하게 됩니다.

The L field contains the length of the V field that is located next to it. The size is two bytes, and it is UNSIGNED SHORT INTEGER like the NL field. If the V field of the corresponding APDU is the V field of the communication service model, then it means that the parameter of the corresponding operation is stored.

마지막으로 V 필드는 본 권고안에 정의되어 있는 서비스관리 정보모델과 응용서비스 정보모델의 모든 클래스들이 기록될 수 있습니다. V 필드는 하나의 클래스 타입의 오브젝트로 구성될 수도 있으며, 경우에 따라 여러 개의 APDU가 나열된 복합형식의 오브젝트들로 구성될 수도 있습니다.

Lastly, any classes of the service management information model and application service information model defined in this recommendation can be stored in the V field. The V field can be configured with an object of one class type, and it can also be configured as a complex format type object where multiple APDU are listed depending on the cases.

앞서 정의한 바와 같이 본 권고안이 정의하는 APDU 의 T, NL, N, L 필드를 헤더 (header) 로 정의하고, 나머지 V 필드를 바디 (body) 로 정의합니다.

As defined earlier, the T, NL, N, L field of the APDU defined in this recommendation is defined as the header, and the V field is defined as the body.

NL 필드의 값이 0 일 경우에는 N 필드의 길이가 0 이므로 N 필드의 값이 없는 상태 즉, 애트리뷰트 혹은 파라메터의 이름을 별도로 명기하지 않고 UM3 APDU 바디부분의 데이터를 송신하는 것으로 정의합니다. 즉, UM3 APDU 의 바디 부분을 위한 오브젝트의 이름을 송수신하는 형식을 2 가지로 정의합니다. 단, 본 권고안은 위의 2 가지 방식들 중 UM3 SER 인코딩 룰을 완전하게 지원하는 첫 번째 방법을 사용토록 권고합니다.

If the value of NL the field is 0, then the length of the N field is 0. In this case, the data in the UM3 APDU body is transferred without specifying the name of attribute or parameter i.e. without the value of the N field. Two methods of transferring object name for body part of UM3 APDU are defined. In this recommendation, the first method that fully supports the UM3 SER encoding rule among these two methods is recommended.

위의 UM3 프로토콜의 APDU에 대한 ASN.1 정규표현식은 다음과 같습니다.

APDU of UM3 can be defined by using ASN.1 regular expressions as follows.

```
UM3 APDU ::= UM3 SEQUENCE {
        Type                    UNSIGNED INTEGER,
        nameFieldValue          NameFieldType,
        Length                  UNSIGNED SHORT INTEGER,
        Value                   ANY DEFINED BY UM3 ASN.1 module
}

NameFieldType ::=
        NoName |
        CharacterStringName |
        UnsignedShortTag
}

NoName ::= nameLength (0)

CharacterStringName ::= UM3 SEQUENCE {
        nameLength              UNSIGNED SHORT INTEGER (1…65534),
        Name                    CHARACTER STRING
}
```

upper-bound UNSIGNED SHORT INTEGER ::= 65535

UnsignedShortTag ::= UM3 SEQUENCE {
nameLength UNSIGNED SHORT INTEGER (upper-bound | upper-bound <)
um3Tag UNSIGNED SHORT INTEGER
}

마찬가지로 상기 UM3 APDU 타입의 정의에 있어서도 ITU-T/CCITT 가 정의한 CLASS 와 그 인코딩 방법은 사용하지 않습니다.

The class and encoding method defined in ITU-T/CCITT is not used in the definition of UM3 APDU type for the same reason.

본 권고안이 정의하는 UM3 SER 이 N, NL 필드의 인코딩 방법을 2 가지의 조합으로 정의하는 이유는 경우에 따라 패킷을 구성하는 byte 의 크기가 통신을 위한 비용에 영향을 크게 미치는 경우, 해당 서비스의 원가상승 요인으로 작용할 수도 있기 때문입니다. 즉, 종량제 방식의 무선 인터넷 환경을 사용하는 경우, 대부분 정액제로 요금이 부과되는 유선환경과는 달리 송수신 데이터의 규모에 따라 비용이 더 투입되어야 할 수도 있기 때문입니다. 더불어 본 권고안이 정의하고 있는 서비스관리정보 모델의 클래스 타입들 중에는 UM3 SEQUENCE OF, UM3 SET OF, UM3 SEQUENCE, UM3 SET 등과 같이 UM3Base 클래스로부터 그 특성을 상속받지 않은 복합형식의 타입들이 있습니다. 이러한 복합 형식의 타입들은 별도의 오브젝트 이름을 필요로 하지 않는 경우도 있으며 이를 위해 위에서 언급한 2 가지의 인코딩 방식들 중 N 필드를 사용하지 않는 인코딩 방식을 사용하기도 합니다.

The reason why the encoding method of the N and NL field is defined as the combination of two types is that the size of bytes that configures the packet can have a significant impact on the communication cost in some cases, and it could be a cost increase factor of the corresponding service. In case of using a wireless internet environment with a pay-per-byte rate, there is fixed fee most of the time, but a sometimes higher cost can result depending on the size of data transfer, unlike internet over cable. Besides, there are complex format type classes that do not inherit the characteristics from the UM3Base class such as UM3 SEQUENCE OF, UM3 SET OF, UM3 SEQUENCE, and UM3 SET among the class types of the service management information model defined in this recommendation. These complex format types may not require object names sometimes, and the encoding method that does not use the N field among the two methods mentioned earlier may be used for this reason.

그러나 앞서 권고한 바와 같이 UM3 APDU 의 N, NL 필드의 인코딩 방법은 반드시 첫 번째 방법 즉, 정상적인 애트리뷰트 오브젝트의 이름과 그 이름의 길이를 나타내는 방식을 사용토록 권고합니다.

The first method using the name and length of name for normal attribute objects is recommended in encoding the N and NL field of UM3 APDU as recommended earlier.

최근의 XML 형식의 데이터 송수신 방식의 경우 UM3 SER 방식의 데이터 송수신 방식에 비해 현저하게 많은 양의 데이터를 주고 받게 됩니다. 이는 본 권고안이 정의하는 XML 방식의 인코딩 및 디코딩 방식에서도 확인할 수 있습니다. 또한 NL 과 N 필드에 저장되는 오브젝트 혹은 애트리뷰트의 이름을 생략한다는 것은 해당 필드의 값들을 V 필드에 저장하여 전송하는 것과 동일한 의미이기도 합니다. 즉, RDN 혹은 DN 으로 오브젝트 혹은 애트리뷰트를 구분해야 하는 상황이라면, 해당 오브젝트 혹은 애트리뷰트의 이름을 전송하거나 수신하는 것도 피할 수 없는 과정이라는 것을 명확하게 인지해야 하는 것입니다. 이와 같은 관점에서 볼 때, NL 필드와 N 필드는 송수신하는 APDU 의 인코딩 혹은 디코딩의 연산속도를 더욱 높여 줄 수 있는 장점을 제공하게 됩니다. 또한 텍스트 처리를 통해 해당 필드의 내용을 손 쉽게 확인할 수 있으므로 오히려 이를 APDU의 헤더부에 위치시켜 우선적으로 처리토록 함으로써 더욱 빠르고 손쉬운 데이터 인코딩과 디코딩이 가능한 장점을 제공하게 됩니다

As opposed to the latest XML type data transfer, UM3 SER type data transfer requires significantly lower bandwidth. This can be checked from the XML type encoding and decoding method defined in this recommendation. The name of the object and attribute stored in the NL and N field is also skipped, which has the same meaning as the data transfer by storing the values of the corresponding field in the V field for data transfer. If the object or attribute should be identified using RDN or DN, then it should be clearly recognized that transmission or reception of the name of the corresponding object or attribute cannot be avoided. From this perspective, the NL field and N field provide an advantage of faster encoding and decoding speed of APDU. As the contents of the corresponding field can be easily checked through text processing, they can be put in the header part of APDU for processing at higher priority, which provides the advantage of faster and easier encoding and decoding.

15.4. Encoding of UM3Integer16 type

이 절에서 정의하는 UM3Integer16 타입은 UM3 기본 타입입니다. UM3Integer16은 2 바이트 크기의 메모리를 사용하여 정수를 표현하며, UM3 SER도 2 바이트 크기로 해당 타입을 인코딩 합니다.

UM3Integer16 type defined in this section is UM3 primitive type. UM3Integer16 uses two bytes of memory to represent integer numbers, and UM3 SER encodes this type with the size of two bytes.

대부분의 컴퓨터는 양의 정수(integer)를 표현할 때는 LSB (Least significant bit)를 2^0로 하여 MSB (Most significant bit)까지의 2진수로 그 값을 메모리에 기록합니다. 음의 정수를 표현할 때는 대부분의 컴퓨터가 2의 보수 (2's complement)의 형태로 음의 정수를 표현합니다.

In most computers, positive integers are implemented as binary numbers with LSB as 2^0 and all bits up to MSB, and they are stored in the computer memory. Most computers implement negative integer in two's complement format.

그림 26-음의 정수 16의 2의 보수 [은 2 바이트의 길이를 갖는 UM3Integer16 타입의 음의 정수 16을 2의 보수로 표현하는 방식입니다. 따라서 UM3Integer16 타입의 -16 의 값을 갖는 정수의 경우 이에 대한 인코딩 결과는 그림 27-UM3Integer16 타입의 인코딩과 같습니다.

Fig 26 shows how a negative integer with a value of -16 of UM3Integer16 type with two bytes of data is represented in two's complement format. And Fig 27 show hows this number is encoded.

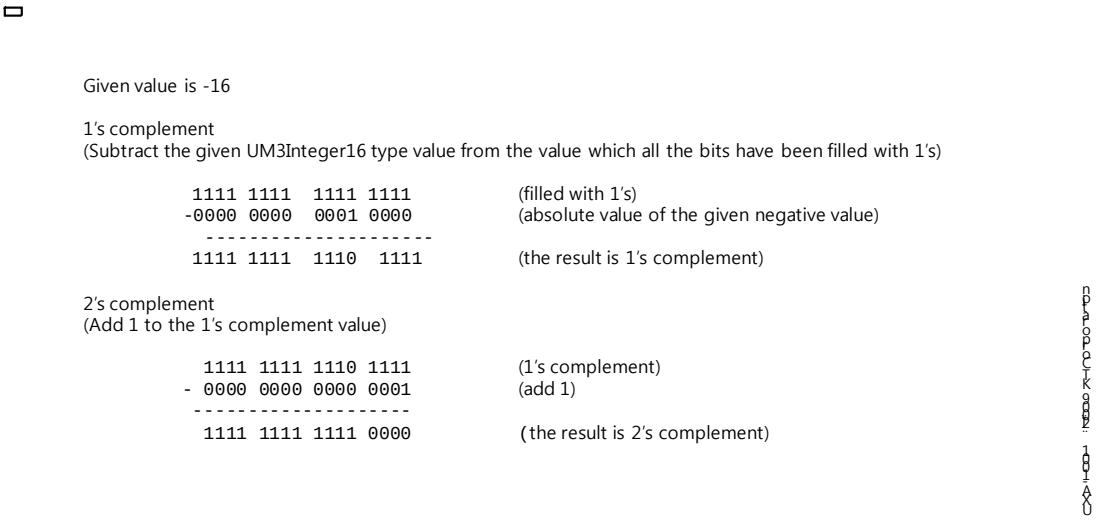


그림 26-음의 정수 16의 2의 보수 [Two's complement of negative number -16]

앞서 정의한 바와 같이 N 필드는 오브젝트의 이름을 기록하는 필드입니다. 그러나 경우에 따라 해당 오브젝트의 이름을 전송하지 않아도 되는 경우, 해당 필드는 생략할 수 있습니다. 그러나 이 때 반드시 NL 필드의 길이를 0_{10}으로 지정하여 N 필드가 사용되지 않음을 표기해야 합니다.

The N field is the field for storing the name of an object as defined earlier. This field can be skipped if there is no need to transfer the name of the corresponding object. The length of the NL field, however, should be set to 0_{10} to indicate that the N field is not used in this case.

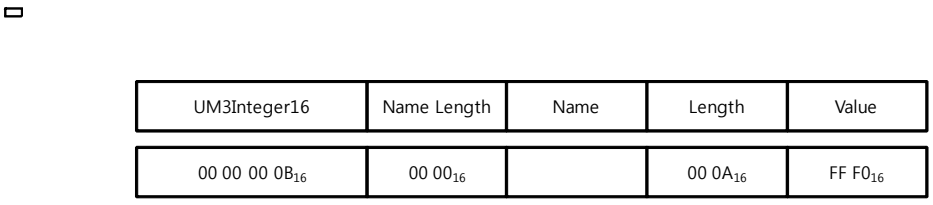


그림 27-UM3Integer16 타입의 인코딩 [Encoding of UM3Integer16 type]

앞서 정의한 바와 같이 대부분의 컴퓨터는 양의 정수와 음의 정수를 상기와 같은 형태로 표현합니다. 그러나, 컴퓨터에 따라 정수를 구성하는 바이트의 배열의 순서를 거꾸로 하는 경우가 있습니다. 이렇게 바이트의 배열 순서를 거꾸로 하는 경우를 little-endian 이라고 부르며, Intel®과 DEC®의 ALPHA® 프로세서가 이러한 방식을 채택하고 있습니다. UM3 SER은 정수를 인코딩 할 때 big-endian 방식을 사용합니다. 즉, Intel® 프로세서와 해당 프로세서를 기반으로 운용되는 운영체제를 사용할 경우 반드시 인코딩 전에 HostToNetworkByteOrder() 혹은 htons() 와 같은 함수를 이용하여 바이트의 순서를 big-endian 으로 변경해주어야 합니다.

As defined earlier, most computers represent positive and negative integers with the method shown above. Some computers, however, reverse the order of byte array configuring integer number, which is called the little-endian method. Intel®, ALPHA®, and DEC® processors use this method. The big-endian method is used for encoding integers in UM3 SER. If the operating system running an Intel® processor or corresponding processor is used, then the order of bytes should be changed to big-endian before encoding by using the functions like HostToNetworkByteOrder() or htons().

UM3Integer16 타입의 정수를 UM3 SER 방식으로 인코딩하기 위한 소프트웨어 프로그램의 예제는 부록 1을 참고 하십시오. 또한 본 권고안이 정의하는 클래스 타입의 클래스 아이디 값과, 이후 정의되는 UM3 SER 의 예에서 나타내는 클래스 아이디는 일치하지 않을 수 있습니다.

Refer to Appendix 1 for a program example of the software to encode UM3Integer16 type integers with the UM3 SER method. The class ID of the class type defined in this recommendation and the class ID shown in the UM3 SER example that is defined later may not match.

15.5. Encoding of UM3Integer32 type

UM3Integer32 타입의 인코딩은 4 바이트 길이의 데이터에 대해, UM3Integer16 타입의 인코딩과 동일한 방식으로 이루어집니다. UM3Integer32 타입은 UM3 기본 타입입니다.

Encoding of UM3Integer32 type is performed for the four bytes of data, and it is done in the same way as the encoding of UM3Integer16 type. UM3Integer32 type is UM3 primitive type.

15.6. Encoding of UM3Integer64 type

UM3Integer64 타입의 인코딩은 8 바이트 길이의 데이터에 대해, UM3Integer16 타입의 인코딩과 동일한 방식으로 이루어집니다. UM3Integer64 타입은 UM3 기본 타입입니다.

Encoding of UM3Integer32 type is performed for the eight bytes of data, and it is done in the same way as the encoding of UM3Integer16 type. UM3Integer64 type is UM3 primitive type.

15.7. Encoding of UM3UnsignedInteger16 type

UM3UnsignedInteger16 타입의 인코딩은 2 바이트 길이의 데이터에 대해, UM3Integer16 타입의 양의 정수에 대한 인코딩과 동일한 방식으로 이루어 집니다. UM3UnsignedInteger16 타입은 UM3 기본 타입입니다.

Encoding of UM3UnsignedInteger16 type is performed for the two bytes of data, and it is done in the same way as the encoding of positive integer of UM3Integer16 type. UM3UnsignedInteger16 type is UM3 primitive type.

15.8. Encoding of UM3UnsignedInteger32 type

UM3UnsignedInteger32 타입의 인코딩은 4 바이트 길이의 데이터에 대해, UM3Integer16 타입의 양의 정수에 대한 인코딩과 동일한 방식으로 이루어 집니다. UM3UnsignedInteger32 타입은 UM3 기본 타입입니다.

Encoding of UM3UnsignedInteger32 type is performed for the four bytes of data, and it is done in the same way as the encoding of positive integer of UM3Integer16 type. UM3UnsignedInteger32 type is UM3 primitive type.

15.9. Encoding of UM3UnsignedInteger64 type

UM3UnsignedInteger64 타입의 인코딩은 8 바이트 길이의 데이터에 대해, UM3Integer16 타입의 양의 정수에 대한 인코딩과 동일한 방식으로 이루어 집니다. UM3UnsignedInteger64 타입은 UM3 기본 타입입니다.

Encoding of UM3UnsignedInteger64 type is performed for the eight bytes of data, and it is done in the same way as the encoding of positive integer of UM3Integer16 type. UM3UnsignedInteger64 type is UM3 primitive type.

15.10. Encoding of UM3CharacterString type

UM3CharacterString 타입은 UM3 기본 타입입니다. UM3CharacterString은 null terminated 문자열로 그 끝을 '\0' 혹은 0x00의 값으로 표시하는 문자열입니다. 대부분의 컴퓨터에 있어서 해당 문자열은 ASCII 코드의 형태로 해당 문자열을 표현합니다. 즉, EBCIDIC 코드체계 등 ASCII 코드 이외의 코드 체계를 사용하는 컴퓨터는 UM3CharacterString 을 인코딩 하기 위해서는 문자열의 모든 문자를 ASCII 코드로 변경해야 합니다.

UM3CharacterString type is UM3 primitive type. UM3CharacterString is a null terminated character array that specifies the end of string with a null character with the value of '\0' or 0x00. In most computers, this character array is expressed in ASCII code. To encode the UM3CharacterString that uses other codes like the EBCIDIC code system besides the ASCII code, all characters in the array should be converted to the ASCII code.

UM3CharacterString 타입의 문자열에 대한 인코딩 방식은 해당 문자열을 구성하고 있는 문자의 ASCII 코드 값을 그대로 V 필드에 기록하는 것으로 완료됩니다. 또한 L 필드에는 해당 문자열의 길이를 UNSIGNED SHORT INTEGER 타입으로 기록합니다. 단, L 필드에 저장되는 값은 상기 문자열의 마지막을 나타내는 '\0' 혹은 0x00의 값을 제외한 길이이어야 합니다.

Encoding of UM3CharacterString type strings is performed by simply copying the ASCII code values of the characters in the string in the V field without modification. The length of the string is stored in the L field as UNSIGNED SHORT INTEGER type. The value written in the L field should be the length without the string termination character of '\0' or 0x00.

그림 28-UM3CharacterString 타입의 인코딩은 UM3CharacterString 타입의 인코딩 결과를 보여주는 예제입니다. 그림 28-UM3CharacterString 타입의 인코딩의 L 필드는 V 필드의 문자열 "Hello, world!"의 길이인 13_{10} 즉, $0D_{16}$가 기록되어 있으며, V 필드에는 문자열의 ASCII 코드 값이 기록되어 있습니다. 앞서 정의한 UM3Integer16 타입의 인코딩 예와는 달리 N 필드에는 해당 오브젝트의 이름인 "um3Name" 문자열이 기록되어 있으며, NL 필드에는 N 필드의 길이를 나타내는 7_{10}이 2진수로 기록되어 있습니다.

그림 28-UM3CharacterString 타입의 인코딩 is an example encoding result of UM3CharacterString type. The L field contains the length of string in the V field "Hello, world!", which is 13_{10} or $0D_{16}$, and the V field contains the ASCII code values of the string. Unlike the encoding example of UM3Integer16 type, the N field contains the "um3Name" string as the object name of the corresponding string, and the NL field contains the length of the N field, which is 7_{10} in binary number.

또한 T 필드에는 UM3CharacterString 타입을 나타내는 클래스 아이디가 4 바이트 길이의 UNSIGNED INTEGER 값으로 기록되어 있습니다.

The T field contains the class ID that represents the UM3CharacterString type with the value of UNSIGNED INTEGER and the length of four bytes.

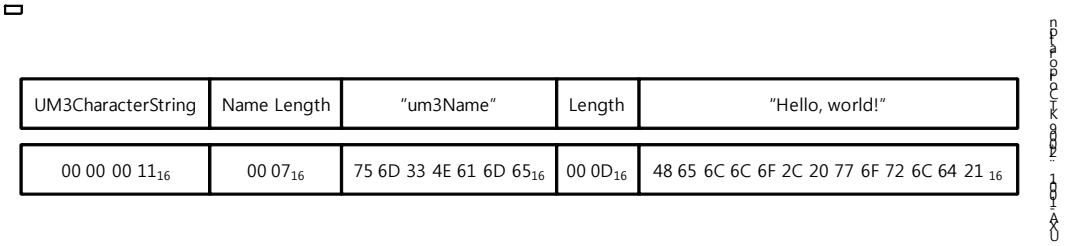


그림 28-UM3CharacterString 타입의 인코딩 [Encoding of UM3CharacterString type]

15.11. Encoding of UM3BitString type

UM3BitString 타입은 UM3 기본 타입 입니다. UM3BitString 타입의 인코딩 방식은 앞서 정의한 UM3CharacterString, 타입의 인코딩 방식과 동일합니다. 즉, T 필드에는 UM3BitString 클래스 아이디 값을 기록하고, NL과 N 필드에도 해당 값을 기록합니다. L 필드에는 V 필드의 길이를 기록하고, V 필드에는 주어진 바이트를 8 비트 단위 (Octet)로 수정 없이 그대로 기록합니다.

UM3BitString type is UM3 primitive type. The encoding method of UM3BitString type is the same as that of UM3CharacterString type that was defined earlier. The UM3BitString class ID is recorded in the T field. The NL and N field also contain the corresponding values. The L field contains the length of the V field, and the provided bytes are stored in the V field in 8 bit units (Octet) without modification.

UM3BitString 타입의 데이터는 1 바이트 단위의 의미 있는 데이터들의 집합으로 볼 수 있습니다. UM3 프로토콜은 해당 데이터 타입의 각 바이트들 혹은 비트들이 갖고 있는 의미에 상관없이 데이터의 송수신을 위한 APDU의 관점에서만 데이터를 처리합니다.

UM3BitString type data can be considered as the set of data with the meaning in one byte units. UM3 protocol processes the data from the perspective of APDU for data transfer independently from the meaning of each byte or bit.

UM3BitString	Name Length	"byte"	Length	11111111 11111111 11111111 11111111
$00\ 00\ 00\ 12_{16}$	$00\ 04_{16}$	$62\ 79\ 74\ 65_{16}$	$00\ 04_{16}$	$FF\ FF\ FF\ FF_{16}$

그림 29-UM3BitString 타입의 인코딩 [Encoding of UM3BitString type]

15.12. Encoding of UM3OctetString type

UM3OctetString 타입은 UM3 기본 타입입니다. OCTET STRING 의 인코딩 또한 CHARACTER STRING 타입과 동일한 방식으로 인코딩이 이루어집니다. 다만 데이터의 값들이 컴퓨터로 프린트 할 수 있으며 사람이 에디터 등을 통해 확인할 수 있는 문자열이 아닌, 다른 목적을 갖는 8 비트 단위의 데이터를 표현하기 위해 사용합니다.

UM3OctetString type is UM3 primitive type. Encoding of Octet string is performed in the same way as the

CHARACTER STRING type. It is usually used to represent the data in 8 bit units that is not the string that can be printed or checked by a human or an editor program.

OCTET STRING의 인코딩은 T 필드에 UM3OctetString 타입의 클래스 아이디를 기록하는 것을 제외하고 나머지 필드에 대한 인코딩은 UM3CharacterString 과 동일한 방식으로 인코딩이 이루어집니다.

The class ID of UM3OctetString type is stored in the T field, and encoding of other fields for OCTET STRING is performed in the same ways as the encoding for UM3CharacterString.

15.13. Encoding of UM3Boolean type

UM3Boolean 타입은 UM3 기본 타입입니다. UM3Boolean 타입은 참 (True) 혹은 거짓 (False) 등 둘 중의 하나의 값으로 지정되며, 1 바이트 길이의 문자로 그 값을 기록합니다. 참일 경우에는 FF_{16}의 값으로 기록하고 거짓일 경우에는 00_{16}의 값으로 V 필드의 값을 기록합니다.

UM3Boolean type is UM3 primitive type. The UM3Boolean type value is assigned by a True or False value, and a one byte character is used to store the data. It is stored as FF_{16} in the V field if the value is True and 00_{16} if it is false.

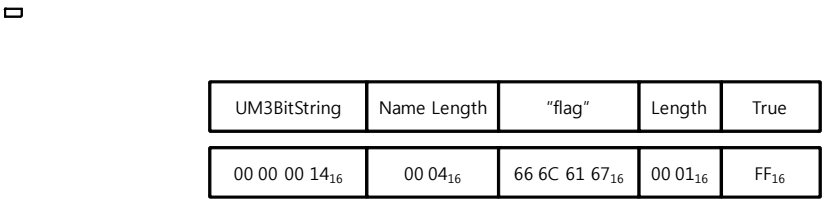


그림 30-UM3Boolean 타입의 인코딩 [Encoding of UM3Boolean type]

15.14. Encoding of UM3Enumerated type

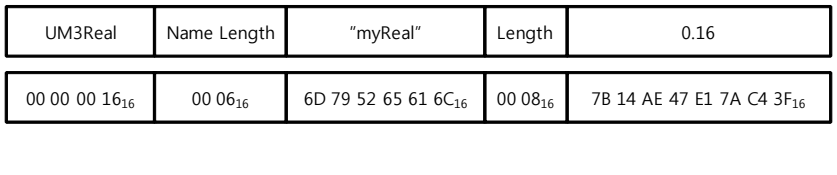
UM3Enumerated 타입의 인코딩은 UM3Integer32 타입의 인코딩과 동일한 방식으로 이루어집니다. UM3Enumerated 타입은 UM3 기본 타입 입니다.

Encoding of UM3Enumerated type is performed in the same way as the encoding of UM3Integer32 type. UM3Enumerated type is UM3 primitive type.

15.15. Encoding of UM3Real type

UM3Real 타입은 UM3 기본 타입입니다. UM3Real 타입은 일반적인 컴퓨터가 사용하는 부동소수점 방식 (floating point)으로 데이터를 기록합니다. UM3Real 타입의 8 바이트를 구성하는 각 비트 별 의미는 IEEE754 권고안의 기준을 따르며 double precision 방식의 비트 구성을 사용합니다. 즉, 비트 64는 양수 혹은 음수를 나타내는 부호를 가리키며, 다음 비트 63에서 비트 53까지의 11개의 비트는 지수부 (exponent) 를, 그리고 비트 52 부터 비트 1 까지의 52개 비트는 가수부 (significand) 를 나타냅니다.

UM3Real type is UM3 primitive type. UM3Real type stores the floating point data that are used by general computers. UM3Real type uses 8 bytes. The IEEE754 recommendation is followed for definition of each bit, and a double precision bit configuration is used. Bit 64 indicates the sign of the number, 11 bits from bit 63 to bit 53 are exponents, and 52 bits from bit 52 to bit 1 is a singificand**.**



UM3Real	Name Length	"myReal"	Length	0.16
00 00 00 16_{16}	00 06_{16}	6D 79 52 65 61 $6C_{16}$	00 08_{16}	7B 14 AE 47 E1 7A C4 $3F_{16}$

그림 31-UM3Real 타입의 인코딩 [Encoding of UM3Real type]

실질적인 구현에 있어서는 대부분의 컴퓨터가 double 타입의 변수를 나타낼 때 상기 double precision 방식으로 데이터를 저장하므로, 해당 변수의 값을 8 바이트의 데이터로 변환하여 송신하고, 수신 측에서는 해당 8 바이트의 데이터를 바로 double 타입의 변수에 입력하는 것으로 간단하게 인코딩 및 디코딩 (decoding) 작업이 완료되게 됩니다.

Since most computers implement the double type variables with the double precision type described above, encoding and decoding tasks simply convert the values of the corresponding variable to 8 bytes of data for transmission, and the received 8 bytes are stored into double type variables directly.

15.16. Encoding of UM3Null type

UM3Null 타입은 UM3 기본 타입 입니다. UM3Null 타입은 1 바이트의 상수 값 00_{16}으로 V 필드의 값을 표현합니다.

UM3Null type is UM3 primitive type. UM3Null type represents the value of the V field with one byte constant of

00_{16}.

15.17. Encoding of UM3DateTime type

UM3DateTime 타입은 UM3 기본 타입입니다. UM3DateTime 타입은 UM3Integer32 타입의 인코딩과 동일한 인코딩 방식을 갖습니다.

UM3DateTime type is UM3 primitive type. Encoding of UM3DateTime type is performed in the same way as the encoding of UM3Integer32 type.

15.18. Encoding of UM3ClassIdentifier type

UM3ClassIdentifier 타입은 UM3 기본 타입 입니다. UM3ClassIdentifier 타입의 인코딩은 UM3UnsignedInteger32 타입과 동일한 방식으로 인코딩이 이루어 집니다.

UM3ClassIdentifier type is UM3 primitive type. Encoding of UM3ClassIdentifier type is performed in the same way as the encoding of UM3UnsignedInteger32 type.

15.19. Encoding of UM3ObjectName type

UM3ObjectName 타입은 UM3 기본 타입 입니다. UM3ObjectName 타입의 인코딩은 UM3CharacterString 타입과 동일한 방식으로 인코딩이 이루어 집니다.

UM3ObjectName type is UM3 primitive type. Encoding of UM3ObjectName type is performed in the same way as the encoding of UM3CharacterString type.

15.20. Encoding of UM3 SET type

UM3 SET 타입은 UM3 복합형식 타입입니다. UM3 SET 타입의 UM3 SER 인코딩은 다음과 같은 형태로 이루어집니다. UM3 SET 타입을 구성하는 요소를 엘리먼트로 정의할 때, 각 엘리먼트를 UM3 SER 인코딩 룰을 적용하여 인코딩하고, 해당 인코딩의 결과인 엘리먼트 APDU 들을 UM3 SET 타입의 인코딩 결과의 body 에 기록하는 것으로 정의합니다. UM3 SET 타입의 header 에 대한 인코딩은 UM3 SER 인코딩 룰이 정의하는 인코딩 방식과 동일한 방식으로 인코딩이 이루어집니다.

예를 들어 임의의 UM3 SET 타입이 아래와 같이 정의되어 있다고 가정합니다.

UM3 SET type is UM3 complex format type. UM3 SER encoding of UM3 SET type is performed as follows. When components of the UM3 SET type are defined as elements, the UM3 SER encoding rule is applied to each element for encoding. The element APDU as a result of encoding is stored in the body of the encoding result of UM3 SET type. Encoding of the header of UM3 SET type is performed in the same way as the encoding method defined in the

UM3 SER encoding rule.

For example, let's assume that a UM3 SET type is defined as follows.

```
MyUM3SetType ::= SET {
        nameOfPerson            UM3CharacterString,
        phoneNumber             UM3CharacterString,
        age                     UM3UnsignedInteger32,
        category                UM3Enumerated {
                                    friendly (0),
                                    hostile (1),
                                    none (2)
                                }
}
```

즉, MyUM3SetType 이라는 이름의 타입은 nameOfPerson, phoneNumber, age, category 라는 이름의 변수들의 집합으로 이루어져 있습니다. 각 변수의 타입들은 UM3CharacterString, UM3UnsignedInteger32, UM3Enumerated 등의 UM3 기본 타입으로 정의되어 있습니다. 변수 category 는 UM3Enumerated 타입이며 friendly 값은 정수 0, hostile 값은 정수 1, none 값은 정수 2 로 맵핑 되도록 정의되어 있습니다.

이상과 같은 형태로 정의된 MyUM3SetType 타입의 변수를 myUM3SetValue 라고 정의하고 각 변수의 값들이 다음과 같이 지정되어 있다고 가정합니다.

The MyUM3SetType type is a set of variables of names called nameOfPerson, phoneNumber, age, and category. The type of each variable is UM3CharacterString, UM3UnsignedInteger32, and UM3Enumerated, which are all UM3 primitive types. The variable category is UM3Enumerated type, where a friendly value is mapped to integer 0, a hostile value is mapped to integer 1, and a none value is mapped to integer 2.

Let's assume that a variable of MyUM3SetType type defined above is defined as myUM3SetValue, and the values of each variable are assigned as follows.

```
myUM3SetValue MyUM3SetType ::= {
        nameOfPerson            "John",
        phoneNumber             "1234",
        Age                     20,
        category                none
}
```

즉, myUM3SetValue 라는 변수는 그 값으로, 사람의 이름을 나타내는 nameOfPerson 변수와, 전화번호를

나타내는 phoneNumber 변수, 나이를 나타내는 age 변수, 나름대로의 기준으로 사람을 분류한 category 변수에, “John”, “1234”, 20, none 의 값을 지정하고, 이들을 하나의 집합으로 묶은 형태를 갖고 있다는 뜻입니다.

This means that the myUM3SetValue variable has the set of values of “John”, “1234”, 20, and none for the nameOfPerson variable with the name of person, phoneNumber variable with the telephone number, age variable with the age, and the category variable with a certain category.

이상과 같은 myUM3SetValue 변수를 인코딩하기 위해 인코딩 대상 변수들을 다이어그램으로 구성하면 그림 32-myUM3SetValue 변수의 인코딩 과정와 같습니다. 그림 32-myUM3SetValue 변수의 인코딩 과정에서 위의 정의와는 다르게 phoneNumber 변수와 nameOfPerson 변수의 순서가 바뀌어 있는 것은 UM3 SET 타입은 엘리먼트의 인코딩 순서에는 관계 없이 엘리먼트를 나열 할 수 있기 때문입니다.

To encode this myUM3SetValue value, the encoding target variables are configured as in diagram in Fig. 32. Unlike the above definition, the order of the phoneNumber variable and nameOfPerson is reversed in Fig. 32. This is because the elements can be listed independently from the encoding sequence of elements in the case of UM3 SET type.

또한 새롭게 정의한 MyUM3SetType 은 새로운 클래스 아이디를 지정해야 합니다. 이는 SET 타입의 클래스 아이디를 그대로 사용할 경우 송수신 할 때 사용되는 APDU 에서 해당 타입을 구분할 수 없기 때문입니다. 이 예에서는 MyUM3SetType 의 클래스 아이디를 20000_{10} 으로 정의합니다.

A new class ID should be assigned for the newly defined MyUM3SetType. This is because SET type cannot be identified in APDU used for data exchange if the class ID of SET type is used. In this example, the class ID of the MyUM3SetType is set to 20000_{10}.

기본적으로 UM3 SET 타입은 본 권고안의 다음절에서 정의할 UM3 정보모델과 유사한 인코딩 방법을 사용합니다. 이는 UM3 프로토콜의 정보모델을 구성하는 클래스는 UM3 SET, UM3 SEQUENCE, UM3 SET OF, UM3 SEQUENCE OF 타입으로 표현할 수 있기 때문입니다.

Basically, the encoding method similar to the UM3 information model to be defined in the next section of this recommendation is used for the UM3 SET typ. This is because the class that configures the information model of THE UM3 protocol can be defined with UM3 SET, UM3 SEQUENCE, UM3 SET OF, and UM3 SEQUENCE OF type.

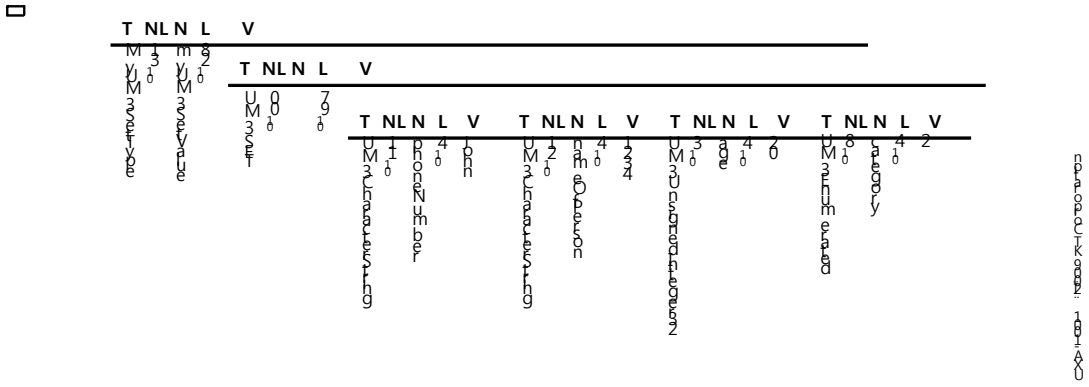


그림 32-myUM3SetValue 변수의 인코딩 과정 [Encoding process of myUM3SetValue variable]

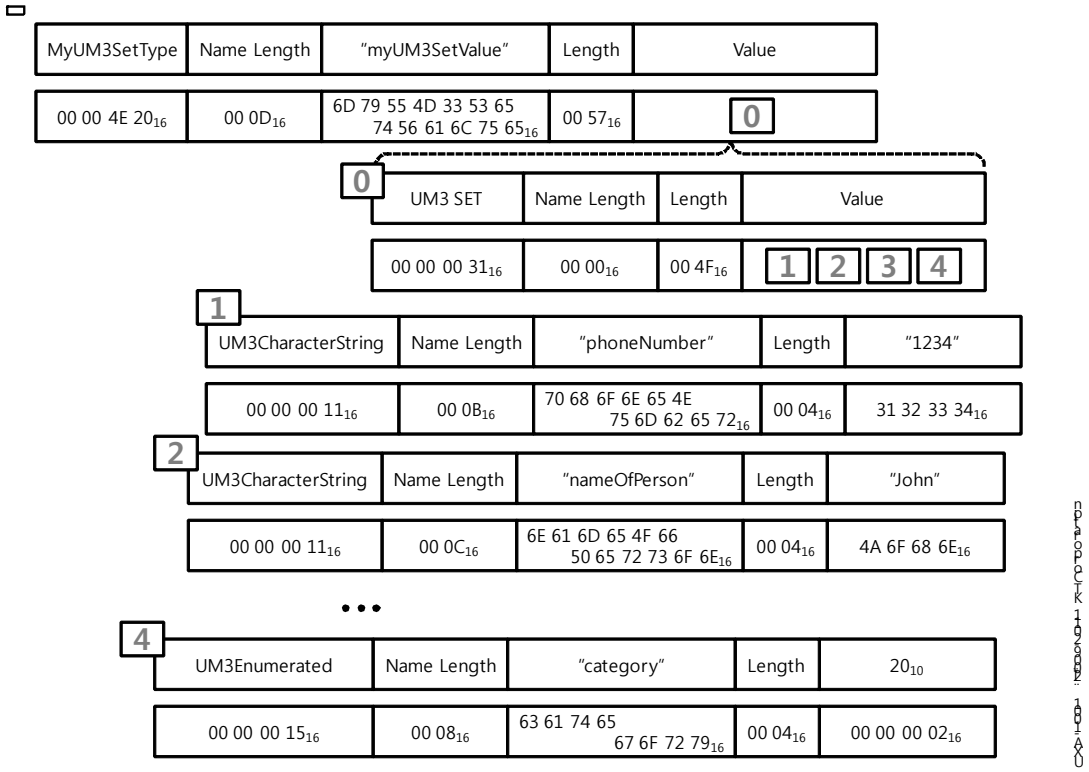


그림 33-myUM3SetValue 의 UM3 SER 인코딩 결과 [UM3 SER encoding result of myUM3SetValue]

그림 33-myUM3SetValue 의 UM3 SER 인코딩 결과의 경우 변수의 이름이 ‘myUM3SetValue’ 로 주어져 있으나, 경우에 따라서는 변수의 이름이 지정되지 않은 채로 데이터의 송수신이 이루어질 수 있습니다. 이런 경우에는 NL 필드의 값은 $00\ 00_{16}$ 으로 표기하고 N 필드는 생략할 수 있습니다.

In Fig. 33, the name of the variable is set to ‘myUM3SetValue’, but the data exchange can be performed without specifying the name of this variable when required. In this case, the value of the NL field should be set to $00\ 00_{16}$ and the N field can be skipped.

그림 33-myUM3SetValue 의 UM3 SER 인코딩 결과 의 예에서는 혼란을 방지하기 위하여 myUM3SetValue 를 변수라는 용어로 표현하였습니다. 본 권고안이 사용하는 변수라는 용어는 오브젝트와 동일한 용어입니다. 또한 UM3 SET 타입과 같은 복합형식 타입의 내부 변수로 정의된 변수들은 애트리뷰트와 동일한 개념으로 간주될 수도 있습니다.

In Fig. 33, myUM3SetValue was expressed with the term of 'variable' to avoid confusion. The meaning of variable used in this recommendation is the same as object. The variables defined as internal variables of complex format type like UM3 SET type are considered to be the same concept as attribute.

15.21. Encoding of UM3 SET OF type

UM3 SET OF 타입의 인코딩은 상기 UM3 SET 타입의 인코딩과 동일한 방식으로 이루어지며, APDU 헤더의 T 필드에 UM3 SET OF 타입의 클래스 아이디가 기록되게 됩니다. UM3 SET OF 타입은 UM3 복합형식 타입입니다.

Encoding of UM3 SET OF type is performed in the same way as the UM3 SET type, and the class ID of UM3 SET OF type will be stored in the T field of the UM3 APDU header. UM3 SET OF type is UM3 complex format type.

15.22. Encoding of UM3 SEQUENCE type

UM3 SEQUENCE 타입의 인코딩은 상기 UM3 SET 타입의 인코딩과 동일한 방식으로 이루어지며, APDU 헤더의 T 필드에 UM3 SEQUENCE 타입의 클래스아이디가 기록되게 됩니다. UM3 SEQUENCE 타입은 UM3 복합형식의 타입입니다.

Encoding of UM3 SEQUENCE type is performed in the same way as the UM3 SET type, and the class ID of UM3 SEQUENCE type will be stored in the T field of the UM3 APDU header. UM3 SEQUENCE type is UM3 complex format type.

15.23. Encoding of UM3 SEQUENCE OF type

UM3 SEQUENCE OF 타입의 인코딩은 상기 UM3 SET 타입의 인코딩과 동일한 방식으로 이루어지며, APDU 헤더의 T 필드에 UM3 SEQUENCE OF 타입의 클래스아이디가 기록되게 됩니다. UM3 SEQUENCE OF 타입은 UM3 복합형식의 타입입니다.

Encoding of UM3 SEQUENCE OF type is performed in the same way as the UM3 SET type, and the class ID of UM3 SEQUENCE OF type will be stored in the T field of the UM3 APDU header. UM3 SEQUENCE OF type is UM3 complex format type.

15.24. Encoding of UM3 information model

UM3 서비스관리 정보모델과 응용서비스 정보모델을 구성하고 있는 각 클래스들에 대한 인코딩은 위의 UM3 기본 타입들의 조합으로 이루어지게 됩니다. 단, 클래스의 정의에 있어서 필수 애트리뷰트 (Mandatory) 들은 반드시 APDU에 포함되어야 하며, 선택 애트리뷰트 (Optional) 들은 그 포함여부를 자유롭게 결정할 수 있습니다.

예를 들어 정보모델에 정의된 PseudoClass 라는 이름의 클래스의 정의가 아래와 같다고 가정합니다.

Encoding of each class configuring the UM3 service management information model and application service information model is performed based on the combination of UM3 primitive types. The mandatory attributes, however, should be included in APDU when class is defined, and the inclusion of optional attributes may be determined freely.

For example, let's assume that a class with the name of PseudoClass in the information model is defined as follows.

```
PseudoClass UM3 OBJECT CLASS
        CHARACTERIZED BY
            Value of the presentValue attribute;
        ATTRIBUTE NAME AS
            um3ClassIdentifier                  um3ClassIdentifier
            um3ObjectName                       um3ObjectName
            um3ObjectNameAlias                  um3ObjectNameAlias
            presentValue                        presentValue;
        ACTION DEFINED AS
            ANY UM3 OPERATION IN UM3 COMMUNICATION SERVICE MODEL
                EXCEPT
                    NONE;
        BEHAVIOR
            Example class to explain UM3 SER encoding;
;;
```

위의 PseudoClass 클래스를 ASN.1 정규표현식을 이용하여 나타내면 다음과 같습니다.

PseudoClass class can be defined by using ASN.1 regular expressions as follows.

```
PseudoClass ::= UM3 CLASS {
        &um3ClassIdentifier         UM3ClassIdentifier,
        &um3ObjectName              UM3ObjectName,
        &um3ObjectNameAlias         UM3CharacterString OPTIONAL,
        &presentValue               UM3Real
}
```

위의 PseudoClass 클래스의 클래스 아이디는 다음과 같이 정의되어 있는 것으로 가정합니다. 즉, um3ClassIdentifier 애트리뷰트의 값은 아래와 같이 정의되어 있는 것으로 가정합니다.

Let's assume that the class ID of the PseudoClass class is defined as follows. In other words, let's assume that the value of um3ClassIdentifier attribute is defined as below.

um3ClassIdentifier UM3ClassIdentifier ::= 256

PseudoClass 클래스가 인스턴스화 되었을 때 특정 오브젝트의 이름을 다음과 같이 가정합니다.

Name of a specific object when PseudoClass class is instantiated is assumed as follows.

um3ObjectName UM3ObjectName ::= "myPseudo"

상기 myPseudo 오브젝트의 presentValue 애트리뷰트의 값은 다음과 같이 가정합니다.

The value of the presentValue attribute of the myPseudo object is assumed as follows.

presentValue UM3Real ::= 0.16

이상과 같이 정의되어 있는 PseudoClass 의 오브젝트 myPseudo 를 UM3 SER 을 이용하여 인코딩하기 위해서는 T 필드에 um3ClassIdentifier의 값을 기록하고, N 필드에 um3ObjectName 값을 기록하며, NL 필

드에 um3ObjectName 애트리뷰트 값의 길이를 기록하며, V 필드에 UM3Real 타입을 갖는 presentValue 애트리뷰트의 인코딩 결과를 기록합니다.

To encode the object myPseudo of the PseudoClass defined above by using UM3 SER, the value of um3ClassIdentifier is stored in the T field, the value of um3ObjectName is stored in the N field, the length of the um3ObjectName attribute value is stored in the NL field, and the encoding result of the presentValue attribute of UM3Real type is stored in the V field.

그림 34-PseudoClass 클래스 오브젝트의 인코딩 예에서 보는 바와 같이 myPseudo 오브젝트의 UM3 SER 인코딩 결과로부터 해당 클래스의 애트리뷰트 값을 추출하기 위해서는 presentValue로 기록된 N 필드를 먼저 확인해야 합니다. 즉, UM3 SER 인코딩은 오브젝트의 애트리뷰트들의 나열 순서에 관계없이 인코딩을 진행합니다. 모든 애트리뷰트는 순서에 관계없이 오브젝트 이름을 기록한 N 필드를 기준으로 추출되어야 합니다.

As shown in Fig. 34, the value of the N field that contains presentValue should be checked first to extract the attribute value of the corresponding class from the UM3 SER encoding result of the myPseudo object. UM3 SER encoding should be performed independently from the listing sequence of attributes of objects. The extraction of all attributes should be done based on the N field that contains the object name independently from the sequence.

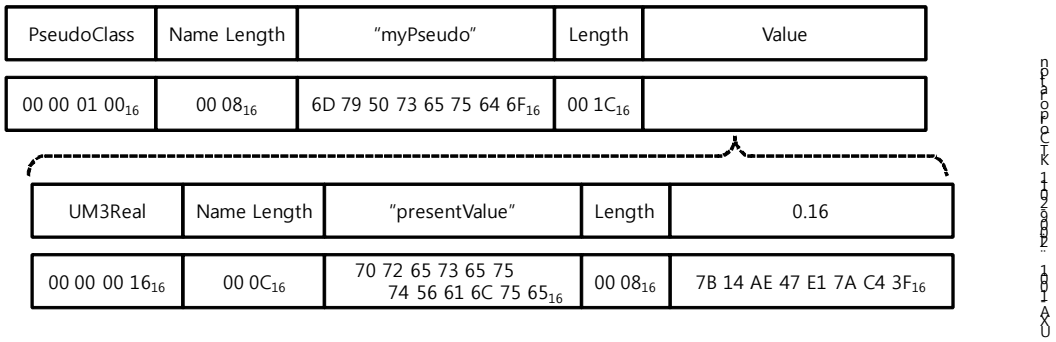


그림 34-PseudoClass 클래스 오브젝트의 인코딩 예 [Example of PseudoClass class object encoding]

PseudoClass 클래스의 um3ObjectNameAlias 애트리뷰트는 OPTIONAL로 지정되어 있으므로 그림 34-PseudoClass 클래스 오브젝트의 인코딩 예에서 보는 바와 같이 처음부터 정의되어 있지 않은채로 생략할 수도 있습니다.

The um3ObjectNameAlias attribute of PseudoClass class is set as OPTIONAL, and it can be skipped without definition from the first place as shown in 그림 34-PseudoClass 클래스 오브젝트의 인코딩 예.

15.25. Encoding of UM3 information model attribute object

서비스관리 정보모델 및 응용서비스 정보모델을 구성하는 모든 클래스들의 애트리뷰트는 앞의 예에서와 같이 클래스의 오브젝트를 인코딩 하는 것과 동일한 방식으로 인코딩 합니다.

Encoding of attributes of all classes of service management information model and application service information model is performed in the same way as the encoding of the object of a class as in the previous example.

특정 오브젝트의 애트리뷰트가 UM3 기본 타입일 경우 해당 기본 타입의 인코딩 방식을 그대로 적용합니다.

If the attribute of a specific object is UM3 primitive type, then the encoding method of the corresponding primitive type should be applied.

만약 특정 오브젝트의 애트리뷰트가 또 다른 클래스 타입으로 정의된 애트리뷰트라면, 해당 애트리뷰트는 상기 UM3 정보모델 오브젝트의 인코딩 방식과 동일한 방식으로 인코딩 룰을 적용합니다.

If the attribute of a specific object is an attribute defined as a class type, then encoding of the corresponding attribute should be performed in the same way as the encoding method of the UM3 information model object.

예를 들어 다음의 예에서와 같이 Another 클래스가 정의되어 있는 것으로 가정합니다.

For example, let’s assume that Another class is defined as follows.

```
Another UM3 OBJECT CLASS
        CHARACTERIZED BY
            Value of contactInfo attribute;
        ATTRIBUTE NAME AS
            um3ClassIdentifier                  um3ClassIdentifier
            um3ObjectName                       um3ObjectName
            um3ObjectNameAlias                  um3ObjectNameAlias
            contactInfo                         contactInfo;
        ACTION DEFINED AS
            ANY UM3 ACTION IN UM3 COMMUNICATION SERVICE MODEL
                EXCEPT
                    NONE;
        BEHAVIOR
            Example class to explain UM3 SER encoding;
;;
```

Another 클래스의 ASN.1 정규표현식으로의 표현은 아래와 같습니다.

Another class can be defined by using ASN.1 regular expressions as follows.

```
Another ::= CLASS {
        &um3ClassIdentifier             UM3ClassIdentifier,
        &um3ObjectName                  UM3ObjectName,
        &um3ObjectNameAlias             UM3CharacterString OPTIONAL,
        &contactInfo                    ContactInfo
}
```

위의 Another 클래스의 오브젝트는 다음과 같은 애트리뷰트 값들을 갖고 있는 것으로 가정합니다.

Let's assume that the object of Another class has the following attribute values.

um3ObjectName UM3ObjectName ::= "another"

다음으로 contactInfo 애트리뷰트의 타입인 ContactInfo 클래스가 다음과 같이 정의되어 있다고 가정합니다.

Let's also assume that the ContactInfo class of the contactInfo attribute type is defined as follows.

```
ContactInfo UM3 OBJECT CLASS
    CHARACTERIZED BY
        Value of cellphone attribute;
    ATTRIBUTE NAME AS
        um3ClassIdentifier              um3ClassIdentifier
        um3ObjectName                   um3ObjectName
        um3ObjectNameAlias              um3ObjectNameAlias
        cellphone                       cellphone
        zipcode                         zipcode;
    ACTION DEFINED AS
        ANY UM3 ACTION IN UM3 COMMUNICATION SERVICE MODEL
            EXCEPT
                NONE;
    BEHAVIOR
        Example class to explain UM3 SER encoding;
```

;;

또한 위의 ContactInfo 클래스의 ASN.1 정규표현식은 다음과 같습니다.

ContactInfo class can be defined by using ASN.1 regular expressions as follows.

```
ContactInfo ::= UM3 CLASS {
        &um3ClassIdentifier          UM3ClassIdentifier,
        &um3ObjectName               UM3ObjectName,
        &um3ObjectNameAlias          UM3CharacterString OPTIONAL,
        &cellphone                   UM3CharacterString,
        &zipcode                     UM3CharacterString
}
```

이 때 ContactInfo 클래스의 클래스 아이디는 다음과 같이 정의된 것으로 가정합니다.

In this case, it is assumed that the class ID of the ContactInfo class is defined as follows.

um3ClassIdentifier UM3ClassIdentifier ::= 256

오브젝트의 이름은 다음과 같이 정의되어 있습니다.

The name of the object is defined as follows.

um3ObjectName UM3ObjectName ::= “contactInfo”

즉, 해당 오브젝트가 다른 상위 오브젝트의 애트리뷰트로 정의되어 있을 경우, 그 오브젝트 애트리뷰트의 이름을 저장하는 um3ObjectName 애트리뷰트는 상위 오브젝트에서 정의한 애트리뷰트의 이름을 해당 오브젝트의 이름으로 사용하게 됩니다.

이 때 해당 오브젝트의 cellphone 애트리뷰트의 값은 다음과 같이 지정되어 있는 것으로 가정합니다.

If the corresponding object is defined as an attribute of upper level object, then the um3ObjectName attribute that stores the name of its object attribute uses the name of the attribute defined in the upper level object as the name of

the corresponding object.

In this case, it is assumed that the value of cellphone attribute of the corresponding object is set as follows.

cellphone UM3CharacterString ::= “000-0000-0000”

또한 zipcode 애트리뷰트의 값은 다음과 같이 정의되어 있는 것으로 가정합니다.

It is also assumed that the value of the zipcode attribute is defined as follows.

zipcode UM3CharacterString ::= “112-111”

이상과 같은 another.contactInfo 라는 이름의 RDN을 갖는 애트리뷰트의 UM3 SER 인코딩의 결과는 그림 35-오브젝트 애트리뷰트의 인코딩 예 [와 같습니다. 앞의 경우와 마찬가지로 cellphone 과 zipcode 등 두 개의 애트리뷰트에 대한 UM3 SER 인코딩과 UM3 SER 디코딩은 순서에 관계 없이 각 APDU의 오브젝트 이름이 기록되는 N 필드의 값을 기준으로 진행해야 합니다.

Fig. 35 is the UM3 SER encoding result of the attribute that has RDN with the name of another.contactInfo. Like the previous example, UM3 SER encoding and UM3 SER decoding for two attributes of cellphone and zipcode should be performed based on the value of the N field that contains the object name of APDU independently from the sequence.

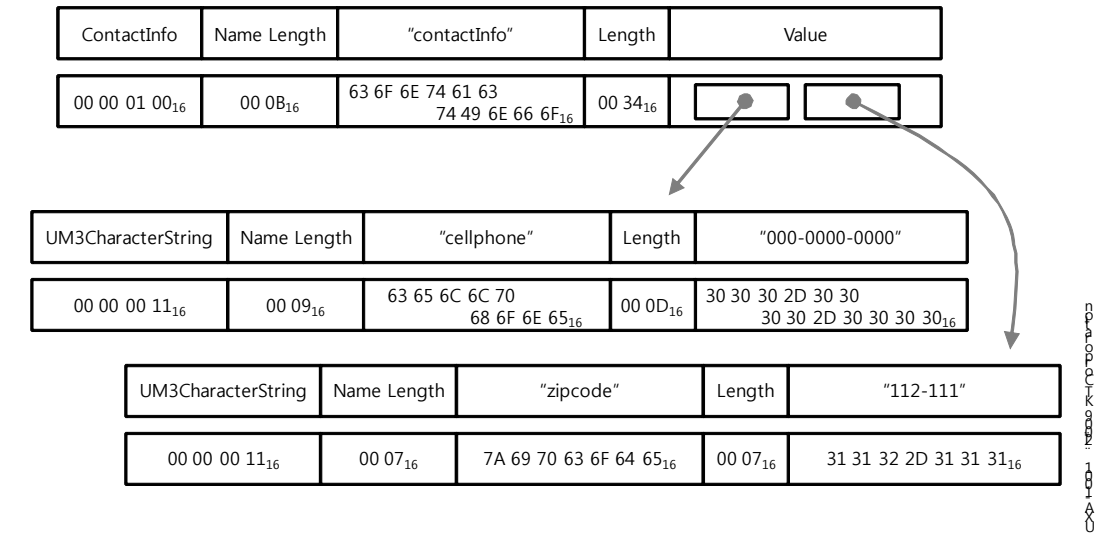


그림 35-오브젝트 애트리뷰트의 인코딩 예 [-Example encoding of object attribute]

15.26. Encoding of UM3 communication service model object

본 권고안은 통신서비스모델의 정의에서 통신서비스모델을 구성하는 오퍼레이션들을 클래스로 정의하였습니다. 해당 클래스가 인스턴스화 되었을 때 이는 오퍼레이션 오브젝트가 됩니다. 오퍼레이션 오브젝트를 구현하는 방법은 프로그래밍 언어에 따라 여러 가지 형태가 가능하지만 C, C++, C#, Java의 경우 method, function, member operation 등의 형태로 구현되게 됩니다.

In the definition of the communication service model in this recommendation, class is defined as operations that configure the communication service model. When the corresponding class is instantiated, it becomes an operation object. There are various ways of implementing operation objects, and it is implemented as a method, function, or member operation in the case of C, C++, C#, or Java.

상기 오퍼레이션 오브젝트를 UM3 SER을 이용하여 인코딩 하는 경우, 그 패킷의 구성은 그림 25-UM3 프로토콜의 APDU 구성 [와 동일합니다. 즉, T 필드에는 오퍼레이션 클래스의 클래스 아이디를 기록하고, N 필드에는 해당 오퍼레이션 오브젝트의 오브젝트 이름을 기록합니다. NL 필드에는 오퍼레이션 오브젝트 이름의 길이를 기록하며, V 필드에는 해당 오퍼레이션의 파라메터 오브젝트의 인코딩 결과를

기록하며, L 필드에는 해당 오브젝트의 V 필드의 길이를 기록합니다.

When the operation object is encoded by using UM3 SER, the configuration of the packet is the same as Fig. 25. The class ID is stored in the T field and object name of the corresponding operation object is stored in the N field. The NL field contains the length of the name of operation object, the V field contains the encoding result of the parameter object of the corresponding operation, and the L field contains the length of the V field of the corresponding object.

위의 오퍼레이션 오브젝트의 이름은 TCP/IP 응용계층의 세션 종료 후 해당 오퍼레이션에 대한 리턴 값을 받을 때 유용하게 활용할 수 있습니다. 즉, 오퍼레이션의 이름은 매니저 혹은 에이전트들 중 오퍼레이션을 요청하는 쪽이 N 필드에 UM3 세션의 이름을 기록하여 전송합니다.

The name of this operation object can be useful when receiving the return value for the corresponding operation after termination of the session in the TCP/IP application layer. In other words, the manager or agent that requests the operation should fill the N field with the name of the UM3 session and send it to the other end.

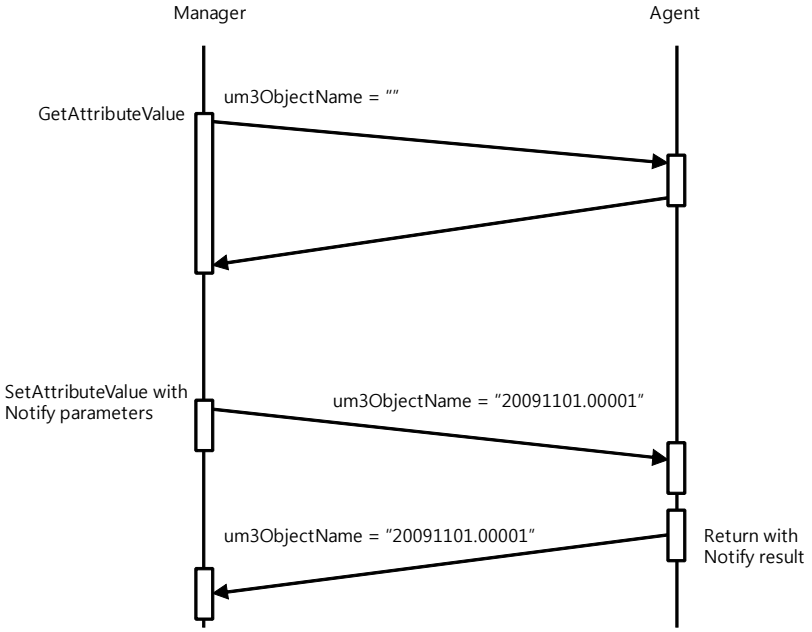


그림 36-오퍼레이션 클래스의 um3ObjectName 애트리뷰트의 활용
[Utilization of um3ObjectName attribute of operation class]

해당 오퍼레이션에 대한 응답을 OperationResponse 오퍼레이션을 통해 송신할 때 해당 응답 APDU 의 N 필드에는 수신한 UM3 세션 아이디 값을 그대로 복사하여 기록한 후 나머지 파라메터들을 인코딩 하여 송신합니다.

When the response is sent to the corresponding operation through the OperationResponse operation, the received UM3 session ID should be copied to the N field of the corresponding response APDU without modification and other parameters should be encoded for transmission.

그림 36-오퍼레이션 클래스의 um3ObjectName 애트리뷰트의 활용 은 오퍼레이션 오브젝트의 um3ObjectName 애트리뷰트를 UM3 세션 아이디로 활용하는 예를 나타내고 있습니다. UM3 세션 아이디의 구성 형식에는 아무런 제약 조건이 없으며 상황에 따라 알맞게 생성하되 특정 매니저가 생성하는 모든 오퍼레이션의 세션 아이디는 모든 송수신 APDU 에 있어서 유일한 세션 아이디로 정의되어야 합니다.

그림 36-오퍼레이션 클래스의 um3ObjectName 애트리뷰트의 활용 is an example of using the um3ObjectName attribute of the operation object as the UM3 session ID. There is a restriction in the configuration format of the UM3 session ID. It should be created properly for the given situation and the session IDs of all operations created by specific manager should have a unique number for all transmission and reception APDU.

UM3 오퍼레이션에 대한 인코딩과 디코딩 과정은 앞서 정의하고 설명한 정보모델을 구성하는 오브젝트들에 대한 인코딩 혹은 디코딩 과정과는 달리 NL 필드의 값은 항상 N 필드를 구성하는 파라메터의 이름의 길이를 반드시 표시해야 합니다. 또한 N 필드에 저장되는 값은 본 권고안에 정의된 각 오퍼레이션의 파라메터 이름을 정확하게 기입해야 합니다. 즉, 정보모델을 구성하는 클래스들의 인코딩과는 달리 NL 필드의 값을 00_{16} 으로 정의할 수 없습니다. 또한 앞서 설명한 바와 같이 N 필드의 값은 반드시 본 권고안에 정의된 파라메터의 이름을 정확하게 기입하여야 합니다.

Unlike the encoding and decoding process of the objects configuring the information model that is defined and explained earlier, the name and length of the parameter that configures the N field should be specified in the NL field for the encoding and decoding of UM3 operation. The values stored in the N field should have the accurate name of the parameter of each operation defined in this recommendation. In other words, unlike the encoding of classes that configure the information model, the value of the NL field cannot be set to 00_{16}. The value of the N field should be accurately filled with the name of the parameter defined in this recommendation.

16. XML Type Data Exchange

본 권고안이 정의하는 XML 형식의 데이터 송수신 방식은 앞서 정의한 UM3 기본 타입, UM3 복합형식 타입, UM3 클래스 타입의 데이터를 송수신하는 방법에 있어서, 각 타입별 데이터를 UM3 SER 인코딩 룰을 사용하여 송수신하는 대신 XML 형식의 텍스트 기반 데이터로 송수신하는 방법으로 정의합니다. .

As a means of exchanging data of UM3 primitive type, UM3 complex format type, and UM3 class type defined earlier, the XML type data exchange defined in this recommendation is defined as the method of exchanging XML type text based data instead of exchanging data each time by using the UM3 SER encoding rule.

따라서, XML 기반의 데이터 송수신 방식은 UM3 SER 인코딩 룰을 사용하는 대신 UM3 primitive type, UM3 complex format type, UM3 class type 등 서비스관리 정보모델과 응용서비스 정보모델을 구성하는 모든 데이터 형식과 더불어 UM3 통신서비스 모델을 구성하는 오퍼레이션까지 모든 데이터를 XML 기반의 데이터로 주고 받음을 의미합니다.

This means that all types of data, including the types of data that configure the service management information model and application service information model such as UM3 primitive type, UM3 complex format type, and UM3 class type and all operations configuring the UM3 communication service model, can be exchanged with XML based data without using the UM3 SER encoding rule.

본 권고안이 앞서 정의한 UM3 primitive type, UM3 complex format type 및 UM3 class type 을 구성하는 모든 데이터 타입과 클래스 타입들을 구성하는 애트리뷰트, 애트리뷰트 혹은 엘리먼트의 나열 순서 등의 규칙은 XML 기반의 데이터 송수신을 위한 데이터 구성에도 동일하게 적용됩니다.

The rules of attributes configuring all data types and class types that configure UM3 primitive type, UM3 complex format type, and UM3 class type defined earlier in this recommendation and the listing sequence of attributes of elements should apply to the configuration of data for XML based data exchange in the same way.

16.1. UM3 XML encoding

위에서 설명한바와 같이 UM3 XML 인코딩 룰 또한 UM3 SER 인코딩 룰이 갖고 있는 특징을 대부분 그대로 갖고 있습니다.

As explained above, the UM3 XML encoding rule has most of the features of the UM3 SER encoding rule.

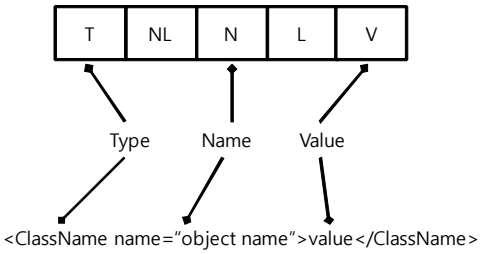


그림 37-UM3 SER 과 UM3 XML Encoding 의 비교 [Comparison of UM3 SER and UM3 XML Encoding]

그림 37-UM3 SER 과 UM3 XML Encoding 의 비교 에서 보는 바와 같이 UM3 XML 인코딩은 UM3 SER 인코딩 룰이 갖고 있는 T-NL-N-L-V 형식의 APDU 와 같은 T-N-V 형식을 갖고 있습니다. 즉, 클래스 타입과 해당 클래스 오브젝트의 오브젝트 이름, 해당 오브젝트의 값으로 표현되는 형식을 갖고 있습니다.

As shown in Fig. 37, UM3 XML encoding has T-N-V format like the APDU of T-NL-N-L-V format of the UM3 SER encoding rule. In other words, it has the format that is represented by class type, object name of the corresponding class object, and value of the corresponding object

16.2. XML Representation of Communication Service Model

UM3 통신서비스 모델을 구성하는 오퍼레이션들에 대한 XML 표현 방식을 정의합니다. 기본적으로 해당 오퍼레이션의 파라메터 구성은 앞서 통신서비스모델에서 정의한 내용과 동일하게 적용되어야 합니다. 또한 통신서비스모델을 구성하는 오퍼레이션들은 그 명칭 그대로 XML 데이터를 구성하는 엘리먼트들로 사용되게 됩니다. 즉, XML 기반의 데이터 송수신 방식을 사용한다 하더라도 앞서 정의한 UM3 통신서비스모델을 명확하게 이해하고 있어야 합니다.

This defines the XML representation method for the operations that configure UM3 communication service model. Basically, the parameter configuration of the corresponding operation should be applied in the same way as the contents defined earlier in the communication service model. Operations that configure the communication service model are used as elements of XML data with their names. This means that the UM3 communication service model defined earlier should be clearly understood, even if the XML based data exchange method is used.

16.2.1. OperationResponse operation

아래의 예는 특정 UM3 오퍼레이션에 대한 성공 signature 를 나타냅니다. 즉, UM3Boolean 타입의 parResult 파라메터에 TRUE 값을 기록하여 전송하는 경우입니다.

The following example shows the SUCCESS signature for the specific UM3 operation. The parResult parameter of UM3Boolean type is set to TRUE, and the result is returned.

```
<OperationResponse name=”op-session-id-100001”>
    <UM3Boolean name=”parResult”>TRUE</UM3Boolean>
</OperationResponse>
```

만약 특정 오퍼레이션의 결과가 실패일 경우에는 다음과 같은 형식으로 OperationResponse 오퍼레이션을 호출 합니다.

If the specific operation fails, then the OperationResponse operation is called as follows.

```
<OperationResponse name=”op-session-id-100001”>
    <UM3Boolean name=”parResult”>FALSE</UM3Boolean>
    <UM3OperationErrorCode name=”parUM3OperationErrorCode>
        noSuchClass
    </UM3OperationErrorCode>
</OperationResponse>
```

위의 예에서 볼 수 있듯이 앞서 UM3 통신서비스 모델에서 정의한 OperationResponse 오퍼레이션의 파라메터와 동일한 형태와 순서의 파라메터들이 UM3 SEQUENCE OF 타입으로 채워져 있어야 합니다.

OperationResponse 오퍼레이션을 구성하는 각각의 signature 들은 상기 열거된 예와 마찬가지의 형태로 OperationResponse 오퍼레이션을 구성하는 각 signature 별로 데이터를 나열하고 이를 상대측으로 전송해야 합니다. 특히, 본 권고안의 XML schema 에 정의되어 있는 것과 같이 오퍼레이션을 구성하는 파라메터는 <SEQUENCE> 타입으로 구성되어 있음에 유의하여야 합니다. 이는 파라메터의 나열 순서가 앞서 정의한 UM3 통신서비스모델에 정의된 것과 같은 순서대로 나열되어야 함을 의미합니다.

As shown in the above example, parameters of the same type and sequence as the parameters of the OperationResponse operation defined in the UM3 communication service model should be filled with UM3 SEQUENCE OF type.

Each signature configuring the OperationResponse operation should list the data by each signature that configures the

OperationResponse operation in the same form as the examples listed above and send it to the other end. Caution is especially required in that the parameter configuring the operation is <SEQUENCE> type like it is defined in the XML schema in this recommendation. This means that the listing sequence of the parameter should be same as the sequence defined in the UM3 communication service model.

16.2.2. GetObjectValue operation

GetObjectValue 오퍼레이션의 요청 signature 는 다음의 예에서와 같이 구성됩니다.

The REQUEST signature of the GetObjectValue operation is configured as the following example.

```
<GetObjectValue name=”some-session-identifier”>
    <UM3ClassIdentifier name=”parUM3ClassIdentifier”>1717</UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3ObjectName>myObject</UM3ObjectName>
</GetObjectValue>
```

즉, ‘some-session-identifier’ 라는 UM3 세션 아이디 값을 오퍼레이션 오브젝트의 이름으로 갖고 있는 GetObjectValue 오퍼레이션이 호출되며, 그 파라메터는 1717 이라는 클래스아이디와 ‘myObject’ 라는 오브젝트 이름으로 구성된다는 의미입니다.

The GetObjectValue operation that has the UM3 session ID of ‘some-session-identifier’ as the name of the operation object is called, and the parameters are class ID of 1717 and object name of ‘myObject’.

상기 오퍼레이션에 대한 응답은 앞서 정의한 바와 같이 OperationResponse 오퍼레이션으로 구성됩니다. 즉, 상기 오퍼레이션이 호출될 경우, 해당 작업을 수행한 에이전트는 다시 매니저 측으로 OperationResponse 오퍼레이션을 호출함으로써 상기 오퍼레이션에 대한 응답을 완료하게 됩니다.

The response to this operation is configured with the OperationResponse operation as defined earlier. In other words, when this operation is called, the response to this operation is completed by the agent that executes the corresponding task to call the OperationResponse operation for the response to the manager.

상기 오퍼레이션이 성공하였을 경우의 XML 응답 데이터 구성의 예는 다음과 같습니다.

When this operation is a success, an example configuration of the XML response data is as follows.

```
<OperationResponse name=”some-session-identifier”>
    <UM3Boolean name=”parResult”>TRUE</UM3Boolean>
    <UM3UnsignedInteger16 name=”parAnyTypeValue”>1717</UM3UnsignedInteger16>
</OperationResponse>
```

parAnyTypeValue 는 GetObjectValue 오퍼레이션이 필요로하는 데이터의 타입에 따라 달라질 수 있으며 해당 데이터의 타입은 UM3UnsignedInteger16 임을 나타냅니다.

. parAnyTypeValue depends on the type of data required by the GetObjectValue operation, and the type of the corresponding data is UM3UnsignedInteger16.

만약 상기 오퍼레이션의 요청이 실패하였을 경우에는 다음과 같은 구성으로 OperationResponse 오퍼레이션을 구성할 수 있습니다.

If this operation fails, then the OperationResponse operation may be configured as follows.

```
<OperationResponse name=”some-session-identifier”>
    <UM3Boolean name=”parResult”>FALSE</UM3Boolean>
    <UM3OperationErrorCode name=”parUM3OperationErrorCode”>
        noSuchClass
    </UM3OperationErrorCode>
</OperationResponse>
```

다른 예에서와 마찬가지로 위의 예에서도 parResult 와 parOperationErrorCode 엘리먼트의 순서에 의미가 있으며 반드시 본권고안의 통신서비스모델의 정의에 따라 그 순서를 정확하게 맞추어야 합니다.

Like in another example, the sequence of the elements of parResult and parOperationErrorCode is also important, and the sequence should be accurately matched based on the definition of the communication service model of this recommendation.

16.2.3. GetObjectAttributeValue operation

GetObjectAttributeValue 오퍼레이션 요청 signature 의 예는 다음과 같습니다.

An example of the REQUEST signature of the GetObjectAttributeValue operation is as follows.

```
<GetObjectAttributeValue name=”THIS-SESSION-AAA-01”>
    <UM3ClassIdentifier name=”parUM3ClassIdentifier”>55</UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3ObjectName”> myGateway.yourSensor</UM3ObjectName>
    <UM3ClassIdentifier name=”parUM3AttributeClassIdentifier>77</UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3AttributeObjectName”>presentValue</UM3ObjectName>
</GetObjectAttributeValue>
```

다른 오퍼레이션들과 마찬가지로 파라메터의 순서는 본 권고안이 정의하는 오퍼레이션의 순서에 따라 나열해야 합니다.

Like other operations, the parameters should be listed in the order of operations defined in this recommendation.

16.2.4. GetObjectGroupValue operation

GetObjectGroupValue 오퍼레이션의 요청 signature 의 예는 다음과 같습니다.

An example of the REQUEST signature of the GetObjectGroupValue operation is as follows.

```
<GetObjectGroupValue name=”some-session-identifier”>
    <UM3ObjectList name=”parUM3ObjectList”>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>
            1717
        </UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>
            firstObjectName
        </UM3ObjectName>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>
            1717
        </UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>
            secondObjectName
        </UM3ObjectName>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>
            1717
        </UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>
            thirdObjectName
        </UM3ObjectName>
    </UM3ObjectList>
</GetObjectGroupValue>
```

위의 예는 3 개의 오브젝트에 대한 모든 값을 가져오기 위해 매니저가 에이전트로 송신하는 GetObjectGroupValue 오퍼레이션의 signature 구성 예입니다.

The above example is the configuration of the signature of the GetObjectGroupValue operation that the manager sends to the agent to get all values of three objects.

다음 예는 그 값을 가져오고자 하는 오브젝트를 지정하지 않고 조건문만을 파라메터로 넘겨 해당 조건을 만족하는 모든 오브젝트를 가져오는 경우의 요청 signature 입니다.

The next example is a REQUEST signature to get all objects that meet the conditions by only passing the conditional statements without specifying the objects to get the values.

```
<GetObjectGroupValue name=”some-session-identifier”>
    <UM3ObjectValueCondition name=”parUM3ObjectValueCondition”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”>111</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”>myObject</UM3ObjectName>
        <UM3CharacterString name=”um3ObjectNameAlias”>my001</UM3CharacterString>
        <UM3CharacterString name=”um3ObjectDescription”>Description goes here</UM3CharacterString>
        <UM3ObjectList name=”um3AttributeList”>
            <UM3ClassIdentifier>1001</UM3ClassIdentifier>
            <UM3ObjectName>um3ClassIdentifier</Um3ObjectName>
            <UM3ClassIdentifier>1002</UM3ClassIdentifier>
            <UM3ObjectName>um3ObjectName</UM3ObjectName>
            <UM3ClassIdentifier>1003</UM3ClassIdentifier>
            <UM3ObjectName>um3ObjectDesciption</UM3ObjectName>
            <UM3ClassIdentifier>1004</UM3ClassIdentifier>
            <UM3ObjectName>um3AttributeList</UM3ObjectName>
            <UM3ClassIdentifier>1005</UM3ClassIdentifier>
            <UM3ObjectName>upperEnd</UM3ObjectName>
            <UM3ClassIdentifier>1006</UM3ClassIdentifier>
            <UM3ObjectName>upperEnd</UM3ObjectName>
            <UM3ClassIdentifier>1007</UM3ClassIdentifier>
            <UM3ObjectName>lowerEnd</UM3ObjectName>
            <UM3ClassIdentifier>1008</UM3ClassIdentifier>
            <UM3ObjectName>condition</UM3ObjectName>
            <UM3ClassIdentifier>1009</UM3ClassIdentifier>
            <UM3ObjectName>targetAttributeClassIdentifier</UM3ObjectName>
            <UM3ClassIdentifier>1111</UM3ClassIdentifier>
            <UM3ObjectName>targetAttributeObjectName</UM3ObjectName>
        </UM3ObjectList>
        <UM3Real name=”upperEnd”>66.6</UM3Real>
        <UM3Real name=”lowerEnd”>-77.7</UM3Real>
        <UM3Enumerated name=”condition”>LELoGEU</UM3Enumerated>
        <UM3ClassIdentifier name=”targetAttributeClassIdentifier”>99</UM3ClassIdentifier>
        <UM3ObjectName name=”targetAttributeObjectName”>presentValue</UM3ObjectName>
    </UM3ObjectValueCondition>
</GetObjectGroupValue>
```

상기 예에서 UM3ObjectList 타입의 애트리뷰트는 생략 가능한 애트리뷰트입니다. 조건문이 나타내는 뜻은 비교대상 애트리뷰트의 값을 *x* 라 할 때 *-77.7 <= x <= 66.6* 의 조건문입니다.

The attribute of UM3ObjectList type in this example can be skipped. The meaning of the conditional statement is *-77.7 <= x <= 66.6*, where x is the value of the target comparison attribute.

만약 위의 오퍼레이션 요청에 따른 수행 결과가 실패로 나타날 경우 에이전트가 매니저로 전송하는

OperationResponse 오퍼레이션의 실패 signature 들 중 하나의 예를 다음에 나타냅니다.

The following example is one of the FAIL signatures of the OperationResponse operation that the agent sends to the manager when the result of this operation is FAIL.

```
<OperationResponse name=”some-session-identifier”>
    <UM3Boolean name=”parResult”>FALSE</UM3Boolean>
    <UM3OperationErrorCode name=”parUM3OperationErrorCode”>
        accessDenied
    </UM3OperationErrorCode>
    <UM3ClassIdentifier name=”parUM3ObjectClassIdentifier”>
        1717
    </UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3ObjectObjectName”>
        thatGateway.thisObject
    </UM3ObjectName>
    <UM3ClassIdentifier name=”parUM3AttributeClassIdentifier”>
        8888
    </UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3AttributeObjectName”>
        veryAttribute
    </UM3ObjectName>
</OperationResponse>
```

위의 예는 요청 signature 가 조건문만으로 이루어져 전송되고 이를 처리하기 위한 오퍼레이션의 수행 도중 에러가 발생한 경우 이를 리턴하는 경우의 XML 데이터의 구성을 보여줍니다. 즉, 상기 OperationResponse 오퍼레이션은 1717 클래스아이디와 thatGateway.thisObject 라는 오브젝트 이름을 갖는 오브젝트의 애트리뷰트들 중 8888 클래스아이디와 veryAttribute 애트리뷰트 이름을 갖는 애트리뷰트에서 에러가 발생했음을 나타내는 OperationResponse 오퍼레이션의 파라메터 구성을 보여주고 있습니다. 이 때 accessDenied 라는 값을 갖고 있는 parUM3OperationErrorCode 파라메터가 나타내는 뜻은 해당 애트리뷰트의 접근이 거부되었다는 뜻을 나타냅니다.

In this example, the REQUEST signature has only conditional statements, and the configuration of XML data returns the error that occurs during the execution of operation to process the conditions.

The above example is the configuration of parameters of the OperationResponse operation that indicates the occurrence of error in the attribute with the class ID of 8888 and attribute name of veryAttribute among the attributes of the object with the class ID of 1717 and object name of thatGateway.thisObject. In this case, the meaning of the parUM3OperationErrorCode parameter that has the value of accessDenied is that the access to the corresponding attribute is denied.

16.2.5. GetObjectAttributeGroupValue operation

GetObjectAttributeGroupValue 오퍼레이션은 앞서 정의한 바와 같이 특정 오브젝트를 구성하는 애트리뷰트들 중 복수의 애트리뷰트 값들을 가져올 때 사용하는 오퍼레이션입니다. GetObjectAttributeGroupValue 오퍼레이션의 signature 는 다음과 같습니다.

The GetObjectAttributeGroupValue operation is used to get one or more attribute values among the attributes of the specific object. Signature of the GetObjectAttributeGroupValue operation is as follows.

```
<GetObjectAttributeGroupValue name=”some-session-identifier”>
    <UM3ClassIdentifier name=”parUM3ClassIdentifier”>777</UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3ObjectName”>someObjectName</UM3ObjectName>
    <UM3ObjectList name=”parUM3AttributeObjectList”>
    <UM3ObjectList name=”parUM3ObjectList”>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>
            1717
        </UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>
            firstObjectName
        </UM3ObjectName>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>
            1717
        </UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>
            secondObjectName
        </UM3ObjectName>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>
            1717
        </UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>
            thirdObjectName
        </UM3ObjectName>
    </UM3ObjectList>
</GetObjectAttributeGroupValue>
```

위의 예에서 보는 바와 같이 UM3ObjectList 는 UM3ObjectIndicator 타입의 엘리먼트들을 파라메터로 넘겨주어야 합니다.

UM3ObjectList should pass the elements of UM3ObjectIndicator type as parameters as shown in the above example.

16.2.6. SetObjectValue operation

SetObjectValue 오퍼레이션 signature 의 예는 다음과 같습니다.

An example of the SetObjectValue operation signature is as follows.

```
<SetObjectValue name=”some-operation-session-id”>
    <UM3ClassIdentifier name=”parUM3ClassIdentifier”>88888</UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3ObjectName”>objectName</UM3ObjectName>
    <UM3Real name=”parAnyTypeValue”>17.1715</UM3Real>
</SetObjectValue>
```

위의 예에서 parAnyTypeValue 파라메터는 UM3Real 타입이며 그 클래스아이디는 8888 로 가정한 경우입니다. 또한, 해당 파라메터의 값은 17.1715 로 정의되어 있는 경우입니다.

위의 예에서와 같이 parAnyTypeValue 파라메터는 본 권고안이 정의한 모든 타입의 엘리먼트를 기록할 수 있으며 수신측은 해당 타입을 판별하고 구분할 수 있어야 합니다. 즉, 위의 예는 UM3 기본 타입의 데이터를 송신하기 위한 예이며 경우에 따라 특정 오브젝트의 모든 애트리뷰트 값들을 모두 포함할 수도 있으며 수신측에서는 이를 모두 해석하고 구분하여 처리할 수 있어야 함을 뜻합니다.

In this example, the parAnyTypeValue parameter is UM3Real type and the class ID is 8888. The value of the corresponding parameter is set to 17.1715.

As shown in this example, the parAnyTypeValue parameter can have any type of element defined in this recommendation, and the receiver should identify the corresponding type. In this example, data of UM3 primitive type is transmitted and all attribute values of the specific object can be included, and the receiver should analyze, identify, and process all of them.

16.2.7. SetObjectAttributeValue operation

SetObjectAttributeValue 오퍼레이션은 앞서 정의한 바와 같이 특정 오브젝트를 구성하고 있는 애트리뷰트들 중 특정 애트리뷰트의 값을 설정하기 위해 사용됩니다.

The SetObjectAttributeValue operation is used to set the value of a specific attribute among the attributes of a specific object as defined earlier.

```
<SetObjectAttributeValue name=”some-operation-session-id”>
    <UM3ClassIdentifier name=”parUM3ClassIdentifier”>777</UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3ObjectName”>objectName</UM3ObjectName>
    <UM3ClassIdentifier name=”parUM3AttributeClassIdentifier”>888</UM3ClassIdentifier>
    <UM3ObjectName name=”parUM3AttributeObjectName”>attributeName</UM3ObjectName>
    <UM3Real name=”parAnyTypeValue”>17.1715</UM3Real>
```

```
</SetObjectAttributeValue>
```

위의 예는 777 을 클래스 아이디 값으로 갖고 있으며 오브젝트의 이름이 objectName 인 특정 오브젝트의 애트리뷰트들 중, 그 클래스아이디 값이 888 이며 애트리뷰트의 이름이 attributeName 인 애트리뷰트의 값을 설정하기 위한 예를 보여주고 있습니다. 위의 예제에서 888 이라는 가상의 클래스 아이디는 UM3Real 타입을 나타내며, parAnyTypeValue 파라메터는 UM3Real 타입의 17.1715 라는 값을 나타내게 됩니다.

This example is to set the value of attribute with the class ID of 888 and attribute name of attributeName among the attributes of the object with the class ID of 777 and object name of objectName. The virtual class ID of 888 is UM3Real type, and the parAnyTypeValue parameter of 17.1715 is UM3Real type.

16.2.8. SetObjectGroupValue operation

앞서 13.16 절에서 정의한 바와 같이 SetObjectGroupValue 오퍼레이션은 3 가지 종류의 요청 signature 로 구성됩니다. 첫 번째 signature 의 예는 아래와 같습니다.

As defined in Section 13.16, the SetObjectGroupValue operation has three types of REQUEST signatures. An example of the first signature is as follows.

```
<SetObjectGroupValue name=”some-operation-session-id”>
    <UM3ObjectList name=”parUM3ObjectDestinationList”>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>555</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>nameOfTheObject</UM3ObjectName>
        <UM3ClassIdentifier name=”um3ClassIdentifierIndicator”>666</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectNameIndicator”>nameOfTheBinaryObject</UM3ObjectName>
    </UM3ObjectList>
    <UM3ObjectList name=”parUM3ObjectSourceList”>
        <AnalogSensor name=”nameOfTheObject”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>555</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>nameOfTheObject</UM3ObjectName>
            <UM3Boolean name=”hasBattery”>FALSE</UM3Boolean>
        </AnalogSensor>
        <BinarySensor name=”nameOfTheBinaryObject”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>666</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>nameOfTheBinaryObject</UM3ObjectName>
            <UM3Real name=”powerConsumption”>40.0</UM3Real>
        </BinarySensor>
    </UM3ObjectList>
</SetObjectGroupValue>
```

위의 예는 AnalogSensor 와 BinarySensor 오브젝트를 구성하는 hasBattery 및 powerConsumption 애트리뷰트의 값을 각각 새롭게 설정하기 위한 SetObjectGroupValue 오퍼레이션의 signature 를 나타내고 있습니다. 즉, parUM3ObjectDestinationList 파라메터의 값들은 555 와 nameOfTheObject 의 이름을 갖고 있는 오브젝트와, 666 과 nameOfTheBinaryObject 라는 이름을 갖고 있는 2 개의 오브젝트들에 대해 해당 오퍼레이션이 적용됨을 나타내고 있습니다. 다음으로 각 오브젝트를 구성하는 애트리뷰트의 값들 중 실제로 재설정되어야 하는 애트리뷰트의 종류들을 parUM3ObjectSourceList 파라메터에 나열하고 있습니다. AnalogSensor 클래스의 nameOfTheObject 라는 이름을 갖고 있는 오브젝트는 hasBattery 애트리뷰트의 값을 TRUE 로 재설정토록 정의되어 있으며, BinarySensor 클래스의 nameOfTheBinaryObject 라는 이름을 갖고 있는 오브젝트는 powerConsumption 애트리뷰트의 값을 재설정토록 지정되어 있습니다.

This example shows the signature of SetObjectGroupValue operation to set the values of hasBattery and powerConsumption attribute that configure AnalogSensor and BinarySensor object with new values. The values of parUM3ObjectDestinationList indicate that the operation is applied to two objects, including the one with the class ID of 555 and name of nameOfTheObject and the other one with the class ID of 666 and name of nameOfTheBinaryObject. The types of attributes to be reset among the attributes that configure each object are listed in the parUM3ObjectSourceList parameter. The object of AnalogSensor class that has the name of nameOfTheObject is defined to reset the value of the hasBattery attribute to TRUE, and the object of BinarySensor class that has the name of nameOfTheBinaryObject is defined to reset the value of powerConsumption attribute.

다음으로 위의 예와는 달리 조건문을 파라메터로 넘기는 경우의 예는 다음과 같습니다.

Unlike this example, the conditional statements are passed as parameters in the following example.

```
<SetObjectGroupValue name=”some-operation-session-id”>
    <UM3ObjectValueCondition name=”parUM3ObjectValueCondition”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”>1212</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”>conditionObjectName</UM3ObjectName>
        <UM3CharacterString name=”um3ObjectDescription”>
            This is the condition object which has the condition as its enumerated value with the reference
            values.
        </UM3CharacterString>
        <UM3Real name=”upperEnd”>17.17</UM3Real>
        <UM3Real name=”lowerend”>2.5</UM3Real>
        <condition>GLLU</condition>
        <UM3ClassIdentifier name=”targetAttributeClassIdentifier”>2121</UM3ClassIdentifier>
        <UM3ObjectName name=”targetAttributeObjectName”>attributeName</UM3ObjectName>
    </UM3ObjectValueCondition>
    <UM3ObjectList name=”parUM3ObjectSourceList”>
```

```
        <AnalogSensor name=”nameOfTheObject”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>555</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>nameOfTheObject</UM3ObjectName>
            <UM3Boolean name=”hasBattery”>FALSE</UM3Boolean>
        </AnalogSensor>
        <BinarySensor name=”nameOfTheBinaryObject”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>666</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>nameOfTheBinaryObject</UM3ObjectName>
            <UM3Real name=”powerConsumption”>40.0</UM3Real>
        </BinarySensor>
    </UM3ObjectList>
</SetObjectGroupValue>
```

위의 예는 parUM3ObjectValueCondition 파라메터로 주어진 특정 조건을 만족하는 오브젝트들에 대해서 parUM3ObjectSourceList 파라메터에 주어진 값으로 그 애트리뷰트의 값들을 재설정하기위한 SetObjectGroupValue 오퍼레이션의 signature 를 보여주고 있습니다. 이 때 한가지 주의할 점은 상기 parUM3ObjectValueCondition 으로 주어진 조건에 부합하는 오브젝트라 할지라도 그 오브젝트의 클래스가 AnalogSensor 혹은 BinarySensor 가 아닐 경우에는 해당 재설정 오퍼레이션이 수행되어서는 안된다는 점입니다. 즉, 조건문을 이용해 검출하는 오퍼레이션의 대상 오브젝트들은 반드시 parUM3ObjectSourceList 에 명기된 클래스에 속하는 오브젝트들이어야 한다는 점입니다.

This example shows the signature of the SetObjectGroupValue operation to reset the values of the attributes with the values provided in the parUM3ObjectSourceList parameter for the objects that meet the conditions provided in the parUM3ObjectValueCondition parameter. One thing to be noted here is that the reset operation should not be performed, even when an object meets the conditions provided by parUM3ObjectValueCondition if the class of the object is not AnalogSensor or BinarySensor. In other words, the target objects for the operation to apply the conditional statements should be the ones of the classes that are specified in the parUM3ObjectSourceList.

SetObjectGroupValue 오퍼레이션의 요청 signature OVLD 2 도 이상과 같은 동일한 방법으로 처리할 수 있습니다.

REQUEST signature OVLD 2 of the SetObjectGroupValue operation can be processed in the same way.

상기 SetObjectGroupValue 오퍼레이션에 대한 실패 signature 의 경우 즉, OperationResponse 오퍼레이션 signature 의 경우, 요청 signature 로 주어지는 대상 오브젝트의 리스트들 중 몇 번째 오브젝트에서 에러가 발생하였는가를 나타내어 이를 매니저로 전송하는 signature 가 있습니다.

In case of FAIL signature of the SetObjectGroupValue operation i.e. in case of OperationResponse operation signature, there is a signature that sends the index of the object that caused an error in the list of target objects to the

manager.

만약 상기 첫 번째 요청 signature 의 예로 주어진 오퍼레이션의 수행 도중 발생한 에러를 다음과 같은 실패 signature 를 갖는 OperationResponse 오퍼레이션으로 리턴하는 경우, 이는 nameOfTheBinaryObject 라는 오브젝트 이름을 갖는 BinarySensor 오브젝트의 처리도중 에러가 발생하였음을 나타냅니다.

When an error occurs while executing the operation given as an example of the first REQUEST and it is returned through the OperationResponse operation that has the following FAIL signature, it indicates that the error occurred while processing the BinarySensor object with the object name of nameOfTheBinaryObject.

```
<OperationResponse name=" some-operation-session-id">
    <UM3Boolean name="parResult">FALSE</UM3Boolean>
    <UM3OperationErrorCode name="parUM3OperationErrorCode>
        accessDenied
    </UM3OperationErrorCode>
    <UM3UnsignedInteger16 name="parErrorObjectElementIndex">
        1
    </UM3UnsignedInteger16>
</OperationResponse>
```

즉, 요청 signature 의 파라메터로 주어진 오브젝트의 리스트들 중 첫 번째 오브젝트의 값을 0 으로 두고 해당 오브젝트의 위치를 계산한 값을 parErrorObjectElementIndex 파라메터의 값으로 넘겨주게 됩니다.

The value of the first object in the list of objects provided as a parameter of REQUEST signature is set 0, and the index value of the corresponding object that indicates the location is returned as the value of the parErrorObjectElementIndex parameter.

16.2.9. SetObjectAttributeGroupValue operation

SetObjectAttributeGroupValue 오퍼레이션은 앞서 정의한 바와 같이 특정 오브젝트를 구성하고 있는 애트리뷰트들 중 1 개 이상 복수개의 애트리뷰트들의 값을 설정하기 위해 사용하는 오퍼레이션입니다.

The SetObjectAttributeGroupValue operation is used to set the values of more than one attribute among the attributes of the specific object as defined earlier.

```
<SetObjectAttributeGroupValue name="some-operation-session-id">
    <UM3ClassIdentifier name="parUM3ClassIdentifier">1234</UM3ClassIdentifier>
    <UM3ObjectName name="parUM3ObjectName">nameOfTheObject</UM3ObjectName>
    <UM3ObjectList name="parUM3ObjectList">
        <UM3Boolean name="hasBattery">TRUE</UM3Boolean>
```

```
        <PresentBatteryValue name="presentBatteryValueInfo">
            <UM3ClassIdentifier name="um3ClassIdentifier">9898</UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName">presentBatteryValueInfo</UM3ObjectName>
            <UM3ObjectList name="um3AttributeList">
                <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
                <UM3ObjectName name="um3ObjectName"></UM3ObjectName>
                <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
                <UM3Real name="presentVoltage"></UM3Real>
                <UM3Real name="previousVoltage"></UM3Real>
                <UM3Real name="remainingBatteryTime"></UM3Real>
            </UM3ObjectList>
            <UM3Real name="presentVoltage">17.5</UM3Real>
            <UM3Real name="previousVoltage">15.5</UM3Real>
            <UM3DateTime name="remainingBatteryTime">2592000</UM3DateTime>
        </PresentBatteryValue>
    </UM3ObjectList>
</SetObjectAttributeGroupValue>
```

위의 예에서 보는 바와 같이 parUM3ClassIdentifier 와 parUM3ObjectName 파라메터는 애트리뷰트의 값을 설정하기 위한 대상 오브젝트를 나타냅니다. 위의 예에서는 1234 라는 클래스아이디를 갖고 있으며 그 오브젝트의 이름으로 nameOfTheObject 를 갖고 있는 오브젝트를 지정하고 있습니다.

As shown in this example, the parUM3ClassIdentifier and parUM3ObjectName parameter represent the target object to set the value of the attribute. It points to the object with the class ID of 1234 and object name of nameOfTheObject in this example.

위에서 지정한 클래스 아이디와 오브젝트의 이름을 갖고 있는 오브젝트에 대하여, parUM3ObjectList 파라메터에 기록된 모든 애트리뷰트의 값으로 해당 오브젝트를 구성하고 있는 애트리뷰트의 값들을 모두 재설정하는 과정을 거치게 됩니다. 위의 예에서는 nameOfTheObject 라는 오브젝트 이름을 갖는 오브젝트의 애트리뷰트들 중 PresentBatteryValue 클래스 타입의 presentBatteryValueInfo 애트리뷰트에 대해 그 값을 새롭게 재설정하기 위한 파라메터의 값을 보여주고 있습니다.

All attributes of the object with the class ID and object name specified above are reset by using the values of attributes stored in the parUM3ObjectList parameter. This example shows the value of the parameter to reset the value of the presentBatteryValueInfo attribute of PresentBatteryValue class type among the attributes of the object with the object name of nameOfTheObject.

16.2.10. CreateObject operation

CreateObject 오퍼레이션은 앞서 정의한 바와 같이 새로운 오브젝트를 생성하기 위해 사용하는 오퍼레이션 입니다.

The CreateObject operation is used to create new object as defined earlier.

```
<CreateObject name="some-operation-session-id">
    <UM3CharacterString name="parUM3DnObjectName">
        distinguished.nameOfObject
    </UM3CharacterString>
    <!-- actual object value goes from here -->
    <BinaryControl name="parAnyTypeValue">
        <UM3ClassIdentifier name="um3ClassIdentifier">5656</UM3ClassIdentifier>
.       <UM3ObjectName name="um3ObjectName">myBinaryControl</UM3ObjectName>
        <UM3ObjectList name="um3AttributeList">
            <!-- attribute list of the object goes from here -->
            <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName"></UM3ObjectName>
            <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
            <UM3Boolean name="hasBattery"></UM3Boolean>
            <UM3Boolean name="hasBackupBattery"></UM3Boolean>
            <PresentBatteryValue name="presentBatteryValueInfo"></PresentBatteryValueInfo>
            <UM3Real name="powerConsumption"></UM3Real>
            <UM3UnsignedInteger16 name="levelInAConfigurationTree"></UM3UnsignedInteger16>
            <DeviceMaintenanceScheduleInfo name="operationalTimeInfo">
            </DeviceMaintenanceScheduleInfo>
            <DeviceOperationStatus name="presentControlStatus"></DeviceOperationStatus>
            <UM3CharacterString name="serialNumber"></UM3CharacterString>
            <PresentBinaryControlValue name="presentValue"></PresentBinaryControlValue>
        </UM3ObjectList>
        <UM3Boolean name="hasBattery">FALSE</UM3Boolean>
        <UM3Boolean name="hasBackupBattery">FALSE</UM3Boolean>
        <PresentBatteryValue name="presentBatteryValueInfo">
            <UM3ClassIdentifier name="um3ClassIdentifier">111</UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName">presentBatteryValueAttribute</UM3ObjectName>
            <UM3ObjectList name="um3AttributeList">
                <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
                <UM3ObjectName name="um3ObjectName"></UM3ObjectName>
                <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
                <UM3Real name="presentVoltage"></UM3Real>
                <UM3Real name="previousVoltage"></UM3real>
                <UM3DateTime name="remainingBatteryTime"></UM3DateTime>
            </UM3ObjectList>
            <UM3Real name="presentVoltage">10.5</UM3Real>
            <UM3Real name="previousVoltage">11.5</UM3Real>
            <UM3DateTime name="remainingBatteryTime">10000</UM3DateTime>
        </PresentBatteryValue>
        <UM3Real name="powerConsumption">25</UM3Real>
        <UM3UnsignedInteger16 name="levelInAConfigurationTree">3</UM3UnsignedInteger16>
        <DeviceMaintenanceScheduleInfo name="operationalTimeInfo">
            <UM3ClassIdentifier name="um3ClassIdentifier">4444</UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName">operationalTimeInfo</UM3ObjectName>
```

```
            <UM3ObjectList name=”um3AttributeList”>
                <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                <UM3ObjectName name=”um3ObjectName”></UM3ObjectName>
                <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                <UM3DateTime name=”inServiceDateTime”></UM3DateTime>
                <UM3DateTime name=”durablePeriod”></UM3DateTime>
            </UM3ObjectList>
            <UM3DateTime name=”inSerivceDateTime”>1100011000</UM3DateTime>
            <UM3DateTime name=”durablePeriod”>11000000</UM3DateTime>
        </DeviceMaintenanceScheduleInfo>
        <DeviceOperationStatus name=”presentControlStatus”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>5555</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>presentControlStatus</UM3ObjectName>
            <UM3ObjectList name=”um3AttributeList”>
                <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
                <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                <presentStatus> </presentStatus>
                <UM3DateTime name=”statusTimestamp”></UM3DateTime>
                <UM3DateTime name=”resumeDateTime”></UM3DateTime>
            </UM3ObjectList>
            <presentStatus>outOfService</presentStatus>
            <UM3DateTime name=”statusTimestamp”>2345678</UM3DateTime>
            <UM3DateTime name=”resumeDateTime”>2245678</UM3DateTime>
        </DeviceOperationStatus>
        <UM3CharacterString name=”serialNumber”>MY12341234</UM3CharacterString>
        <PresentBinaryControlValue name=”presentValue”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>6666</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>presentValue</UM3ObjectName>
            <UM3ObjectList name=”um3AttributeList”>
                <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
                <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                <UM3Boolean name=”presentValue”> </UM3Boolean>
                <UM3DateTime name=”presentValueTimestamp”></UM3DateTime>
                <UM3Boolean name=”previousValue”> </UM3Boolean>
                <UM3DateTime name=previousValueTimestamp”></UM3DateTime>
                <UM3UnsignedInteger16 name=”stateChangedCount”></UM3UnsignedInteger16>
                <UM3DateTime name=”stateCountStartedDateTime”></UM3DateTime>
            </UM3ObjectList>
            <UM3Boolean name=”presentValue”>TRUE</UM3Boolean>
            <UM3DateTime name=”presentValueTimestamp”>111222333444</UM3DateTime>
            <UM3Boolean name=”previousValue”>FALSE</UM3Boolean>
            <UM3DateTime name=previousValueTimestamp”>11112222333400</UM3DateTime>
            <UM3UnsignedInteger16 name=”stateChangedCount”>300</UM3UnsignedInteger16>
            <UM3DateTime name=”stateCountStartedDateTime”>11112222300000</UM3DateTime>
        </PresentBinaryControlValue>
    </BinaryControl>
</CreateObject>
```

위의 예에서 보는바와 같이 CreateObject 오퍼레이션을 위해서는 완전한 형태의 오브젝트를 파라메터로 넘겨주어야 합니다.

As shown in this example, the complete form of the object should be passed as a parameter for CreateObject operation.

16.2.11. CreateObjectGroup operation

CreateObjectGroup 오퍼레이션은 1 개 이상 복수개의 오브젝트를 생성하기 위해 사용됩니다.

The CreateObjectGroup operation is used to create one or more objects.

```
<CreateObjectGroup name=”operation-session-identifier”>
    <UM3UnsignedInteger16 name=”parNoOfObjectToBeCreated”>2</UM3UnsignedIngeger16>
    <UM3ObjectListToBeCreated name=”parUM3ObjectListToBeCreated”>
        <UM3SequenceOf name=””>
            <UM3Sequence name=””>
                <UM3CharacterString name=”dn”>
                    distinguished.nameOfObject
                </UM3CharacterString>
                <!-- actual object value goes from here -->
                <BinaryControl name=”anyTypeValue”>
                    <UM3ClassIdentifier name=”um3ClassIdentifier”>5656</UM3ClassIdentifier>
                    <UM3ObjectName name=”um3ObjectName”>myBinaryControl-0001</UM3ObjectName>
                    <UM3ObjectList name=”um3AttributeList”>
                        <!-- attribute list of the object goes from here -->
                        <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                        <UM3ObjectName name=”um3ObjectName”></UM3ObjectName>
                        <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                        <UM3Boolean name=”hasBattery”></UM3Boolean>
                        <UM3Boolean name=”hasBackupBattery”></UM3Boolean>
                        <PresentBatteryValue name=”presentBatteryValueInfo”></PresentBatteryValueInfo>
                        <UM3Real name=”powerConsumption”></UM3Real>
                        <UM3UnsignedInteger16 name=”levelInAConfigurationTree”>
                        </UM3UnsignedInteger16>
                        <DeviceMaintenanceScheduleInfo name=”operationalTimeInfo”>
                        </DeviceMaintenanceScheduleInfo>
                        <DeviceOperationStatus name=”presentControlStatus”></DeviceOperationStatus>
                        <UM3CharacterString name=”serialNumber”></UM3CharacterString>
                        <PresentBinaryControlValue name=”presentValue”></PresentBinaryControlValue>
                    </UM3ObjectList>
                    <UM3Boolean name=”hasBattery”>FALSE</UM3Boolean>
                    <UM3Boolean name=”hasBackupBattery”>FALSE</UM3Boolean>
                    <PresentBatteryValue name=”presentBatteryValueInfo”>
                        <UM3ClassIdentifier name=”um3ClassIdentifier”>111</UM3ClassIdentifier>
```

```
            <UM3ObjectName name=”um3ObjectName”>
                presentBatteryValueAttribute
            </UM3ObjectName>
            <UM3ObjectList name=”um3AttributeList”>
                <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                <UM3ObjectName name=”um3ObjectName”></UM3ObjectName>
                <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                <UM3Real name=”presentVoltage”></UM3Real>
                <UM3Real name=”previousVoltage”></UM3real>
                <UM3DateTime name=”remainingBatteryTime”></UM3DateTime>
            </UM3ObjectList>
            <UM3Real name=”presentVoltage”>10.5</UM3Real>
            <UM3Real name=”previousVoltage”>11.5</UM3Real>
            <UM3DateTime name=”remainingBatteryTime”>10000</UM3DateTime>
        </PresentBatteryValue>
        <UM3Real name=”powerConsumption”>25</UM3Real>
        <UM3UnsignedInteger16 name=”levelInAConfigurationTree”>3</UM3UnsignedInteger16>
        <DeviceMaintenanceScheduleInfo name=”operationalTimeInfo”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>4444</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>operationalTimeInfo</UM3ObjectName>
            <UM3ObjectList name=”um3AttributeList”>
                <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                <UM3ObjectName name=”um3ObjectName”></UM3ObjectName>
                <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                <UM3DateTime name=”inServiceDateTime”></UM3DateTime>
                <UM3DateTime name=”durablePeriod”></UM3DateTime>
            </UM3ObjectList>
            <UM3DateTime name=”inSerivceDateTime”>1100011000</UM3DateTime>
            <UM3DateTime name=”durablePeriod”>11000000</UM3DateTime>
        </DeviceMaintenanceScheduleInfo>
        <DeviceOperationStatus name=”presentControlStatus”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>5555</UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”>
                presentControlStatus
            </UM3ObjectName>
            <UM3ObjectList name=”um3AttributeList”>
                <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
                <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                <presentStatus> </presentStatus>
                <UM3DateTime name=”statusTimestamp”></UM3DateTime>
                <UM3DateTime name=”resumeDateTime”></UM3DateTime>
                    </UM3ObjectList>
            <presentStatus>outOfService</presentStatus>
            <UM3DateTime name=”statusTimestamp”>2345678</UM3DateTime>
            <UM3DateTime name=”resumeDateTime”>2245678</UM3DateTime>
        </DeviceOperationStatus>
    <UM3CharacterString name=”serialNumber”>MY12341234</UM3CharacterString>
    <PresentBinaryControlValue name=”presentValue”>
```

```
            <UM3ClassIdentifier name="um3ClassIdentifier">6666</UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName">presentValue</UM3ObjectName>
            <UM3ObjectList name="um3AttributeList">
                <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
                <UM3ObjectName name="um3ObjectName"> </UM3ObjectName>
                <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
                <UM3Boolean name="presentValue"> </UM3Boolean>
                <UM3DateTime name="presentValueTimestamp"></UM3DateTime>
                <UM3Boolean name="previousValue"> </UM3Boolean>
                <UM3DateTime name=previousValueTimestamp"></UM3DateTime>
                <UM3UnsignedInteger16 name="stateChangedCount"></UM3UnsignedInteger16>
                <UM3DateTime name="stateCountStartedDateTime"></UM3DateTime>
            </UM3ObjectList>
            <UM3Boolean name="presentValue">TRUE</UM3Boolean>
            <UM3DateTime name="presentValueTimestamp">111222333444</UM3DateTime>
            <UM3Boolean name="previousValue">FALSE</UM3Boolean>
            <UM3DateTime name=previousValueTimestamp">11112222333400</UM3DateTime>
            <UM3UnsignedInteger16 name="stateChangedCount">300</UM3UnsignedInteger16>
            <UM3DateTime name="stateCountStartedDateTime">11112222300000</UM3DateTime>
        </PresentBinaryControlValue>
    </BinaryControl>
</UM3Sequence>

<UM3Sequence name="">
    <UM3CharacterString name="dn">
        distinguished.nameOfObject
    </UM3CharacterString>
    <!-- actual object value goes from here -->
    <BinaryControl name="anyTypeValue">
        <UM3ClassIdentifier name="um3ClassIdentifier">5656</UM3ClassIdentifier>
.       <UM3ObjectName name="um3ObjectName">myBinaryControl-0002</UM3ObjectName>
        <UM3ObjectList name="um3AttributeList">
            <!-- attribute list of the object goes from here -->
            <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName"></UM3ObjectName>
            <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
            <UM3Boolean name="hasBattery"></UM3Boolean>
            <UM3Boolean name="hasBackupBattery"></UM3Boolean>
            <PresentBatteryValue name="presentBatteryValueInfo"></PresentBatteryValueInfo>
            <UM3Real name="powerConsumption"></UM3Real>
            <UM3UnsignedInteger16 name="levelInAConfigurationTree"></UM3UnsignedInteger16>
            <DeviceMaintenanceScheduleInfo name="operationalTimeInfo">
            </DeviceMaintenanceScheduleInfo>
            <DeviceOperationStatus name="presentControlStatus"></DeviceOperationStatus>
            <UM3CharacterString name="serialNumber"></UM3CharacterString>
            <PresentBinaryControlValue name="presentValue"></PresentBinaryControlValue>
        </UM3ObjectList>
        <UM3Boolean name="hasBattery">FALSE</UM3Boolean>
        <UM3Boolean name="hasBackupBattery">FALSE</UM3Boolean>
        <PresentBatteryValue name="presentBatteryValueInfo">
```

```
    <UM3ClassIdentifier name=”um3ClassIdentifier”>111</UM3ClassIdentifier>
    <UM3ObjectName name=”um3ObjectName”>
        presentBatteryValueAttribute
    </UM3ObjectName>
    <UM3ObjectList name=”um3AttributeList”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”></UM3ObjectName>
        <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
        <UM3Real name=”presentVoltage”></UM3Real>
        <UM3Real name=”previousVoltage”></UM3real>
        <UM3DateTime name=”remainingBatteryTime”></UM3DateTime>
    </UM3ObjectList>
    <UM3Real name=”presentVoltage”>10.5</UM3Real>
    <UM3Real name=”previousVoltage”>11.5</UM3Real>
    <UM3DateTime name=”remainingBatteryTime”>10000</UM3DateTime>
</PresentBatteryValue>

<UM3Real name=”powerConsumption”>25</UM3Real>

<UM3UnsignedInteger16 name=”levelInAConfigurationTree”>3</UM3UnsignedInteger16>

<DeviceMaintenanceScheduleInfo name=”operationalTimeInfo”>
    <UM3ClassIdentifier name=”um3ClassIdentifier”>4444</UM3ClassIdentifier>
    <UM3ObjectName name=”um3ObjectName”>operationalTimeInfo</UM3ObjectName>
    <UM3ObjectList name=”um3AttributeList”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”></UM3ObjectName>
        <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
        <UM3DateTime name=”inServiceDateTime”></UM3DateTime>
        <UM3DateTime name=”durablePeriod”></UM3DateTime>
    </UM3ObjectList>
    <UM3DateTime name=”inSerivceDateTime”>1100011000</UM3DateTime>
    <UM3DateTime name=”durablePeriod”>11000000</UM3DateTime>
</DeviceMaintenanceScheduleInfo>

<DeviceOperationStatus name=”presentControlStatus”>
    <UM3ClassIdentifier name=”um3ClassIdentifier”>5555</UM3ClassIdentifier>
    <UM3ObjectName name=”um3ObjectName”>presentControlStatus</UM3ObjectName>
    <UM3ObjectList name=”um3AttributeList”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
        <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
        <presentStatus> </presentStatus>
        <UM3DateTime name=”statusTimestamp”></UM3DateTime>
        <UM3DateTime name=”resumeDateTime”></UM3DateTime>
            </UM3ObjectList>
            <presentStatus>outOfService</presentStatus>
            <UM3DateTime name=”statusTimestamp”>2345699</UM3DateTime>
            <UM3DateTime name=”resumeDateTime”>2245008</UM3DateTime>
        </DeviceOperationStatus>

        <UM3CharacterString name=”serialNumber”>MY12341235</UM3CharacterString>

        <PresentBinaryControlValue name=”presentValue”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”>6666</UM3ClassIdentifier>
```

```
                    <UM3ObjectName name=”um3ObjectName”>presentValue</UM3ObjectName>
                    <UM3ObjectList name=”um3AttributeList”>
                        <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
                        <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
                        <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
                        <UM3Boolean name=”presentValue”> </UM3Boolean>
                        <UM3DateTime name=”presentValueTimestamp”></UM3DateTime>
                        <UM3Boolean name=”previousValue”> </UM3Boolean>
                        <UM3DateTime name=previousValueTimestamp”></UM3DateTime>
                        <UM3UnsignedInteger16 name=”stateChangedCount”></UM3UnsignedInteger16>
                        <UM3DateTime name=”stateCountStartedDateTime”></UM3DateTime>
                    </UM3ObjectList>
                    <UM3Boolean name=”presentValue”>FALSE</UM3Boolean>
                    <UM3DateTime name=”presentValueTimestamp”>111222333444</UM3DateTime>
                    <UM3Boolean name=”previousValue”>FALSE</UM3Boolean>
                    <UM3DateTime name=previousValueTimestamp”>11112222333400</UM3DateTime>
                    <UM3UnsignedInteger16 name=”stateChangedCount”>765</UM3UnsignedInteger16>
                    <UM3DateTime name=”stateCountStartedDateTime”>
                        11112222300000
                    </UM3DateTime>
                </PresentBinaryControlValue>
            </BinaryControl>
        </UM3Sequence>
    </UM3SequenceOf>
</UM3ObjectListToBeCreated>
</CreateObjectGroup>
```

위의 예는 2 개의 BinaryControl 오브젝트를 생성하기 위해 CreateObjectGroup 오퍼레이션을 사용하는 경우의 signature 를 보여주고 있습니다.

This example shows the signature when the CreateObjectGroup operation is used to create two BinaryControl objects.

16.2.12. DeleteObject operation

DeleteObject 오퍼레이션의 signature 예는 다음과 같습니다.

An example signature of the DeleteObject operation is as follows.

```
<DeleteObject name=”some-session-id”>
    <UM3ClassIdentifier name=”um3ClassIdentifier”>1212</UM3ClassIdentifier>
    <UM3ObjectName name=”um3ObjectName”>myObject-001</UM3ObjectName>
</DeleteObject>
```

16.2.13. DeleteObjectGroup operation

DeleteObjectGroup 오퍼레이션은 특정 오브젝트를 삭제하기 위해 사용하는 오퍼레이션입니다.

DeleteObjectGroup operation is used to delete specific object.

```
<DeleteObjectGroup name=”some-session-id”>
    <UM3ObjectList name=”parUM3ObjectList”>
        <!-- UM3ObjectIndicator type element goes here -->
        <UM3ClassIdentifier name=”um3ClassIdentifier”>1212</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”>myObject-001</UM3ObjectName>
        <UM3ClassIdentifier name=”um3ClassIdentifier”>1212</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”>myObject-002</UM3ObjectName>
    </UM3ObjectList>
</DeleteObjectGroup>
```

위의 예에서는 클래스 아이디 1212 로 정의되는 myObject-001 과 myObject-002 등 두 개의 오브젝트를 삭제하기 위한 DeleteObjectGroup 오퍼레이션의 예를 보여주고 있습니다.

This example shows DeleteObjectGroup operation for deleting two objects of myObject-001 and myObject-002 defined as the class ID of 1212.

16.2.14. CancelGetObjectValue operation

다음은 CancelGetObjectValue 오퍼레이션의 signature 예를 나타내고 있습니다. 예에서 보는 바와 같이 UM3 SER 을 이용한 인코딩과 동일한 종류의 파라메터들이 사용됩니다.

The following is the example signature of CancelGetObjectValue operation. As shown in this example, the parameters of the same type as the encoding using UM3 SER are used.

```
<CancelGetObjectValue name=”present-session-id”>
    <UM3ObjectName name=”parUM3OperationToBeCancelled”>previous-session-id</UM3ObjectName>
</CancelGetObject>
```

16.2.15. CancelGetObjectGroupValue operation

CancelGetObjectGroupValue 오퍼레이션은 CancelGetObjectValue 오퍼레이션과 동일한 형태의 signature 로 활용됩니다. 즉, 취소하고자 하는 오퍼레이션의 이름 혹은 오퍼레이션의 세션 아이디를 파라메터로 넘겨줍니다.

CancelGetObjectGroupValue operation is used as signature of the same type as the CancelGetObjectValue operation. Name of operation or session ID of the operation to cancel is passed as parameter.

16.2.16. CancelGetObjectAttributeValue operation

CancelGetObjectValue 와 동일한 signature 로 사용됩니다.

It is used as same signature as CancelGetObjectValue.

16.2.17. CancelGetObjectAttributeGroupValue operation

CancelGetObjectValue 와 동일한 signature 로 사용됩니다.

It is used as same signature as CancelGetObjectValue.

16.2.18. RequestChangeOfAttributeValueReport operation

RequestChangeOfAttributeValueReport 오퍼레이션의 요청 signature 는 다음의 예와 같은 형태로 사용됩니다.

REQUEST signature of RequestChangeOfAttributeValueReport operation is used as shown in the following example.

```
<RequestChangeOfAttributeValueReport name=”session-id”>
    <UM3ActionBroker name=”parUM3ActionBroker”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”>555</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”>parUM3ActionBroker</UM3ObjectName>
        <UM3ObjectList name=”um3AttributeList”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
            <UM3ClassIdentifier name=”targetObjectClassIdentifier”></UM3ClassIdentifier>
            <UM3ObjectName name=”targetObjectObjectName”> </UM3ObjectName>
            <UM3ClassIdentifier name=”targetAttributeClassIdentifier”></UM3ClassIdentifier>
            <UM3ObjectName name=”targetAttributeObjectName”> </UM3ObjectName>
            <UM3DateTime name=”targetPeriod”></UM3DateTime>
            <UM3Boolean name=”isOnlyOnceAction”> </UM3Boolean>
            <UM3DateTime name=”reservedActionTimestamp”></UM3DateTime>
            <UM3SequenceOf name=”recipientAddress”></UM3SequenceOf>
            <UM3ObjectValueCondition name=”targetCondition”> </UM3ObjectValueCondition>
        </UM3ObjectList>
        <UM3ClassIdentifier name=”targetObjectClassIdentifier”>8888</UM3ClassIdentifier>
        <UM3ObjectName name=”targetObjectObjectName”>targetObjectName</UM3ObjectName>
        <UM3ClassIdentifier name=”targetAttributeClassIdentifier”>99</UM3ClassIdentifier>
        <UM3ObjectName name=”targetAttributeObjectName”>targetAttributeName</UM3ObjectName>
        <UM3DateTime name=”targetPeriod”>300</UM3DateTime>
        <UM3Boolean name=”isOnlyOnceAction”>TRUE</UM3Boolean>
        <UM3DateTime name=”reservedActionTimestamp”>111222333444</UM3DateTime>
```

```
        <UM3SequenceOf name=”recipientAddress”>.
            <UM3CharacterString name=”recipientAddress01”>10.10.10.10</UM3CharacterString>
            <UM3CharacterString name=”recipientAddress02”>10.10.10.11</UM3CharacterString>
        </UM3SequenceOf>
        <UM3ObjectValueCondition name=”targetCondition”>COV</UM3ObjectValueCondition>
    </UM3ActionBroker>
</RequestChangeOfAttributeValueReport>
```

16.2.19. RequestConditionDetectedReport operation

RequestConditionDetectedReport 오퍼레이션의 예는 16.2.18 절의 예와 동일하며, 단 <UM3ObjectValueCondtion></UM3ObjectValueCondition> 의 값이 COV 뿐만 아니라 다른 조건들을 나타내는 값들이 올 수 있다는 점만 차이가 있습니다.

This example of RequestConditionDetectedReport operation is the same as the example in Section 16.1.18, and the only difference is that the value of <UM3ObjectValueCondtion></UM3ObjectValueCondition> can be set to COV as well as the other value used for specifying conditions.

16.2.20. ChangeOfAttributeValueReport operation

ChangeOfAttributeValueReport 오퍼레이션은 앞서 정의한 바와 같이 RequestChangeOfAttributeValueReport 오퍼레이션에 대한 응답 오퍼레이션이며 그 예는 다음과 같습니다.

The ChangeOfAttributeValueReport operation is the response operation to the RequestChangeOfAttributeValueReport operation as defined earlier. An example is as follows.

```
<ChangeOfAttributeValueReport name=”previous-session-id”>
    <UM3EventReport name=”parUM3EventReport”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”>555</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”>parUM3EventReport</UM3ObjectName>
        <UM3ObjectList name=”um3AttributeList”>
            <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
            <UM3ClassIdentifier name=”targetObjectClassIdentifier”></UM3ClassIdentifier>
            <UM3ObjectName name=”targetObjectObjectName”> </UM3ObjectName>
            <UM3ClassIdentifier name=”targetAttributeClassIdentifier”></UM3ClassIdentifier>
            <UM3ObjectName name=”targetAttributeObjectName”> </UM3ObjectName>
            <UM3Real name=”targetAttributeValue”></UM3Real>
            <UM3DateTime name=”eventTimestamp”></UM3DateTime>
        </UM3ObjectList>
        <UM3ClassIdentifier name=”targetObjectClassIdentifier”>8888</UM3ClassIdentifier>
        <UM3ObjectName name=”targetObjectObjectName”>targetObjectName</UM3ObjectName>
        <UM3ClassIdentifier name=”targetAttributeClassIdentifier”>99</UM3ClassIdentifier>
        <UM3ObjectName name=”targetAttributeObjectName”>targetAttributeName</UM3ObjectName>
        <UM3Real name=”targetAttributeValue”>15.7</UM3Real>
```

```
        <UM3DateTime name="eventTimestamp">1111222233334444</UM3DateTime>
    </UM3EventReport>
</ChangeOfAttributeValueReport>
```

16.2.21. ConditionDetectedReport operation

상기 ChangeOfAttributeValueReport 오퍼레이션과 동일한 형식으로 구성됩니다.

Its configuration is the same as the ChangeOfAttributeValueReport operation.

16.2.22. CancelEventReport operation

CancelEventReport 오퍼레이션의 요청 signature 의 예는 다음과 같습니다.

An example of the REQUEST signature of the CancelEventReport operation is as follows.

```
<CancelEventReport name="session-id">
    <UM3CharacterString name="parUM3SessionIdentifier">
        Some-session-id-to-be-cancelled
    </UM3CharacterString>
</CancelEventReport>
```

16.3. XML representation of UM3 primitive type

앞서 정의하고 기술한 XML 데이터 표현방식과 마찬가지로 UM3 프로토콜을 구성하는 UM3 Primitive type 을 XML 방식의 데이터로 표현하는 예를 기술합니다. 각 UM3 기본 타입별 XML schema 의 정의는 16.5 절을 참조하시기 바랍니다.

Like the XML data representation method described earlier, an example representation of UM3 Primitive type with XML type data is described. Refer to Section 16.4 for the definition of XML schema by UM3 primitive type.

16.3.1. UM3Integer16 type

만약 특정 변수가 UM3Integer16 타입으로 정의되어 있으며 해당 변수의 이름이 'myInteger' 이고, 그 값이 정수 17 로 정의된다면 이를 다음과 같은 XML 데이터로 표현할 수 있습니다.

If a specific variable is UM3Integer16 type, the name of the corresponding variable is 'myInteger', and the value is integer 17, then it can be represented with XML data as follows.

```
<myInteger>17</myInteger>
```

참고로 UM3Integer16 타입은 본 권고안의 16.5 절이 정의하는 XML 스키마에 다음과 같이 정의되어 있습니다.

For reference, the UM3Integer16 type as defined in XML schema in Section 16.4 of this recommendation is defined as follows.

```
<xs:element name="UM3Integer16" type="xs:short"/>
<xs:element name="UM3Integer16" type="xs:short"/>

<xs:attribute name="name" type="xs:string"/>
<xs:attribute name="um3ObjectIdentifier" type="xs:string"/>
```

즉, 새롭게 정의되는 XML 엘리먼트의 이름은 'UM3Integer16' 이며 이는 short 타입 혹은 short int 타입임을 나타낸다는 뜻으로 정의되어 있습니다. 또한 해당 XML 엘리먼트가 활용할 수 있는 XML 애트리뷰트들 중에는 'name' 이라는 이름을 갖고 있는 애트리뷰트가 있으며 그 애트리뷰트의 타입은 string 타입임을 정의하고 있습니다.

This means that the name of newly defined XML element is 'UM3Integer16', and it is short type or short int type. There is an attribute that has the name of 'name' among the XML attributes that can be utilized by the corresponding XML element, and the corresponding attribute is string type.

본 권고안이 정의하는 다른 정수 타입들 즉, UM3Integer32, UM3Integer64, UM3UnsignedInteger16, UM3UnsignedInteger32, UM3UnsignedInteger64 타입들도 동일한 형식으로 정의되고 사용됩니다.

Other integer types defined in this recommendation such as UM3Integer32, UM3Integer64, UM3UnsignedInteger16, UM3UnsignedInteger32, and UM3UnsignedInteger64 type can be defined and used in the same manner.

16.3.2. UM3Ineger32 type

상기 UM3Integer16 타입과 동일합니다.

This is the same as the above UM3Integer16 type.

16.3.3. UM3Ineger64 type

상기 UM3Integer16 타입과 동일합니다.

This is the same as the above UM3Integer16 type.

16.3.4. UM3UnsignedInteger16 type

상기 UM3Integer16 타입과 동일합니다.

This is the same as the above UM3Integer16 type.

16.3.5. UM3UnsignedInteger32 type

상기 UM3Integer16 타입과 동일합니다.

This is the same as the above UM3Integer16 type.

16.3.6. UM3UnsignedInteger64 type

상기 UM3Integer16 타입과 동일합니다.

This is the same as the above UM3Integer16 type.

16.3.7. UM3CharacterString type

UM3CharacterString 타입은 앞서 정의한 바와 같이 주로 ASCII 코드로 이루어진 혹은 UTF-8 텍스트인코딩 등 여러가지 텍스트인코딩 방식으로 이루어진 문자열을 나타내기 위해 사용됩니다.

UM3CharacterString type is used to represent strings comprised of various text encoding methods including ASCII code or UTF-8 text encoding as defined earlier.

특정 문자열을 저장하기 위한 변수의 이름이 paszString 이고 그 변수에 저장되는 값이 “I am UM3.” 라면 XML 데이터 양식으로의 표현은 아래의 예와 같습니다.

When the name of a variable to store a specific string is paszString and if the value stored in the variable is “I am UM3.”, then the representation in XML format is as follows.

<UM3CharacterString name=”paszString”>I am UM3.</UM3CharacterString>

이하 정의되는 UM3BitString 타입과 UM3OctetString 타입은 UM3CharacterString 타입과는 다른 형태로 정의되고 사용됩니다. UM3CharacterString 은 mull terminated string 타입임에 유의해야 합니다.

UM3BitString type and UM3OctetString type defined below is defined and used in a different way from the UM3CharacterString type. Take caution because UM3CharacterString is mull terminated string type.

16.3.8. UM3BitString type

UM3BitString 타입은 binary 데이터 타입을 전송하기 위해 사용합니다.

UM3BitString type is used to transmit binary data type.

<UM3BitString name=”pachString”>9*9*(^I;laslllksd</UM3CharacterString>

16.3.9. UM3OctetString type

<UM3OctetString name=”pachString”>al;skdfksdfwlkasafd</UM3CharacterString>

16.3.10. UM3Boolean type

UM3Boolean 타입은 TRUE 혹은 FALSE 등 두 가지의 값을 갖는 데이터 타입이며 그 사용예는 다음과 같습니다.

UM3Boolean type has two values of TRUE or FALSE, and an example of its use is as follows.

<UM3Boolean name=”some-name”>TRUE</UM3Boolean>

16.3.11. UM3Real type

UM3Real 타입의 사용예는 다음과 같습니다.

An example of the use of UM3Real type is as follows.

<UM3Real name=”myName”>12.117</UM3Real>

16.3.12. UM3Null type

UM3Null 타입의 사용예는 다음과 같습니다.

An example of the use of UM3Null type is as follows.

<UM3Null name=”nameOfTheType”></UM3NULL>

16.3.13. UM3DateTime type

UM3DateTime 타입의 예는 다음과 같습니다.

An example of the use of UM3DateTime type is as follows.

<UM3DateTime name=”yourTime”>1020304050</UM3DateTime>

혹은 XML 엘리먼트의 애트리뷰트를 별도로 표기하는 다음과 같은 형식으로 표현할 수도 있습니다.

Alternatively, it can be expressed as follows where the attribute of XML element is separately specified.

```
<UM3DateTime>
    <name>yourTime</name>
    <value>1020304050</value>
</UM3DateTime>
```

혹은 value 애트리뷰트의 값을 다음과 같이 표현할 수도 있습니다.

The value of value attribute may also be expressed as follows.

<UM3DateTime name=”yourTime” value=”1020304050”/>

이상과 같은 XML 애트리뷰트의 표현 방식은 본 권고안이 정의하는 모든 타입에 동일하게 적용할 수 있습니다.

This type of XML attribute representation method can be applied to any type defined in this recommendation.

16.3.14. UM3ClassIdentifier type

UM3ClassIdentifier 타입의 사용예는 다음과 같습니다.

An example of the use of UM3ClassIdentifier type is as follows.

<UM3ClassIdentifier name=”um3ClassIdentifier”>3333</UM3ClassIdentifier>

16.3.15. UM3ObjectName type

UM3ObjectName 타입의 사용예는 다음과 같습니다.

An example of the use of UM3ObjectName type.

```
<UM3ObjectName name=”um3ObjectName”>nameOfTheObject</UM3ObjectName>
```

16.4. XML representation of service management information model and application service information model

서비스 관리 정보모델과 응용서비스 정보모델의 XML 표현은 앞서 정의한 통신서비스모델의 XML 표현에서 나열한 오브젝트의 표현과 동일합니다.

예를 들어 BinaryControl 클래스 타입 오브젝트는 아래와 같은 형식의 XML 로 표현할 수 있습니다.

XML representation of service management information model and application service information model is the same as the definition of object listed in the XML representation of the communication service model defined earlier.

For example, the BinaryControl class type object can be represented as XML format as follows

```
<BinaryControl>
    <UM3ClassIdentifier name=”um3ClassIdentifier”>5656</UM3ClassIdentifier>
.   <UM3ObjectName name=”um3ObjectName”>myBinaryControl-0002</UM3ObjectName>
    <UM3ObjectList name=”um3AttributeList”>
        <!-- attribute list of the object goes from here -->
        <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”></UM3ObjectName>
        <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
        <UM3Boolean name=”hasBattery”></UM3Boolean>
        <UM3Boolean name=”hasBackupBattery”></UM3Boolean>
        <PresentBatteryValue name=”presentBatteryValueInfo”></PresentBatteryValueInfo>
        <UM3Real name=”powerConsumption”></UM3Real>
        <UM3UnsignedInteger16 name=”levelInAConfigurationTree”></UM3UnsignedInteger16>
        <DeviceMaintenanceScheduleInfo name=”operationalTimeInfo”>
        </DeviceMaintenanceScheduleInfo>
        <DeviceOperationStatus name=”presentControlStatus”></DeviceOperationStatus>
        <UM3CharacterString name=”serialNumber”></UM3CharacterString>
        <PresentBinaryControlValue name=”presentValue”></PresentBinaryControlValue>
    </UM3ObjectList>
    <UM3Boolean name=”hasBattery”>FALSE</UM3Boolean>
    <UM3Boolean name=”hasBackupBattery”>FALSE</UM3Boolean>
    <PresentBatteryValue name=”presentBatteryValueInfo”>
        <UM3ClassIdentifier name=”um3ClassIdentifier”>111</UM3ClassIdentifier>
        <UM3ObjectName name=”um3ObjectName”>
```

```
            presentBatteryValueAttribute
        </UM3ObjectName>
        <UM3ObjectList name="um3AttributeList">
            <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName"></UM3ObjectName>
            <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
            <UM3Real name="presentVoltage"></UM3Real>
            <UM3Real name="previousVoltage"></UM3real>
            <UM3DateTime name="remainingBatteryTime"></UM3DateTime>
        </UM3ObjectList>
        <UM3Real name="presentVoltage">10.5</UM3Real>
        <UM3Real name="previousVoltage">11.5</UM3Real>
        <UM3DateTime name="remainingBatteryTime">10000</UM3DateTime>
    </PresentBatteryValue>
    <UM3Real name="powerConsumption">25</UM3Real>
    <UM3UnsignedInteger16 name="levelInAConfigurationTree">3</UM3UnsignedInteger16>
    <DeviceMaintenanceScheduleInfo name="operationalTimeInfo">
        <UM3ClassIdentifier name="um3ClassIdentifier">4444</UM3ClassIdentifier>
        <UM3ObjectName name="um3ObjectName">operationalTimeInfo</UM3ObjectName>
        <UM3ObjectList name="um3AttributeList">
            <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName"></UM3ObjectName>
            <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
            <UM3DateTime name="inServiceDateTime"></UM3DateTime>
            <UM3DateTime name="durablePeriod"></UM3DateTime>
        </UM3ObjectList>
        <UM3DateTime name="inSerivceDateTime">1100011000</UM3DateTime>
        <UM3DateTime name="durablePeriod">11000000</UM3DateTime>
    </DeviceMaintenanceScheduleInfo>
    <DeviceOperationStatus name="presentControlStatus">
        <UM3ClassIdentifier name="um3ClassIdentifier">5555</UM3ClassIdentifier>
        <UM3ObjectName name="um3ObjectName">presentControlStatus</UM3ObjectName>
        <UM3ObjectList name="um3AttributeList">
            <UM3ClassIdentifier name="um3ClassIdentifier"></UM3ClassIdentifier>
            <UM3ObjectName name="um3ObjectName"> </UM3ObjectName>
            <UM3ObjectList name="um3AttributeList"></UM3ObjectList>
            <presentStatus> </presentStatus>
            <UM3DateTime name="statusTimestamp"></UM3DateTime>
            <UM3DateTime name="resumeDateTime"></UM3DateTime>
        </UM3ObjectList>
        <presentStatus>outOfService</presentStatus>
        <UM3DateTime name="statusTimestamp">2345699</UM3DateTime>
        <UM3DateTime name="resumeDateTime">2245008</UM3DateTime>
    </DeviceOperationStatus>
    <UM3CharacterString name="serialNumber">MY12341235</UM3CharacterString>
    <PresentBinaryControlValue name="presentValue">
        <UM3ClassIdentifier name="um3ClassIdentifier">6666</UM3ClassIdentifier>
        <UM3ObjectName name="um3ObjectName">presentValue</UM3ObjectName>
        <UM3ObjectList name="um3AttributeList">
```

```
            <UM3ClassIdentifier name=”um3ClassIdentifier”></UM3ClassIdentifier>
            <UM3ObjectName name=”um3ObjectName”> </UM3ObjectName>
            <UM3ObjectList name=”um3AttributeList”></UM3ObjectList>
            <UM3Boolean name=”presentValue”> </UM3Boolean>
            <UM3DateTime name=”presentValueTimestamp”></UM3DateTime>
            <UM3Boolean name=”previousValue”> </UM3Boolean>
            <UM3DateTime name=previousValueTimestamp”></UM3DateTime>
            <UM3UnsignedInteger16 name=”stateChangedCount”></UM3UnsignedInteger16>
            <UM3DateTime name=”stateCountStartedDateTime”></UM3DateTime>
        </UM3ObjectList>
        <UM3Boolean name=”presentValue”>FALSE</UM3Boolean>
        <UM3DateTime name=”presentValueTimestamp”>111222333444</UM3DateTime>
        <UM3Boolean name=”previousValue”>FALSE</UM3Boolean>
        <UM3DateTime name=previousValueTimestamp”>11112222333400</UM3DateTime>
        <UM3UnsignedInteger16 name=”stateChangedCount”>765</UM3UnsignedInteger16>
        <UM3DateTime name=”stateCountStartedDateTime”>11112222300000</UM3DateTime>
    </PresentBinaryControlValue>
</BinaryControl>
```

16.5. XML 스키마의 정의

XML 스키마 (schema) 는 본 권고안이 정의하는 정보모델, 통신서비스모델 등을 XML 로 표현하기 위한 기본 문법을 정의합니다. 즉, XML schema 는 UM3 프로토콜의 XML 구조를 정의합니다.

```
<?xml version=”1.0” encoding=”ISO-8859-1” ?>
<xs:schema xmlns:xs=”http://www.w3c.org/2001/XMLSchema”/>


<!-- definition of UM3 primitive type elements -->


<xs:simpleType name=”UM3Integer16”>
    <xs:restriction base=”xs:short”/>
</xs:simpleType>


<xs:simpleType name=”UM3Integer32”>
    <xs:restriction base=”xs:int”/>
</xs:simpleType>


<xs:simpleType name=”UM3Integer64”>
    <xs:restriction base=”xs:long”/>
</xs:simpleType>
```

```
<xs:simpleType name="UM3UnsignedInteger16">
    <xs:restriction base="xs:unsignedShort"/>
</xs:simpleType>

<xs:simpleType name="UM3UnsignedInteger32">
    <xs:restriction base="xs:unsignedInt"/>
</xs:simpleType>

<xs:simpleType name="UM3UnsignedInteger64">
    <xs:restriction base="xs:unsignedLong"/>
</xs:simpleType>

<xs:simpleType name="UM3CharacterString">
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="UM3BitString">
    <xs:restriction base="xs:base64Binary"/>
</xs:simpleType>

<xs:simpleType name="UM3OctetString">
    <xs:restriction base="xs:hexBinary"/>
</xs:simpleType>

<xs:simpleType name="UM3Boolean">
    <xs:restriction base="xs:booean"/>
</xs:simpleType>

<xs:simpleType name="UM3Real">
    <xs:restriction base="xs:decimal"/>
</xs:simpleType>

<xs:simpleType name="UM3Null">
    <xs:restriction base="xs:string fixed="null"/>
</xs:simpleType>

<xs:simpleType name="UM3DateTime">
```

```
    <xs:restriction base=”xs:decimal”/>
</xs:simpleType>


<xs:simpleType name=”UM3ClassIdentifier”>
    <xs:restriction base=”xs:positiveInteger”/>
</xs:simpleType>


<xs:simpleType name=”UM3ObjectName”>
    <xs:restriction base=”xs:string”/>
</xs:simpleType>


<!-- definition of attribute of the elements -->


<xs:attribute name=”name” type=”xs:string” use=”optional” maxOccurs=”1”/>


<!-- UM3 Information model definitions -->


<xs:complexType name=”AccumulatorSensor”>
    <xs:complexContent>
        <xs:extention base=”SensorBase”>
            <xs:sequence>
                <xs:element name=”presentValue” type=”PresentAccumulatorSensorValue”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”AnalogControl”>
    <xs:complexContent>
        <xs:extention base=”ControlBase”>
            <xs:sequence>
                <xs:element name=”presentValue” type=”PresentAnalogControlValue”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”AnalogSensor”>
    <xs:complexContent>
        <xs:extention base=”SensorBase”>
```

```
            <xs:sequence>
                <xs:element name="location" type="UM3CharacterString"/>
                <xs:element name="presentValue" type="PresentAnalogSensorValue"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="BinaryControl">
    <xs:complexContent>
        <xs:extention base="ControlBase">
            <xs:sequence>
                <xs:element name="presentValue" type="PresentBinaryControlValue"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="BinarySensor">
    <xs:complexContent>
        <xs:extention base="SensorBase">
            <xs:sequence>
                <xs:element name="presentValue" type="PresentBinarySensorValue"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="CompanyInfo">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="name" type="UM3CharacterString"/>
                <xs:element name="phoneNumber" type="UM3CharacterString"/>
                <xs:element name="faxNumber" type="UM3CharacterString" minOccurs="0"/>
                <xs:element name="email" type="UM3CharacterString"/>
                <xs:element name="address" type="UM3CharacterString" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="ControlBase">
    <xs:complexContent>
        <xs:extention base="UM3Base">
```

```
            <xs:sequence>
                <xs:element name="manufacturerInfo" type="CompanyInfo" minOccurs="0"/>
                <xs:element name="vendorInfo" type="CompanyInfo" minOccurs="0"/>
                <xs:element name="maintenancePersonel" type="PersonInfo" minOccurs="0"/>
                <xs:element name="operationalCondition" type="DeviceOperationalCondition"
minOccurs="0"/>
                <xs:element name="hasBattery" type="UM3Boolean"/>
                <xs:element name="batteryInfo" type="InstalledBattery" minOccurs="0"/>
                <xs:element name="hasBackupBattery" type="UM3Boolean"/>
                <xs:element name="backupBatteryInfo" type="InstalledBattery" minOccurs="0"/>
                <xs:element name="presentBatteryValueInfo" type="PresentBatteryValue"/>
                <xs:element name="presentBackupBatteryValueInfo" type="PresentBatteryValue"
minOccurs="0"/>
                <xs:element name="installedEnvironment" type="InstalledEnvironment" minOccurs="0"/>
                <xs:element name="powerConsumption" type="UM3Real"/>
                <xs:element name="levelInAConfigurationTree" type="UM3UnsignedInteger16"/>
                <xs:element name="upperGatewayAddress" type="UM3TcpIpAddress" minOccurs="0"/>
                <xs:element name="operationalTimeInfo" type="DeviceMaintenanceScheduleInfo"/>
                <xs:element name="presentControlStatus" type="DeviceOperationStatus"/>
                <xs:element name="serialNumber" type="UM3CharacterString"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="DeviceMaintenanceScheduleInfo">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="inServiceDateTime" type="UM3DateTime"/>
                <xs:element name="latestMaintenanceDateTime" type="UM3DateTime" minOccurs="0"/>
                <xs:element name="nextMaintenanceDateTime" type="UM3DateTime" minOccurs="0"/>
                <xs:element name="durablePeriod" type="UM3DateTime"/>
                <xs:element name="maintenancePeriod" type="UM3DateTime" minOccurs="0"/>
                <xs:element name="numberOfRebooting" type="UM3UnsignedInteger16" minOccurs="0"/>
                <xs:element name="shutdownScheduleInfo" type="UM3DateTime" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="DeviceOperationalCondition">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="temperatureMax" type="UM3Real"/>
                <xs:element name="temperatureMin" type="UM3Real"/>
                <xs:element name="ratedVoltage" type="UM3UnsignedInteger32"/>
```

```
            <xs:element name="powerConsumption" type="UM3UnsignedInteger32"/>
            <xs:element name="humidityMax" type="UM3UnsignedInteger16"/>
        </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>


<xs:complexType name="DeviceOperationStatus">
  <xs:complexContent>
    <xs:extention base="UM3Base">
      <xs:sequence>
        <xs:element name="presentStatus">
          <xs:restriction base="xs:string">
            <xs:enumeration value="outOfService"/>
            <xs:enumeration value="inService"/>
            <xs:enumeration value="inMaintenance"/>
            <xs:enumeration value="inFault"/>
            <xs:enumeration value="communicationFailure"/>
            <xs:enumeration value="unknownCause"/>
            <xs:enumeration value="detached"/>
          </xs:restriction>
        </xs:element>
        <xs:element name="statusTimestamp" type="UM3DateTime"/>
        <xs:element name="resumeDateTime" type="UM3DateTime"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>


<xs:complexType name="Gateway">
  <xs:complexContent>
    <xs:extention base="UM3Base">
      <xs:sequence>
        <xs:element name="manufacturerInfo" type="CompanyInfo" minOccurs="0"/>
        <xs:element name="vendorInfo" type="CompanyInfo" minOccurs="0"/>
        <xs:element name="maintenancePersonel" type="PersonInfo" minOccurs="0"/>
        <xs:element name="operationalCondition" type="DeviceOperationalCondition"
minOccurs="0"/>
        <xs:element name="hasBattery" type="UM3Boolean"/>
        <xs:element name="batteryInfo" type="InstalledBattery" minOccurs="0"/>
        <xs:element name="hasBackupBattery" type="UM3Boolean"/>
        <xs:element name="backupBatteryInfo" type="InstalledBattery" minOccurs="0"/>
        <xs:element name="installedEnvironment" type="InstalledEnvironment" minOccurs="0"/>
        <xs:element name="hasCoolingFan" type="UM3Boolean" minOccurs="0"/>
        <xs:element name="cpuInfo" type="InstalledCPU"/>
        <xs:element name="memoryInfo" type="InstalledMemory"/>
        <xs:element name="um3ProtocolVersion" type="UM3CharacterString"/>
        <xs:element name="powerConsumption" type="UM3Real"/>
```

```
            <xs:element name="presentBatteryValueInfo" type="PresentBatteryValue"/>
            <xs:element name="presentBackupBatteryValueInfo" type="PresentBatteryValue
minOccur="0"/>
            <xs:element name="numberOfAnalogInputPort" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfAnalogOutputPort" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfBinaryInputPort" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfBinaryOutputPort" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfRS232Port" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfRS485Port" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfWirelessPort" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfEthernetPort" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfIRPort" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfOpticalPort" type="UM3UnsignedInteger16"/>
            <xs:element name="doesSupportExternalMemoryCard" type="UM3Boolean"/>
            <xs:element name="externalMemoryCardType" type="UM3CharacterString minOccur="0"/>
            <xs:element name="analogInputPortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="analogOutputPortDescription" type="UM3CharacterString
minOccur="0"/>
            <xs:element name="binaryInputPortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="binaryOutputPortDescription" type="UM3CharacterString
minOccur="0"/>
            <xs:element name="rs232PortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="rs485PortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="wirelessPortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="ethernetPortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="irPortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="opticalPortDescription" type="UM3CharacterString minOccur="0"/>
            <xs:element name="softwareInfo" type="InstalledSoftware"/>
            <xs:element name="hasHardDrive" type="UM3Boolean"/>
            <xs:element name="hardDriveInfo" type="InstalledHardDrive minOccur="0"/>
            <xs:element name="isDedicatedForOneNode" type="UM3Boolean minOccur="0"/>
            <xs:element name="levelInAConfigurationTree" type="UM3UnsignedInteger16"/>
            <xs:element name="numberOfLowerGateway" type="UM3UngisnedInteger32"/>
            <xs:element name="numberOfLowerNode" type="UM3UnsignedInteger32"/>
            <xs:element name="upperNodeAddress" type="UM3TcpIpAddress"/>
            <xs:element name="address" type="UM3TcpIpAddress"/>
            <xs:element name="doesSupportTelnet" type="UM3Boolean"/>
            <xs:element name="doesSupportFtp" type="UM3Boolean"/>
            <xs:element name="analogSensorList" type="AnalogSensorList"/>
            <xs:element name="binarySensorList" type=" BinarySensorList"/>
            <xs:element name="accumulatedAnalogSensorList" type="AccumulatedAnalogSensorList"/>
            <xs:element name="analogControlList" type="AnalogControlList"/>
            <xs:element name="binaryControlList" type="BinaryControlList"/>
            <xs:element name="multiStateSensorList" type="MultiStateSensorList"/>
            <xs:element name="multiStateControlList" type="MultiStateControlList"/>
            <xs:element name="operationalTimeInfo" type="DeviceMaintenanceScheduleInfo"/>
            <xs:element name="presentValue" type="PresentGatewayValue"/>
            <xs:element name="presentGatewayStatus" type="DeviceOperationStatus minOccur="0"/>
            <xs:element name="maxAPDU" type="UM3UnsignedInteger32"/>
            <xs:element name="sereialNumber" type="UM3CharacterString"/>
        </xs:sequence>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="AnalogSensorList">
    <xs:sequence>
        <xs:element name="analogSensorList" type="AnalogSensor" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="BinarySensorList">
    <xs:sequence>
        <xs:element name="binarySensorList" type="BinarySensor" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="AccumulatedAnalogSensorList">
    <xs:sequence>
        <xs:element name="accumulatedAnalogSensorList" type="AccumulatorSensor"
maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="AnalogControlList">
    <xs:sequence>
        <xs:element name="analogControlList" type="AnalogControl" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="BinaryControlList">
    <xs:sequence>
        <xs:element name="binaryControlList" type="BinaryControl" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="MultiStateSensorList">
    <xs:sequence>
        <xs:element name="multiStateSensorList" type="MultiStateSensor" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name=”MultiStateControlList”>
    <xs:sequence>
        <xs:element name=”multiStateControlList” type=”MultiStateControl” maxOccurs=”unbounded”/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name=”InstalledBattery”>
    <xs:complexContent>
        <xs:extention base=”UM3Base”>
            <xs:sequence>
                <xs:element name=”temperatureMax” type=”UM3Real” minOccurs=”0”/>
                <xs:element name=”temperatureMin” type=”UM3Real” minOccurs=”0”/>
                <xs:element name=”isBackupBattery” type=”UM3Boolean” minOccurs=”0”/>
                <xs:element name=”isChargable” type=”UM3Boolean” minOccurs=”0”/>
                <xs:element name=”ratedVoltage” type=”UM3UnsignedInteger32”/>
                <xs:element name=”ratedCurrent” type=”UM3Real”/>
                <xs:element name=”type” type=”UM3CharacterString” minOccurs=”0”/>
                <xs:element name=”size” type=”UM3CharacterString”/>
                <xs:element name=”weight” type=”UM3Real” minOccurs=”0”/>
                <xs:element name=”presentVoltage” type=”UM3Real”/>
                <xs:element name=”remainingBatteryTime” type=”UM3DateTime” minOccurs=”0”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”InstalledCPU”>
    <xs:complexContent>
        <xs:extention base=”UM3Base”>
            <xs:sequence>
                <xs:element name=”numberOfCPU” type=”UM3UnsignedInteger16”/>
                <xs:element name=”manufacturer” type=”UM3CharacterString”/>
                <xs:element name=”model” type=”UM3CharacterString”/>
                <xs:element name=”speed” type=”UM3CharacterString”/>
                <xs:element name=”busWidth” type=”UM3UnsignedInteger16”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”InstalledEnvironment”>
    <xs:complexContent>
        <xs:extention base=”UM3Base”>
            <xs:sequence>
                <xs:element name=”isIndoor” type=”UM3Boolean”/>
                <xs:element name=”hasEnclosure” type=”UM3Boolean”/>
                <xs:element name=”isWaterproof” type=”UM3Boolean”/>
```

```
                <xs:element name="altitude" type="UM3Integer16" minOccurs="0"/>
                <xs:element name="isRackMounted" type="UM3Boolean" minOccurs="0"/>
                <xs:element name="size" type="UM3CharacterString" minOccurs="0"/>
                <xs:element name="pointDescription" type="UM3CharacterString" minOccurs="0"/>
                <xs:element name="installedDate" type="UM3DateTime"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="InstalledHardDrive">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="diskSize" type="UM3UnsignedInteger32"/>
                <xs:element name="speed" type="UM3CharacterString" minOccurs="0"/>
                <xs:element name="manufacturer" type="UM3CharacterString minOccurs="0"/>
                <xs:element name="diskDiameter" type="UM3Real"/>
                <xs:element name="hasReplaced" type="UM3Boolean"/>
                <xs:element name="replaceDateTime" type="UM3DateTime"/>
                <xs:element name="serialNumber" type="UM3CharacterString"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="InstalledMemory">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="installedMemorySize" type="UM3UnsignedInteger32"/>
                <xs:element name="maximumExpandableSize" type="UM3UnsignedInteger32"/>
                <xs:element name="accessTime" type="UM3Real" minOccurs="0"/>
                <xs:element name="manufacturer" type="UM3CharacterString" minOccurs="0"/>
                <xs:element name="numberOfMemorySlot" type="UM3UnsignedInteger16" minOccurs="0"/>
                <xs:element name="currentMemorySizePerSlot" type="UM3UnsignedInteger32"
minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="InstalledSoftware">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="isOS" type="UM3Boolean"/>
```

```
                <xs:element name=”nameOfSoftware” type=”UM3CharacterString”/>
                <xs:element name=”version” type=”UM3CharacterString”/>
                <xs:element name=”updateDateTime” type=”UM3DateTime”/>
                <xs:element name=”numberOfUpdateUpToNow” type=”UM3UnsignedInteger16”
minOccurs=”0”/>
                <xs:element name=”manufacturer” type=”UM3CharacterString” minOccurs=”0”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”MultiStateControl”>
    <xs:complexContent>
        <xs:extention base=”ControlBase”>
            <xs:sequence>
                <xs:element name=”presentValue” type=”PresentMultiStateControlValue”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”MultiStateSensor”>
    <xs:complexContent>
        <xs:extention base=”ControlBase”>
            <xs:sequence>
                <xs:element name=”presentValue” type=”PresentMultiStateSensorValue”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”PersonInfo”>
    <xs:complexContent>
        <xs:extention base=”UM3Base”>
            <xs:sequence>
                <xs:element name=”name” type=”UM3CharacterString”/>
                <xs:element name=”phoneNumber” type=”UM3CharacterString” minOccurs=”0”/>
                <xs:element name=”cellPhoneNumber” type=”UM3CharacterString”/>
                <xs:element name=”email” type=”UM3CharacterString”/>
                <xs:element name=”address” type=”UM3CharacterString” minOccurs=”0”/>
                <xs:element name=”companyName” type=”UM3CharacterString” minOccurs=”0”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

```
<xs:complexType name="PresentAccumulatorSensorValue">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="presentValue" type="UM3Real"/>
                <xs:element name="presentValueTimestamp" type="UM3DateTime"/>
                <xs:element name="previousValue" type="UM3Real"/>
                <xs:element name="previousValueTimestamp" type="UM3DateTime"/>
                <xs:element name="minPresentValue" type="UM3Real"/>
                <xs:element name="minPresentValueTimestamp" type="UM3DateTime"/>
                <xs:element name="units" type="UM3ChracterString"/>
                <xs:element name="updatePeriod" type="UM3DateTime"/>
                <xs:element name="acceptableMaxPresentValue" type="UM3Real"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="PresentAnalogControlValue">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="presentValue" type="UM3Real"/>
                <xs:element name="presentValueTimestamp" type="UM3DateTime"/>
                <xs:element name="previousValue" type="UM3Real"/>
                <xs:element name="previousValueTimestamp" type="UM3DateTime"/>
                <xs:element name="maxPresentValue" type="UM3Real"/>
                <xs:element name="minPresentValue" type="UM3Real"/>
                <xs:element name="resolution" type="UM3Real"/>
                <xs:element name="units" type="UM3CharacterString"/>
                <xs:element name="acceptableMaxPresentValue" type="UM3Real"/>
                <xs:element name="acceptableMinPresentValue" type="UM3Real"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="PresentAnalogSensorValue">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="presentValue" type="UM3Real"/>
                <xs:element name="presentValueTimestamp" type="UM3DateTime"/>
                <xs:element name="previousValue" type="UM3Real"/>
                <xs:element name="previousValueTimestamp" type="UM3DateTime"/>
                <xs:element name="maxPresentValue" type="UM3Real"/>
                <xs:element name="minPresentValue" type="UM3Real"/>
```

```
            <xs:element name=”resolution” type=”UM3Real”/>
            <xs:element name=”units” type=”UM3CharacterString”/>
            <xs:element name=”updatePeriod” type=”UM3DateTime”/>
            <xs:element name=”acceptableMaxPresentValue” type=”UM3Real”/>
            <xs:element name=”acceptableMinPresentValue” type=”UM3Real”/>
        </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>


<xs:complexType name=”PresentBatteryValue”>
  <xs:complexContent>
    <xs:extention base=”UM3Base”>
      <xs:sequence>
        <xs:element name=”presentVoltage” type=”UM3Real”/>
        <xs:element name=”previousVoltage” type=”UM3Real”/>
        <xs:element name=”remainingBatteryTime” type=”UM3DateTime”/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>


<xs:complexType name=”PresentBinaryControlValue”>
  <xs:complexContent>
    <xs:extention base=”UM3Base”>
      <xs:sequence>
        <xs:element name=”presentValue” type=”UM3Boolean”/>
        <xs:element name=”presentValueTimestamp” type=”UM3DateTime”/>
        <xs:element name=”previousValue” type=”UM3Boolean”/>
        <xs:element name=”previousValueTimestamp” type=”UM3DateTime”/>
        <xs:element name=”stateChangedCount” type=”UM3UnsignedInteger16”/>
        <xs:element name=”stateCountStartedDateTime” type=”UM3DateTime”/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>


<xs:complexType name=”PresentBinarySensorValue”>
  <xs:complexContent>
    <xs:extention base=”UM3Base”>
      <xs:sequence>
        <xs:element name=”presentValue” type=”UM3Boolean”/>
        <xs:element name=”presentValueTimestamp” type=”UM3DateTime”/>
        <xs:element name=”previousValue” type=”UM3Boolean”/>
        <xs:element name=”previousValueTimestamp” type=”UM3DateTime”/>
        <xs:element name=”updatePeriod” type=”UM3DateTime”/>
        <xs:element name=”stateChangedCount” type=”UM3UnsignedInteger16”/>
```

```
                <xs:element name="stateCountStartedDateTime" type="UM3DateTime"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="PresentGatewayValue">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="kbytesMemoryInUse" type="UM3Real"/>
                <xs:element name="kbytesMemoryInUseTimestamp" type="UM3DateTime"/>
                <xs:element name="mbytesHarddiskInUse" type="UM3Real" minOccurs="0"/>
                <xs:element name="mbytesHarddiskInUseTimestamp" type="UM3DateTime"
minOccurs="0"/>
                <xs:element name="percentCpuTime" type="UM3Real"/>
                <xs:element name="percentCpuTimeTimestamp" type="UM3DateTime"/>
                <xs:element name="bytesSentPerSecond" type="UM3UnsignedInteger32"/>
                <xs:element name="bytesReceivedPerSecond" type="UM3UnsignedInteger32"/>
                <xs:element name="bytesPerSecondTimestamp" type="UM3DateTime"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="PresentMultiStateControlValue">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="presentValue" type="UM3UnsignedInteger16"/>
                <xs:element name="presentValueTimestamp" type="UM3DateTime"/>
                <xs:element name="previousValue" type="UM3UnsignedInteger16"/>
                <xs:element name="previousValueTimestamp" type="UM3DateTime"/>
                <xs:element name="numberOfStates" type="UM3UnsignedInteger16">
                <xs:element name="stateNames" type="MultiStateNameList"/>
                <xs:element name="updatePeriod" type="UM3DateTime"/>
                <xs:element name="stateChangedCount" type="UM3UnsignedInteger16"/>
                <xs:element name="stateCountStartedDateTime" type="UM3DateTime"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="PresentMultiStateSensorValue">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
```

```
                <xs:element name="presentValue" type="UM3UnsignedInteger16"/>
                <xs:element name="presentValueTimestamp" type="UM3DateTime"/>
                <xs:element name="previousValue" type="UM3UnsignedInteger16"/>
                <xs:element name="previousValueTimestamp" type="UM3DateTime"/>
                <xs:element name="numberOfStates" type="UM3UnsignedInteger16">
                <xs:element name="stateNames" type="MultiStateNameList"/>
                <xs:element name="updatePeriod" type="UM3DateTime"/>
                <xs:element name="stateChangedCount" type="UM3UnsignedInteger16"/>
                <xs:element name="stateCountStartedDateTime" type="UM3DateTime"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="MultiStateNameList">
    <xs:sequence>
        <xs:element name="multiStateNameList" type="UM3CharacterString" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="SensorBase">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="manufacturerInfo" type="CompanyInfo" minOccurs="0"/>
                <xs:element name="vendorInfo" type="CompanyInfo" minOccurs="0"/>
                <xs:element name="maintenancePersonel" type="PersonInfo" minOccurs="0"/>
                <xs:element name="operationalCondition" type="DeviceOperationalCondition"
minOccurs="0"/>
                <xs:element name="hasBattery" type="UM3Boolean"/>
                <xs:element name="batteryInfo" type="InstalledBattery" minOccurs="0"/>
                <xs:element name="hasBackupBattery" type="UM3Boolean"/>
                <xs:element name="backupBatteryInfo" type="InstalledBattery" minOccurs="0"/>
                <xs:element name="presentBatteryValueInfo" type="PresentBatteryValue"/>
                <xs:element name="presentBackupBatteryValueInfo" type="PresentBatteryValue"
minOccurs="0"/>
                <xs:element name="installedEnvironment" type="InstalledEnvironment" minOccurs="0"/>
                <xs:element name="powerConsumption" type="UM3Real"/>
                <xs:element name="levelInAConfigurationTree" type="UM3UnsignedInteger16"/>
                <xs:element name="upperGatewayAddress" type="UM3TcpIpAddress" minOccurs="0"/>
                <xs:element name="operationalTimeInfo" type="DeviceMaintenanceScheduleInfo"/>
                <xs:element name="presentSensorStatus" type="DeviceOperationStatus"/>
                <xs:element name="serialNumber" type="UM3CharacterString"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

```
<xs:complexType name="SensorMaintenanceScheduleInfo">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="inServiceDateTime" type="UM3DateTime"/>
                <xs:element name="latestMaintenanceDateTime" type="UM3DateTime" minOccurs="0"/>
                <xs:element name="nextMaintenanceDateTime" type="UM3DateTime" minOccurs="0"/>
                <xs:element name="durablePeriod" type="UM3DateTime"/>
                <xs:element name="maintenancePeriod" type="UM3DateTime" minOccurs="0"/>
                <xs:element name="numberOfRewired" type="UM3UnsignedInteger16" minOccurs="0"/>
                <xs:element name="detachScheduleInfo" type="UM3DateTime" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="UM3ActionBroker">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="targetObjectClassIdentifier" type="UM3ClassIdentifier"/>
                <xs:element name="targetObjectObjectName" type="UM3ObjectName"/>
                <xs:element name="targetAttributeClassIdentifier" type="UM3ClassIdentifier"/>
                <xs:element name="targetAttributeObjectName" type="UM3ObjectName"/>
                <xs:element name="targetPeriod" type="UM3DateTime"/>
                <xs:element name="isOnlyOnceAction" type="UM3Boolean"/>
                <xs:element name="reservedActionTimestamp" type="UM3DateTime"/>
                <xs:element name="receipientAddress" type="ReceipientAddressList"/>
                <xs:element name="targetCondition" type="UM3ObjectValueCondition"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="ReceipientAddressList">
    <xs:sequence>
        <xs:element name="receipientAddressList" type="UM3CharacterString" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:simpleType name="UM3ActionInvoker">
    <xs:restriction base="xs:UM3UnsignedInteger16"/>
</xs:simpleType>
```

```
<xs:complexType name="UM3Base">
    <xs:sequence>
        <xs:element name="um3ClassIdentifier" type="UM3ClassIdentifier"/>
        <xs:element name="um3ObjectName" type="UM3ObjectName"/>
        <xs:element name="um3ObjectNameAlias" type="UM3CharacterString" minOccurs="0"/>
        <xs:element name="um3ObjectDescription" type="xs:string" minOccurs="0"/>
        <xs:element name="um3AttributeList" type="UM3ObjectList"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="UM3EventReport">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="targetObjectClassIdentifier" type="UM3ClassIdentifier"/>
                <xs:element name="targetObjectObjectName" type="UM3ObjectName"/>
                <xs:element name="targetAttributeClassIdentifier" type="UM3ClassIdentifier"/>
                <xs:element name="targetAttributeObjectName" type="UM3ObjectName"/>
                <xs:any name="targetAttributeValue"/>
                <xs:element name="eventTimestamp" type="UM3DateTime"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="UM3ObjectIndicator">
    <xs:sequence>
        <xs:element name="um3ClassIdentifierIndicator" type="UM3ClassIdentifier"/>
        <xs:element name="um3ObjectNameIndicator" type="UM3ObjectName"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="UM3ObjectList">
    <xs:sequence>
        <xs:any maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="UM3ObjectListToBeCreatedList">
    <xs:sequence>
        <xs:element name="dn" type="UM3ObjectName"/>
        <xs:any maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name=”UM3ObjectListToBeCreated”>
    <xs:sequence>
        <xs:element name=”um3ObjectListToBeCreatedList” type=”UM3ObjectListToBeCreatedList”
maxOccurs=”unbounded”/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name=”UM3ObjectValueCondtion”>
    <xs:complexContent>
        <xs:extention base=”UM3Base”>
            <xs:sequence>
                <xs:element name=”upperEnd” type=”EndLimitType”/>
                <xs:element name=”lowerEnd” type=”EndLimitType”/>
                <xs:element name=”condition”>
                    <xs:restriction base=”xs:string”>
                        <xs:enumeration value=”LL”/>
                        <xs:enumeration value=”GLLU”/>
                        <xs:enumeration value=”GU”/>
                        <xs:enumeration value=”LEL”/>
                        <xs:enumeration value=”GELaLEU”/>
                        <xs:enumeration value=”GEU”/>
                        <xs:enumeration value=”GELaLU”/>
                        <xs:enumeration value=”GLaLEU”/>
                        <xs:enumeration value=”EU”/>
                        <xs:enumeration value=”EL”/>
                        <xs:enumeration value=”LLoGU”/>
                        <xs:enumeration value=”LELoGU”/>
                        <xs:enumeration value=”LELoGEU”/>
                        <xs:enumeration value=”LLoGEU”/>
                        <xs:enumeration value=”COV”/>
                    </xs:restriction>
                </xs:element>
                <xs:element name=”targetAttributeClassIdentifier” type=”UM3ClassIdentifier”/>
                <xs:element name=”targeetAttributeObjectName” type=”UM3ObjectName”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name=”EndLimitType”>
    <xs:restriction base=”xs:string”>
        <xs:choice>
            <xs:element ref=”UM3Integer16”/>
            <xs:element ref=”UM3Integer32”/>
            <xs:element ref=”UM3Integer64”/>
            <xs:element ref=”UM3UnsignedInteger16”/>
```

```
            <xs:element ref="UM3UnsignedInteger32"/>
            <xs:element ref="UM3UnsignedInteger64"/>
            <xs:element ref="UM3Real"/>
            <xs:element ref="UM3CharacterString"/>
            <xs:element ref="UM3Boolean"/>
            <xs:element ref="UM3DateTime"/>
        </xs:choice>
    </xs:restriction>
</xs:complexType>


<xs:complexType name="UM3OperationErrorCode">
    <xs:restriction base="xs:string">
        <xs:enumeration value="notFound"/>
        <xs:enumeration value="accessDenied"/>
        <xs:enumeration value="noSuchClass"/>
        <xs:enumeration value="noSuchAttribute"/>
        <xs:enumeration value="processingFailure"/>
        <xs:enumeration value="unrecognizedOperation"/>
        <xs:enumeration value="unknownSignature"/>
        <xs:enumeration value="eventDrivenNotSupported"/>
        <xs:enumeration value="noSuchSession"/>
    </xs:restriction>
</xs:complexType>


<xs:complexType name="UM3ProtocolSupportDescription">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="version" type="UM3CharacterString"/>
                <xs:element name="level" type="UM3CharacterString"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="UM3TcpIpAddress">
    <xs:complexContent>
        <xs:extention base="UM3Base">
            <xs:sequence>
                <xs:element name="ipAddressV4" type="UM3CharacterString"/>
                <xs:element name="ipAddressV6" type="UM3CharacterString" minOccurs="0"/>
                <xs:element name="portNumber" type="UM3UnsignedInteger16"/>
                <xs:element name="tcpOrUdp" type="UM3Boolean"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

</xs:schema>

17. JSON type data exchange

본 권고안은 앞서 정의한 바이너리 형식의 데이터 송수신을 위한 UM3 SER 인코딩 방식, 텍스트 형식을 지원하는 XML 형식의 송수신 방식과 더불어 JSON 형식의 데이터 송수신 방식을 정의하고 있습니다.

This recommendation defined the JSON type data exchange method as well as the UM3 SER encoding method for the binary type data exchange and XML type data exchange supporting text type data.

본 권고안이 정의하는 JSON 형식의 데이터 송수신 방식은 앞서 정의한 UM3 기본 타입, UM3 복합형식 타입, UM3 클래스 타입의 데이터를 송수신하는 방법에 있어서, 각 타입별 데이터를 UM3 SER 인코딩 룰 혹은 XML 인코딩 룰을 사용하여 송수신하는 대신 JSON 형식의 텍스트 기반 데이터로 송수신하는 방법을 정의합니다.

In the ISON type data exchange method defined in this recommendation, it uses JSON type text based data instead of UM3 SER coding rule or XML encoding rule for data of each type when exchanging data of UM3 primitive type, UM3 complex format type, or UM3 class type.

본 권고안이 앞서 정의한 UM3 primitive type, UM3 complex format type 및 UM3 class type 을 구성하는 모든 데이터 타입과 클래스 타입들을 구성하는 애트리뷰트, 애트리뷰트 혹은 엘리먼트의 나열 순서 등의 규칙은 JSON 기반의 데이터 송수신을 위한 데이터 구성에도 동일하게 적용됩니다.

Attributes that configure all data types and class types for UM3 primitive type, UM3 complex format type, and UM3 class type defined earlier and rules like listing sequence of attribute of element are applied to the data configuration for JSON based data exchange in the same way.

17.1. JSON encoding of UM3 Primitive type class

UM3 JSON 인코딩 룰은 앞서 정의한 UM3 SER 및 UM3 XML 인코딩 룰과 동일한 개념으로 정의됩니다. 즉, 송수신 되는 APDU 는 비정형데이터를 처리하기 위해 self descriptive 한 특성을 갖고 있으며 특히 UM3 프로토콜을 사용하는 M2M 관련 서비스 시스템의 확장가능성에 초점을 두고 있습니다.

The UM3 JSON encoding rule is defined in the same concept as the UM3 SER and UM3 XML encoding rules defined earlier. That is, transmitted and received APDU has the self-descriptive characteristic to process atypical data, and it is especially focused on the scalability of the M2M related service system by using the UM3 protocol

아래의 예는 UM3Integer32 타입의 클래스 오브젝트에 대한 UM3 JSON 인코딩 결과입니다.

“um3ClassIdentifier”와 “um3ObjectName” 및 “value” 로 표현되는 애트리뷰트의 이름들이 UM3 SER 인코딩 룰 및 UM3 XML 인코딩 룰과 동일한 개념으로 사용됩니다.

The following example is the result of UM3 JSON encoding for the class object of UM3Integer32 type. Names of attributes represented as “um3ClassIdentifier” and “um3ObjectName” and “value” are used as the same concept as the UM3 SER encoding rule and UM3 XML encoding rule.

```
{
    “um3ClassIdentifier” : 31,
    “um3ObjectName” : “MyInteger”,
    “value” : 17
}
```

위와 마찬가지로 아래는 UM3CharacterString 타입의 UM3 primitive 타입의 UM3 JSON 인코딩 예입니다.

Like the above example, the example shown below is UM3 JSON encoding of UM3 primitive type of UM3CharacterString type.

```
{
    “um3ClassIdentifier” : 36,
    “um3ObjectName” : “personName”,
    “um3ObjectNameAlias” : “name”,
    “um3ObjectDescription” : “The name of the person who is responsible for the device. Office and cellphone number should be described in the other object definition.”,
    “value” : “Person Name”
}
```

앞서 정의한 UM3 XML 인코딩 룰과 마찬가지로 UM3 JSON 인코딩 룰 또한 T-N-V 형식을 취하고 있습니다. UM3 primitive 타입을 구성하고 있는 다른 오브젝트 클래스들 또한 동일한 방식으로 인코딩 과정을 수행하게 됩니다.

Like the UM3 XML encoding rule defined earlier, the UM3 JSON encoding rule also has the T-N-V format. The same encoding procedure is applied to other object classes configuring the UM3 primitive type.

17.2. UM3 Complex type class의 JSON encoding

아래는 본 권고안이 정의하고 있는 UM3 complex type 오브젝트 클래스들 중 PersonInfo 오브젝트 클래스의 인코딩 예입니다.

The following is an example of encoding for the PersonInfo object class, one of UM3 complex type object classes defined in this recommendation.

```
{
    “um3ClassIdentifier” : 120,
    “value” :
    [
        {
            “um3ClassIdentifier” : 31,
            “um3ObjectName” : “name”,
            “value” : “John Anderson”,
        }
        {
            “um3ClassIdentifier” : 31,
            “um3ObjectName” : “phoneNumber”,
            “value” : “999-7765-9987”
        }
        {
            “um3ClassIdentifier” : 31,
            “um3ObjectName” : “cellPhoneNumber”,
            “value” : “999-7765-9986”
        }
        {
            “um3ClassIdentifier” : 31,
            “um3ObjectName” : “email”,
            “value” : “person.name@company.com”,
        }
        {
            “um3ClassIdentifier” : 31,
            “um3ObjectName” : “address”,
            “value” : “17 Eastern st. NY. NY.”
        }
        {
            “um3ClassIdentifier” : 31,
            “um3ObjectName” : “companyName”,
            “value” : “The Company Corporation”
        }
    ]
}
```

위의 예에서 보는 바와 같이 ‘um3ObjectName’ 애트리뷰트는 경우에 따라 생략될 수 있으나 ‘um3ClassIdentifier’ 애트리뷰트는 생략될 수 없습니다. 또한 일반적인 JSON 표현방식과 같이 복합형식으로 구성된 값에 대해서는 ‘[‘와 ‘]’ 를 이용해 구분합니다.

As shown in this example, the ‘um3ObjectName’ attribute can be skipped when necessary, but the

‘um3ClassIdentifier’ attribute is mandatory. The values configured as complex format can be identified with ‘[‘ and ‘]’ like the general JSON representation method.

부록 1. UM3 프로토콜의 구현 (프로그래밍 예)

부록 1 에 예시된 프로그램은 본 권고안이 정의하고 있는 UM3 primitive type 과 몇 개의 UM3 오퍼레이션에 대한 UM3 SER 의 예를 보여주기 위한 예제프로그램의 일부 입니다. 해당 프로그램이 사용하는 클래스아이디 및 오퍼레이션의 signature 는 본 권고안의 정의와 차이가 있을 수 있으며 이는 구현 예를 보여주기 위한 목적임을 밝혀 둡니다.

Encoder.c

```
/*
   Encoder.c

   (c)2009 KT Corporation

   WunBae Jeon (ubjeon@kt.com)
*/
/*
   README +

   Encoder.c is the source file for 'libum3.a' static library.
   If you would like to use the dynamic linking library of the
   encoding functions, then you should make those dynamic libraries
   youself.

   README ++

   Encoder.c collects the functions which encodes and decodes the following
   UM3 primitive types;

      UM3Integer16,
      UM3Integer32,
      UM3Integer64,
      UM3UnsignedInteger16,
      UM3UNsignedInteger32,
      UM3UnsignedInteger64,
      UM3CharacterString,
      UM3BitString,
      UM3OctetString,
      UM3Boolean,
      UM3Enumerated,
      UM3Real,
      UM3Null,
      UM3DateTime,
      UM3ClassIentifier,
      UM3ObjectName

   The utility functions are as follows;

      GetUM3Integer16ApduLength(),
      GetUM3Integer32ApduLength(),
      GetUM3Integer64ApduLength(),
      GetUM3UInteger16ApduLength(),
      GetUM3UInteger32ApduLength(),
      GetUM3UInteger64ApduLength(),
      GetUM3EnumeratedApduLength(),
      GetUM3BooleanApduLength(),
      GetUM3RealApduLength(),
      GetUM3CharStrApduLength(),
      GetUM3BitStrApduLength(),
      GetUM3OctetStrApduLength(),
GetUM3RealApduLength(),
      GetUM3CharStrApduLength(),
```

```
     GetUM3BitStrApduLength(),
     GetUM3OctetStrApduLength(),
     GetUM3ClassIdentifierApduLength(),
     GetUM3ObjectNameApduLength(),
     GetUM3NullApduLength(),
     GetUM3DateTimeApduLength(),

     and

     PrintOutUM3Adpu(),

     and

     _getUM3IntegerApduLength(char* parpaszObjectName,
     _printOutUM3Apdu(),
     _um3ByteSwap(),
     _um3GetObjectNameLength()

   The function starts with '_xxxx' is the function which should not be
   directly called from the users. Those are the functions that are used
   by only the functions that start with the capital letters.

   You can refer to the following recommendation to clearly
   understand the primitive types of UM3 protocol.

     UM3 Protocol Recommendation (2009.11)

   There's currently no corrigendum or amendment recommendation to the above
   2009.11 version on 10th November, 2009.

   README +++

   We do not provide the manual to the above encoding library routines.
   It is recommended that you refer to the comments at the
   beginning of each of the functions.

   README ++++

   The functions that start and end with 'GetUM3....ApduLength()' should
   be called prior to the dynamic allocation or definition of the
   char array in your program.

   README +++++

   If you are to construct the software system which will be run at the
   server side, not at the gateway or RTU, then it is highly recommended
   to construct your system with the object-oriented programming languages.

*/
#include <Encoder.h>

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>

// EncodeUM3Integer16()
//
// descriptions:
//   Encode the given UM3Integer16 type object value to the UM3
//   APDU with UM3 SER (Simple Encoding Rule) by calling the
//   _encodeUM3Integer() function.
//   You should check the length of the APDU via calling of
//   GetUM3Integer16ApduLength() function before calling this function.
//
// parameters:
```

```
//   char* parpachApdu
//      Dynamic allocated char array which has the length that has been
//      checked with GetUM3Integer16ApduLength() function.
//   int pardApduLen
//      The length of the UM3 APDU or equivalently, the length of the
//      given parpachApdu parameter.
//   char* parpaszObjectName
//      The name of the object i.e. the name of the integer value which
//      is given via pardValue parameter.
//   short int pardValue
//      The object value to be encoded.
//
// returns:
//   int 1
//      success
//   int -1
//      The given parpachApdu has not been allocated. It has a null pointer.
//
int
EncodeUM3Integer16(char*    parpachApdu,
                   int      pardApduLen,
                   char*    parpaszObjectName,
                   short int pardValue)
{
  if (parpachApdu == NULL) {
    return (-1);
  }

  _encodeUM3Integer(parpachApdu,
                    pardApduLen,
                    parpaszObjectName,
                    &pardValue,
                    UM3_INTEGER16_CID,
                    SIZEOF_UM3_INTEGER16);

  return (1);
}

// GetUM3Integer16ApduLength()
//
// descriptions:
//   Returns the whole length of the UM3Integer16 type object value.
//
// parameters:
//   char* parpaszObjectName
//      The name of the object. Usually the length of the UM3Integer16 type
//      is as follows;
//
//        length = length of object name + (4 + 2 + 2 + 2)
//
// returns:
//   int
//      length of the UM3 APDU which has the type of UM3Integer32
//
int
GetUM3Integer16ApduLength(char* parpaszObjectName)
{
  int dTemp = _getUM3IntegerApduLength(parpaszObjectName,
                                       UM3_INTEGER16_CID);
  return (dTemp);
}

// EncodeUM3Integer32()
//
// descriptions:
//   Encode the given UM3Integer32 type object value to the UM3
```

```
//   APDU with UM3 SER (Simple Encoding Rule) by calling of
//   _encodeUM3Integer() function.
//   You should check the length of the APDU via calling of the
//   GetUM3Integer32ApduLength() function before calling this function.
//
// parameters:
//   char* parpachApdu
//      Dynamic allocated char array which has the length that has been
//      checked with GetUM3Integer32ApduLength() function.
//   int pardApduLen
//      The length of the UM3 APDU or equivalently, the length of the
//      given parpachApdu parameter.
//   char* parpaszObjectName
//      The name of the object i.e. the name of the integer value which
//      is given via the pardValue parameter.
//   int pardValue
//      The object value to be encoded.
//
// returns:
//   int 1
//      success
//   int -1
//      The given parpachApdu has not been allocated. It has a null pointer.
//
int
EncodeUM3Integer32(char* parpachApdu,
                int   pardApduLen,
                char* parpaszObjectName,
                int   pardValue)
{
  if (parpachApdu == NULL) {
    return (-1);
  }

  _encodeUM3Integer(parpachApdu,
                pardApduLen,
                parpaszObjectName,
                &pardValue,
                UM3_INTEGER32_CID,
                SIZEOF_UM3_INTEGER32);

  return (1);
}
```

Operation.c

```
/*
  Operation.c

  (c)2009 KT Corporation

  WunBae Jeon (ubjeon@kt.com)
*/


#include <Operation.h>
#include <Encoder.h>
#include <OperationEncoder.h>

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
// GetObjectAttributeValue()
//
// descriptions:
//   Controls the execution flow of encoding of sending packet,
//   transmitting of those packets, and receiving and decoding of
//   OperationResponse operation.
//
// parameters:
//   int sd
//      Connected socket descriptor which will be used as a connection
//      to the agent.
//   char* parpaszUM3OperationObjectName
//      The sessiion identifier which will be used as a unique identi-
//      fier to the UM3 sessions.
//   UM3ClassIdentifier parpaszObjectClassIdentifier
//      The UM3ClassIdentifier type value which discriminates the
//      object class from the other classes.
//   UM3ObjectName parpaszObjectObjectName
//      The UM3ObjectName type value which represents the RDN of the
//      attribute to be fetched its value.
//   UM3ClassIdentifier parpaszAttributeClassIdentifier
//      The class identifier value of the attribute to be fetched.
//   UM3ObjectName parpaszAttributeObjectName
//      The object name of the attribute to be fetched.
//
// returns:
//   int
//      dummy
//
// huhu
int
GetObjectAttributeValue(int             sd,
                    char*           parpaszUM3OperationObjectName,
                    UM3ClassIdentifier parpaszObjectClassIdentifier,
                    UM3ObjectName     parpaszObjectObjectName,
                    UM3ClassIdentifier parpaszAttributeClassIdentifier,
                    UM3ObjectName     parpaszAttributeObjectName)
{
  // construct the APDU
  int dApduLen =
    CalGetObjectAttributeValueApduLen(parpaszUM3OperationObjectName,
                                parpaszObjectClassIdentifier,
                                parpaszObjectObjectName,
                                parpaszAttributeClassIdentifier,
                                parpaszAttributeObjectName);


  char* pachApdu = new char [dApduLen];

  int fdResult =
    EncodeGetObjectAttributeValue(pachApdu,
                              dApduLen,
                              parpaszUM3OperationObjectName,
                              parpaszObjectClassIdentifier,
                              parpaszObjectObjectName,
                              parpaszAttributeClassIdentifier,
                              parpaszAttributeObjectName);

  char* pachApduReceive = new char [1024];

  int rlen;
  if ((rlen = write(sd,
                pachApdu,
                dApduLen)) != dApduLen) {
    printf("send error: \n");
```

```
    exit(0);
  }
  else {
    // read, terminates stopwatch
    if ((rlen = read(sd,
                pachApduReceive,
                1024)) < 0) {
      printf("receive error: \n");
      exit(0);
    }

    // decode the received APDU and print out to the stdout
    printf("\nR E C E I V E D .........\n\n");
    PrintOutUM3OperationApdu(pachApduReceive);
  }
  // terminate the operation

  delete [] pachApdu;
  delete [] pachApduReceive;
}


// GetObjectAttributeValue()
//
// descriptions:
//   UM3-GET service category operation, GetObjectAttributeValue() operation
//   is performed.
//
// parameters:
//
// returns:
//
int
GetObjectAttributeValue(int sd)
{
  // construct the APDU
  int dApduLen =
    CalGetObjectAttributeValueApduLen("mama-dodo-####!@!-001",
                                ANALOG_SENSOR_CID,
                                "jrGw.water01",
                                UM3_REAL_CID,
                                "jrGw.water01.presentValue");

  char* pachApdu = new char [dApduLen];

  int fdResult =
    EncodeGetObjectAttributeValue(pachApdu,
                              dApduLen,
                              "mama-dodo-####!@!-001",
                                 ANALOG_SENSOR_CID,
                                 "jrGw.water01",
                                 UM3_REAL_CID,
                                 "jrGw.water01.presentValue");

  char* pachApduReceive = new char [1024];

  int rlen;
  if ((rlen = write(sd,
                pachApdu,
                dApduLen)) != dApduLen) {
    printf("send error: \n");
    exit(0);
  }
  else {
    // read, terminates stopwatch
    if ((rlen = read(sd,
```

```
                pachApduReceive,
                1024)) < 0) {
       printf("receive error: \n");
       exit(0);
     }

     // decode the received APDU and print out to the stdout
     printf("\nR E C E I V E D ..........\n\n");
     PrintOutUM3OperationApdu(pachApduReceive);
   }
   // terminate the operation

   delete [] pachApdu;
   delete [] pachApduReceive;
}

// OperationResponseSuccess()
//
// descriptions:
//   Function for constructing and sending a OperationResponse() operation
//   APDU. The function is called when the result of the UM3 operation
//   that is received from the manager is successful.
//
// parameters:
//   char* parpachApdu
//      Byte array for UM3 APDU
//   int pardApduLen
//      The size of the above UM3 APDU
//   char* parpaszOperationObjectName
//      The session identifier which makes the above UM3 APDU for the rest of
//      the sessions.
//   UM3Boolean parResult
//      The result of the operation, TRUE as a default value
//   UM3ClassIdentifier parAnyValueType,
//      The UM3ClassIdentifier value which has been received from the
//      calling environment.
//   AnyType parAnyValue
//      The ANY DEFINED BY UM3 ASN.1 module type value
//
// returns:
//   int
//      dummy
//
int
OperationResponseSuccess(char*            parpachApdu,
                    int             pardApduLen,
                    char*           parpaszOperationObjectName,
                    UM3Boolean       parResult,
                    UM3ClassIdentifier parAnyValueType,
                    AnyType          parAnyValue)
{
   int dLen =
     CalOperationResponseSuccessApduLen(parpaszOperationObjectName,
                                parResult,
                                parAnyValueType,
                                parAnyValue);

   EncodeOperationResponseSuccess(parpachApdu,
                           dLen,
                           parpaszOperationObjectName,
                           parResult,
                           parAnyValueType,
                           parAnyValue);

   return (1);
}
```

```
// heyhey
// SetObjectAttributeValue()
//
// descriptions:
//   Controls the execution flow of encoding of sending packet,
//   transmitting of those packets, and receiving and decoding of the
//   OperationResponse operation.
//
//   Caller is server-set.
//
// parameters:
//
//
// returns:
//   int
//      dummy
//
int
SetObjectAttributeValue(int             sd,
                    char*           parpaszUM3OperationObjectName,
                    UM3ClassIdentifier parpaszObjectClassIdentifier,
                    UM3ObjectName     parpaszObjectObjectName,
                    UM3ClassIdentifier parpaszAttributeClassIdentifier,
                    UM3ObjectName     parpaszAttributeObjectName,
                    AnyType          parpAnyTypeValue)
{
   // At first, you should make the value to be set at the gateway side
   double dlTemp = 152.7;

   // now construct the APDU
   int dApduLen =
      CalSetObjectAttributeValueApduLen(parpaszUM3OperationObjectName,
                                 parpaszObjectClassIdentifier,
                                 parpaszObjectObjectName,
                                 parpaszAttributeClassIdentifier,
                                 parpaszAttributeObjectName,
                                 parpAnyTypeValue);

   char* pachApdu = new char [dApduLen];

   int fdResult =
      EncodeSetObjectAttributeValue(pachApdu,
                               dApduLen,
                               parpaszUM3OperationObjectName,
                               parpaszObjectClassIdentifier,
                               parpaszObjectObjectName,
                               parpaszAttributeClassIdentifier,
                               parpaszAttributeObjectName,
                               parpAnyTypeValue);

   char* pachApduReceive = new char [1024];

   int rlen;
   if ((rlen = write(sd,
                 pachApdu,
                 dApduLen)) != dApduLen) {
      printf("send error: \n");
      exit(0);
   }
   else {
      // read, terminates stopwatch
      if ((rlen = read(sd,
                    pachApduReceive,
                    1024)) < 0) {
         printf("receive error: \n");
```

```
      exit(0);
    }

    // decode the received APDU and print out to the stdout
    printf("\nR E C E I V E D  ..........\n\n");
    PrintOutUM3OperationApdu(pachApduReceive);
  }
  // terminate the operation

  delete [] pachApdu;
  delete [] pachApduReceive;
}

// SetObjectAttributeValue()
//
// descriptions:
//   UM3-SET service category operation, SetObjectAttributeValue() operation
//   is performed.
//
// parameters:
//
// returns:
//
int
SetObjectAttributeValue(int sd)
{
  // At first, you should make the value to be set at the gateway side
  double dlTemp = 152.7;

  // now construct the APDU
  int dApduLen =
    CalSetObjectAttributeValueApduLen("mama-dodo-####!@!-001",
                                ANALOG_SENSOR_CID,
                                "jrGw.water01",
                                UM3_REAL_CID,
                                "jrGw.water01.presentValue",
                                (void*)&dlTemp);

  char* pachApdu = new char [dApduLen];

  int fdResult =
    EncodeSetObjectAttributeValue(pachApdu,
                              dApduLen,
                              "mama-dodo-####!@!-001",
                                 ANALOG_SENSOR_CID,
                                 "jrGw.water01",
                                 UM3_REAL_CID,
                                 "jrGw.water01.presentValue",
                                 (void*)&dlTemp);

  char* pachApduReceive = new char [1024];

  int rlen;
  if ((rlen = write(sd,
                pachApdu,
                dApduLen)) != dApduLen) {
    printf("send error: \n");
    exit(0);
  }
  else {
    // read, terminates stopwatch
    if ((rlen = read(sd,
                  pachApduReceive,
                  1024)) < 0) {
      printf("receive error: \n");
      exit(0);
```

```
    }

    // decode the received APDU and print out to the stdout
    printf("\nS E N T  ..........\n\n");
    PrintOutUM3OperationApdu(pachApduReceive);
  }
  // terminate the operation

  delete [] pachApdu;
  delete [] pachApduReceive;
}
```

OperationEncoder.c

```
/*
  OperationEncoder.c

  (c)2009 KT Corporation

  WunBae Jeon (ubjeon@kt.com)
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <OperationEncoder.h>
#include <Encoder.h>

// EncodeGetObjectAttributeValue()
//
// descriptions:
//   Constructs the UM3 operation APDU for GetObjectAttributeValue().
//   The T field will be encoded with the class identifier of the
//   EncodeGetObjectAttributeValue() operation class identifier. The NL field
//   will be filled with the unsigned short int type value that represents
//   the length of the operation object name i.e. the session identifier.
//   The N field contains the name of the operation object name, as stated
//   above, the session identifier. The L field will have the encoded value of
//   the size of the V field. The V field contains multiple APDU which
//   represent the parameters of the operation GetObjectAttributeValue().
//
// parameters:
//   char* parpachApdu
//     The char byte array to be filled with the encoded UM3 operation.
//   int pardApduLen
//     The length of the given parpachApdu byte array. It is important
//     that this information does not collapse when running the encoding
//     function is running.
//   UM3ClassIdentifier parUM3ClassIdentifier
//     The class identifier value which discriminates the target
//     object's class type.
//   UM3ObjectName parUM3ObjectName
//     The name of the target object that has the attribute object which
//     we would like to get the value
//   UM3ClassIdentifier parUM3AttributeClassIdentifier
//     The class identifier of the attribute object
//   UM3ObjectName parUM3AttributeObjectName
//     The name of the attribute object which the value to be get
//
// returns:
//
int
EncodeGetObjectAttributeValue(char*           parpachApdu,
```

```
                        int             pardApduLen,
                        char*           parpaszOperationObjectName,
                        UM3ClassIdentifier parUM3ClassIdentifier,
                        UM3ObjectName     parUM3ObjectName,
                        UM3ClassIdentifier
                               parUM3AttributeClassIdentifier,
                        UM3ObjectName
                               parUM3AttributeObjectName)
{
  char* pachPseudo = NULL;
  pachPseudo = parpachApdu;

  memset(pachPseudo, 0x00, pardApduLen);

  // T field
  unsigned int dTValue = GET_OBJ_ATTR_VALUE_CID;

  memcpy(pachPseudo, &dTValue, SIZEOF_UM3_T);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_T);

  pachPseudo += SIZEOF_UM3_T;

  // NL field
  unsigned short int dNLen =
    _um3GetObjectNameLength(parpaszOperationObjectName);
  memcpy(pachPseudo, &dNLen, SIZEOF_UM3_NL);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_NL);

  pachPseudo += SIZEOF_UM3_NL;

  // N field
  memcpy(pachPseudo, parpaszOperationObjectName, dNLen);
  pachPseudo += dNLen;

  // Here, we need to enter into the nested packet generation mode
  // L field

  // get UM3ClassIdentifier type parameter length
  int dTempLenAll;
  int dTempLenOne = GetUM3ClassIdentifierApduLength("parUM3ClassIdentifier");

  // get UM3ObjectName type parameter length
  int dTempLenTwo = GetUM3ObjectNameApduLength("parUM3ObjectName",
                                       parUM3ObjectName);

  // get UM3AttributeClassIdentifier type parameter length
  int dTempLenThree =
    GetUM3ClassIdentifierApduLength("parUM3AttributeClassIdentifier");

  // get UM3AttributeObjectName type parameter length
  int dTempLenFour =
    GetUM3ObjectNameApduLength("parUM3AttributeObjectName",
                           parUM3AttributeObjectName);

  dTempLenAll = dTempLenOne + dTempLenTwo + dTempLenThree + dTempLenFour;

  memcpy(pachPseudo, &dTempLenAll, SIZEOF_UM3_L);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_L);

  pachPseudo += SIZEOF_UM3_L;

  // V field (There are 4 parameters to be encoded.)

  // parUM3ClassIdentifier
  EncodeUM3ClassIdentifier(pachPseudo,
                       dTempLenOne,
```

```
                        "parUM3ClassIdentifier",
                        parUM3ClassIdentifier);
   pachPseudo += dTempLenOne;

   // parUM3ObjectName
   EncodeUM3ObjectName(pachPseudo,
                   dTempLenTwo,
                   "parUM3ObjectName",
                   parUM3ObjectName);
   pachPseudo += dTempLenTwo;

   // parUM3AttributeClassIdentifier
   EncodeUM3ClassIdentifier(pachPseudo,
                        dTempLenThree,
                        "parUM3AttributeClassIdentifier",
                        parUM3AttributeClassIdentifier);
   pachPseudo += dTempLenThree;

   // parUM3AttributeObjectName
   EncodeUM3ObjectName(pachPseudo,
                   dTempLenFour,
                   "parUM3AttributeObjectName",
                   parUM3AttributeObjectName);
   // x;pachPseudo += dTempLenFour;

   return (1);
}

// CalGetObjectAttributeValueApduLen()
//
// descriptions:
//   Calculate the whole length of the GetObjectAttributeValue() UM3
//   operation APDU including the length of 4 parameters.
//
//   length =
//         T(4) + NL(2) + N(?) + L(2) + V(
//                                    T(4) + NL(2) + N(?) + L(2) + V(4)
//                                    T(4) + NL(2) + N(?) + L(2) + V(?)
//                                    T(4) + NL(2) + N(?) + L(2) + V(4)
//                                    T(4) + NL(2) + N(?) + L(2) + V(?)
//                                  )
//
//   In most cases, you can use a fixed length char array. In that case,
//   you must give them enough memory space to the static char array.
//   Otherwise, the segmentation fault will be occur and the execution
//   will be stopped. If you would like to avoid those fatal situation,
//   it is highly recommended that you first calculate the whole length
//   of the operation APDU and allocate memory space dynamically.
//
// parameters:
//   char* parpaszOperationObjectName
//      The parpaszOperationObjectName is the session identifier which has
//      been generated by the requesting side, usually the manager side
//   UM3ClassIdentifier parUM3ClassIdentifier
//      The class identifier which contains the attribute object.
//   UM3ObjectName parUM3ObjectName
//      The object name of the object which has the name of parUM3Object-
//      Name value
//   UM3ClassIdentifier parUM3AttributeClassIdentifier
//      The attribute class identifier
//   UM3ObjectName parUM3AttributeObjectName
//      The attribute object name
//
// returns:
//   int
//      The length of the whole operation APDU.
```

```
//
int
CalGetObjectAttributeValueApduLen(char* parpaszOperationObjectName,
                            UM3ClassIdentifier parUM3ClassIdentifier,
                            UM3ObjectName     parUM3ObjectName,
                            UM3ClassIdentifier
                                parUM3AttributeClassIdentifier,
                            UM3ObjectName
                                parUM3AttributeObjectName)
{
  int dTempLenOne = _um3GetObjectNameLength(parpaszOperationObjectName);

  // get UM3ClassIdentifier type parameter length
  dTempLenOne += GetUM3ClassIdentifierApduLength("parUM3ClassIdentifier");

  // get UM3ObjectName type parameter length
  dTempLenOne += GetUM3ObjectNameApduLength("parUM3ObjectName",
                                    parUM3ObjectName);

  // get UM3AttributeClassIdentifier type parameter length
  dTempLenOne +=
    GetUM3ClassIdentifierApduLength("parUM3AttributeClassIdentifier");

  // get UM3AttributeObjectName type parameter length
  dTempLenOne +=
    GetUM3ObjectNameApduLength("parUM3AttributeObjectName",
                           parUM3AttributeObjectName);

  dTempLenOne += 8;

  return (dTempLenOne);
}

// CalSetObjectAttributeValueApduLen()
//
// descriptions:
//   Calculate the whole length of the GetObjectAttributeValue() UM3
//   operation APDU including the length of 4 parameters.
//
//   length =
//         T(4) + NL(2) + N(?) + L(2) + V(
//                                   T(4) + NL(2) + N(?) + L(2) + V(4)
//                                   T(4) + NL(2) + N(?) + L(2) + V(?)
//                                   T(4) + NL(2) + N(?) + L(2) + V(4)
//                                   T(4) + NL(2) + N(?) + L(2) + V(?)
//                                 )
//
//   In most cases, you can use a fixed length char array. In that case,
//   you must give enough memory space to the static char array.
//   Otherwise, the segmentation fault will be occur and the execution
//   will be stopped. If you would like to avoid those fatal situation,
//   then it is highly recommended that you first calculate the whole length
//   of the operation APDU and allocate memory space dynamically.
//
// parameters:
//   char* parpaszOperationObjectName
//      The parpaszOperationObjectName is the session identifier which has
//      been generated by the requesting side, usually the manager side
//   UM3ClassIdentifier parUM3ClassIdentifier
//      The class identifier which contains the attribute object.
//   UM3ObjectName parUM3ObjectName
//      The object name of the object which has the name of parUM3Object-
//      Name value
//   UM3ClassIdentifier parUM3AttributeClassIdentifier
//      The attribute class identifier
//   UM3ObjectName parUM3AttributeObjectName
```

```
//      The attribute object name
//   AnyType parAnyValue
//      AnyType means ANY DEFINED BY UM3 ASN.1 module,
//      Thus, we have to consider every possible or present types as its
//      values.
//
// returns:
//   int
//      The length of the whole operation APDU.
//   -2
//      Wrong function has been called. You should call 'CalSetObject-
//      AttributeValueApduLen(......., AnyType parAnyValue,
//      int parValueLength) instead.
//
int
CalSetObjectAttributeValueApduLen(char* parpaszOperationObjectName,
                            UM3ClassIdentifier parUM3ClassIdentifier,
                            UM3ObjectName     parUM3ObjectName,
                            UM3ClassIdentifier
                                parUM3AttributeClassIdentifier,
                            UM3ObjectName
                                parUM3AttributeObjectName,
                            AnyType
                                parAnyValue)
{
   int dTempLenOne = _um3GetObjectNameLength(parpaszOperationObjectName);

   // get UM3ClassIdentifier type parameter length
   dTempLenOne += GetUM3ClassIdentifierApduLength("parUM3ClassIdentifier");

   // get UM3ObjectName type parameter length
   dTempLenOne += GetUM3ObjectNameApduLength("parUM3ObjectName",
                                     parUM3ObjectName);

   // get UM3AttributeClassIdentifier type parameter length
   dTempLenOne +=
      GetUM3ClassIdentifierApduLength("parUM3AttributeClassIdentifier");

   // get UM3AttributeObjectName type parameter length
   dTempLenOne +=
      GetUM3ObjectNameApduLength("parUM3AttributeObjectName",
                            parUM3AttributeObjectName);

   if (parUM3AttributeClassIdentifier == UM3_INTEGER16_CID) {
      dTempLenOne += GetUM3Integer16ApduLength("parAnyValue");
   }
   else if (parUM3AttributeClassIdentifier == UM3_INTEGER32_CID) {
      dTempLenOne += GetUM3Integer32ApduLength("parAnyValue");
   }
   else if (parUM3AttributeClassIdentifier == UM3_INTEGER64_CID) {
      dTempLenOne += GetUM3Integer64ApduLength("parAnyValue");
   }
   else if (parUM3AttributeClassIdentifier == UM3_UINTEGER16_CID) {
      dTempLenOne += GetUM3UInteger16ApduLength("parAnyValue");
   }
   else if (parUM3AttributeClassIdentifier == UM3_UINTEGER32_CID) {
      dTempLenOne += GetUM3UInteger32ApduLength("parAnyValue");
   }
   else if (parUM3AttributeClassIdentifier == UM3_UINTEGER64_CID) {
      dTempLenOne += GetUM3UInteger64ApduLength("parAnyValue");
   }
   else if (parUM3AttributeClassIdentifier == UM3_CHARACTER_STR_CID) {
      dTempLenOne += GetUM3CharStrApduLength("parAnyValue", (char*)parAnyValue);
   }
   else if (parUM3AttributeClassIdentifier == UM3_BOOLEAN_CID) {
      dTempLenOne += GetUM3BooleanApduLength("parAnyValue");
```

```
  }
  else if (parUM3AttributeClassIdentifier == UM3_ENUMERATED_CID) {
    dTempLenOne += GetUM3EnumeratedApduLength("parAnyValue");
  }
  else if (parUM3AttributeClassIdentifier == UM3_REAL_CID) {
    dTempLenOne += GetUM3RealApduLength("parAnyValue");
  }
  else if (parUM3AttributeClassIdentifier == UM3_NULL_CID) {
  }
  else if (parUM3AttributeClassIdentifier == UM3_DATETIME_CID) {
    dTempLenOne += GetUM3DateTimeApduLength("parAnyValue");
  }
  else if (parUM3AttributeClassIdentifier == UM3_CLASS_IDENTIFIER_CID) {
    dTempLenOne += GetUM3ClassIdentifierApduLength("parAnyValue");
  }
  else if (parUM3AttributeClassIdentifier == UM3_OBJECT_NAME_CID) {
    dTempLenOne +=
        GetUM3ObjectNameApduLength("parAnyValue", (char*)parAnyValue);
  }
  else {
    return (-2);
  }

  dTempLenOne += 8;

  return (dTempLenOne);
}

// CalSetObjectAttributeValueApduLen()
//
// descriptions:
//   Calculate the whole length of the GetObjectAttributeValue() UM3
//   operation APDU including the length of 4 parameters.
//
//   Only UM3BitString and UM3OctetString types will be accepted.
//
//   length =
//         T(4) + NL(2) + N(?) + L(2) + V(
//                                    T(4) + NL(2) + N(?) + L(2) + V(4)
//                                    T(4) + NL(2) + N(?) + L(2) + V(?)
//                                    T(4) + NL(2) + N(?) + L(2) + V(4)
//                                    T(4) + NL(2) + N(?) + L(2) + V(?)
//                                  )
//
//   In most cases, you can use a fixed length char array. In that case,
//   you must give enough memory space to the static char array.
//   Otherwise, the segmentation fault will be occur and the execution
//   will be stopped. If you would like to avoid those fatal situation,
//   it is highly recommended that you first calculate the whole length
//   of the operation APDU and allocate memory space dynamically.
//
// parameters:
//   char* parpaszOperationObjectName
//     The parpaszOperationObjectName is the session identifier which has
//     been generated by the requesting side, usually the manager side
//   UM3ClassIdentifier parUM3ClassIdentifier
//     The class identifier which contains the attribute object.
//   UM3ObjectName parUM3ObjectName
//     The object name of the object which has the name of the parUM3Object-
//     Name value
//   UM3ClassIdentifier parUM3AttributeClassIdentifier
//     The attribute class identifier
//   UM3ObjectName parUM3AttributeObjectName
//     The attribute object name
//   AnyType parAnyValue
//     AnyType means ANY DEFINED BY UM3 ASN.1 module,
```

```
//      Thus, we have to consider every possible or present types as its
//      values.
//      The type of AnyType can be determined with the parUM3Attribute-
//      ClassIdentifier parameter value.
//   int pardValueLength
//      The length of the value type.
//
//      CAUTION: The length of the value means the length of the
//      bit or octet, not the length of the encoded APDU.
//
// returns:
//   int
//      The length of the whole operation APDU.
//   -2
//      Wrong function has been called. 'CalSetObjectAttributeValue-
//      ApduLen(....AnyType parAnyValue) should be called.
//
int
CalSetObjectAttributeValueApduLen(char* parpaszOperationObjectName,
                            UM3ClassIdentifier parUM3ClassIdentifier,
                            UM3ObjectName     parUM3ObjectName,
                            UM3ClassIdentifier
                                parUM3AttributeClassIdentifier,
                            UM3ObjectName
                                parUM3AttributeObjectName,
                            AnyType
                                parAnyValue,
                            int  pardValueLength)
{
   int dTempLenOne = _um3GetObjectNameLength(parpaszOperationObjectName);

   // get UM3ClassIdentifier type parameter length
   dTempLenOne += GetUM3ClassIdentifierApduLength("parUM3ClassIdentifier");

   // get UM3ObjectName type parameter length
   dTempLenOne += GetUM3ObjectNameApduLength("parUM3ObjectName",
                                     parUM3ObjectName);

   // get UM3AttributeClassIdentifier type parameter length
   dTempLenOne +=
      GetUM3ClassIdentifierApduLength("parUM3AttributeClassIdentifier");

   // get UM3AttributeObjectName type parameter length
   dTempLenOne +=
      GetUM3ObjectNameApduLength("parUM3AttributeObjectName",
                            parUM3AttributeObjectName);

   if (parUM3AttributeClassIdentifier == UM3_BIT_STR_CID) {
      dTempLenOne +=
         GetUM3BitStrApduLength("parAnyValue", pardValueLength);
   }
   else if (parUM3AttributeClassIdentifier == UM3_OCTET_STR_CID) {
      dTempLenOne +=
         GetUM3OctetStrApduLength("parAnyValue", pardValueLength);
   }

   dTempLenOne += 8;

   return (dTempLenOne);
}

// EncodeSetObjectAttributeValue()
//
// descriptions:
//   Encodes the SetObjectAttributeValue() UM3 operation with UM3 SER.
//   The encoding steps of SetObjectAttributeValue() operation is similar
```

```
//   to the steps of GetObjectAttributeValue() UM3 operation except that
//   the very value which has to be set to the peer's information model
//   object is included in the parameter, 'AnyType parValue'.
//   The exact type of the given 'AnyType parValue' is given with the
//   parameter 'UM3ClassIdentifier parUM3AttributeClassIdentifier'.
//   The object name of the attribute to be set is 'UM3ObjectName
//   parUM3AttributeObjectName' parameter.
//
// parameters:
//   char* parpachApdu
//      The packet stream to be filled with the encoded content of the
//      operation
//   int pardApduLen
//      The length of the APDU array, not the length of the encoded packet
//   char* parpaszOperationObjectName
//      Operation object name i.e. the identifier of the operation
//      which is represented by the session identifier.
//   UM3ClassIdentifier parUM3ClassIdentifier
//      Object which has the attribute to be set with the given value
//   UM3ObjectName parUM3ObjectName
//      Object name of the above 'parUM3ClassIdentifier' object
//   UM3ClassIdentifier parUM3AttributeClassIdentifier
//      UM3 class identifier of the attribute object which is contained by
//      the object 'parUM3ClassIdentifier' and 'parUM3ObjectName'
//   UM3ObjectName parUM3AttributeObjectName
//      Object name of the attribute
//   AnyType parValue
//      Attribute value which has to be set as the new value of the
//      attribute identified with 'parUM3AttributeClassIdentifier' and
//      'parUM3AttributeObjectName' parameters
//
// returns:
//   1
//      success
//  -1
//      Failed because the given parpachApdu array has not been allocated
//      properly.
int
EncodeSetObjectAttributeValue(char*           parpachApdu,
                        int             pardApduLen,
                        char*           parpaszOperationObjectName,
                        UM3ClassIdentifier parUM3ClassIdentifier,
                        UM3ObjectName     parUM3ObjectName,
                        UM3ClassIdentifier
                               parUM3AttributeClassIdentifier,
                        UM3ObjectName
                               parUM3AttributeObjectName,
                        AnyType  parValue)
{
  char* pachPseudo = NULL;
  pachPseudo = parpachApdu;

  memset(pachPseudo, 0x00, pardApduLen);

  // T field
  unsigned int dTValue = SET_OBJ_ATTR_VALUE_CID;

  memcpy(pachPseudo, &dTValue, SIZEOF_UM3_T);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_T);

  pachPseudo += SIZEOF_UM3_T;

  // NL field
  unsigned short int dNLen =
     _um3GetObjectNameLength(parpaszOperationObjectName);
  memcpy(pachPseudo, &dNLen, SIZEOF_UM3_NL);
```

```
_um3ByteSwap(pachPseudo, SIZEOF_UM3_NL);

pachPseudo += SIZEOF_UM3_NL;

// N field
memcpy(pachPseudo, parpaszOperationObjectName, dNLen);
pachPseudo += dNLen;

// Here, we need to enter into the nested packet generation mode
// L field

// get UM3ClassIdentifier type parameter length
int dTempLenAll;
int dTempLenOne = GetUM3ClassIdentifierApduLength("parUM3ClassIdentifier");

// get UM3ObjectName type parameter length
int dTempLenTwo = GetUM3ObjectNameApduLength("parUM3ObjectName",
                                     parUM3ObjectName);

// get UM3AttributeClassIdentifier type parameter length
int dTempLenThree =
   GetUM3ClassIdentifierApduLength("parUM3AttributeClassIdentifier");

// get UM3AttributeObjectName type parameter length
int dTempLenFour =
   GetUM3ObjectNameApduLength("parUM3AttributeObjectName",
                          parUM3AttributeObjectName);

// get AnyType type value parameter length
// check the type and ...

int dTempLenFive;

if (parUM3AttributeClassIdentifier == UM3_INTEGER16_CID) {
   dTempLenFive = GetUM3Integer16ApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_INTEGER32_CID) {
   dTempLenFive = GetUM3Integer32ApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_INTEGER64_CID) {
   dTempLenFive = GetUM3Integer64ApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_UINTEGER16_CID) {
   dTempLenFive = GetUM3UInteger16ApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_UINTEGER32_CID) {
   dTempLenFive = GetUM3UInteger32ApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_UINTEGER64_CID) {
   dTempLenFive = GetUM3UInteger64ApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_CHARACTER_STR_CID) {
   dTempLenFive = GetUM3CharStrApduLength("parValue", (char*)parValue);
}
else if (parUM3AttributeClassIdentifier == UM3_BOOLEAN_CID) {
   dTempLenFive = GetUM3BooleanApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_ENUMERATED_CID) {
   dTempLenFive = GetUM3EnumeratedApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_REAL_CID) {
   dTempLenFive = GetUM3RealApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_NULL_CID) {
}
else if (parUM3AttributeClassIdentifier == UM3_DATETIME_CID) {
```

```
   dTempLenFive = GetUM3DateTimeApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_CLASS_IDENTIFIER_CID) {
   dTempLenFive = GetUM3ClassIdentifierApduLength("parValue");
}
else if (parUM3AttributeClassIdentifier == UM3_OBJECT_NAME_CID) {
   dTempLenFive = GetUM3ObjectNameApduLength("parValue", (char*)parValue);
}

dTempLenAll = dTempLenOne + dTempLenTwo + dTempLenThree + dTempLenFour
            + dTempLenFive;

memcpy(pachPseudo, &dTempLenAll, SIZEOF_UM3_L);
_um3ByteSwap(pachPseudo, SIZEOF_UM3_L);

pachPseudo += SIZEOF_UM3_L;

// V field (There are 4 parameters to be encoded.)

// parUM3ClassIdentifier
EncodeUM3ClassIdentifier(pachPseudo,
                     dTempLenOne,
                     "parUM3ClassIdentifier",
                     parUM3ClassIdentifier);
pachPseudo += dTempLenOne;

// parUM3ObjectName
EncodeUM3ObjectName(pachPseudo,
                 dTempLenTwo,
                 "parUM3ObjectName",
                 parUM3ObjectName);
pachPseudo += dTempLenTwo;

// parUM3AttributeClassIdentifier
EncodeUM3ClassIdentifier(pachPseudo,
                     dTempLenThree,
                     "parUM3AttributeClassIdentifier",
                     parUM3AttributeClassIdentifier);
pachPseudo += dTempLenThree;

// parUM3AttributeObjectName
EncodeUM3ObjectName(pachPseudo,
                 dTempLenFour,
                 "parUM3AttributeObjectName",
                 parUM3AttributeObjectName);
pachPseudo += dTempLenFour;

// AnyType parValue
if (parUM3AttributeClassIdentifier == UM3_INTEGER16_CID) {
   short int* pdTemp = (short int*)parValue;
   EncodeUM3Integer16(pachPseudo,
                   dTempLenFive,
                   "parValue",
                   *pdTemp);
}
else if (parUM3AttributeClassIdentifier == UM3_INTEGER32_CID) {
   int* pdTemp = (int *)parValue;
   EncodeUM3Integer16(pachPseudo,
                   dTempLenFive,
                   "parValue",
                   *pdTemp);
}
else if (parUM3AttributeClassIdentifier == UM3_INTEGER64_CID) {
   long long int* pdTemp = (long long int *)parValue;
   EncodeUM3Integer64(pachPseudo,
                   dTempLenFive,
```

```
                  "parValue",
                  *pdTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_UINTEGER16_CID) {
    unsigned short int* pdTemp = (unsigned short int*)parValue;
    EncodeUM3Integer16(pachPseudo,
                  dTempLenFive,
                  "parValue",
                  *pdTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_UINTEGER32_CID) {
    unsigned int* pdTemp = (unsigned int *)parValue;
    EncodeUM3Integer16(pachPseudo,
                  dTempLenFive,
                  "parValue",
                  *pdTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_UINTEGER64_CID) {
    unsigned long long int* pdTemp = (unsigned long long int *)parValue;
    EncodeUM3Integer64(pachPseudo,
                  dTempLenFive,
                  "parValue",
                  *pdTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_CHARACTER_STR_CID) {
    EncodeUM3CharacterString(pachPseudo,
                        dTempLenFive,
                        "parValue",
                        (char*)parValue);
  }
  else if (parUM3AttributeClassIdentifier == UM3_BOOLEAN_CID) {
    int *dTemp = (int *)parValue;
    EncodeUM3Boolean(pachPseudo,
                 dTempLenFive,
                 "parValue",
                 *dTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_ENUMERATED_CID) {
    int *dTemp = (int *)parValue;
    EncodeUM3Enumerated(pachPseudo,
                   dTempLenFive,
                   "parValue",
                   *dTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_REAL_CID) {
    double *dTemp = (double *)parValue;
    EncodeUM3Real(pachPseudo,
                dTempLenFive,
                "parValue",
                *dTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_NULL_CID) {
  }
  else if (parUM3AttributeClassIdentifier == UM3_DATETIME_CID) {
    int *dTemp = (int *)parValue;
    EncodeUM3DateTime(pachPseudo,
                  dTempLenFive,
                  "parValue",
                  *dTemp);
  }
  else if (parUM3AttributeClassIdentifier == UM3_CLASS_IDENTIFIER_CID) {
    unsigned int * pdTemp = (unsigned int *)parValue;
    EncodeUM3ClassIdentifier(pachPseudo,
                        dTempLenFive,
                        "parValue",
                        *pdTemp);
```

```
  }
  else if (parUM3AttributeClassIdentifier == UM3_OBJECT_NAME_CID) {
    EncodeUM3ObjectName(pachPseudo,
                        dTempLenFive,
                        "parValue",
                        (char*)parValue);
  }
  else {
    return (-2);
  }

  return (1);
}

// EnocdeSetObjectAttributeValue()
//
// alias: overloaded
//   EncodeSetObjectAttributeValueWithValueLength()
//
// descriptions:
//   Provides the same functionality with the EncodeSetObjectAttribute-
//   Value() function, but 'EncodeSetObjectAttributeValueWithValueLength()'
//   concerns the encoding of the following UM3 primitive types;
//
//      UM3BitString
//      UM3OctetString
//
//   You should remember that the C++ language supports the function
//   overloading, which have different parameters with the same
//   function name. Change the name of the function if you are going to
//   compile this source file with C compiler.
//
//   Encodes the SetObjectAttributeValue() UM3 operation with UM3 SER.
//   The encoding steps of SetObjectAttributeValue() operation is similar
//   to the steps of GetObjectAttributeValue() UM3 operation except that
//   the very value which has to be set to the peer's information model
//   object is included in the parameter, 'AnyType parValue'.
//   The exact type of the given 'AnyType parValue' is given with the
//   parameter 'UM3ClassIdentifier parUM3AttributeClassIdentifier'.
//   The object name of the attribute to be set is 'UM3ObjectName
//   parUM3AttributeObjectName' parameter.
//
// parameters:
//   char* parpachApdu
//      The packet stream to be filled with the encoded content of the
//      operation
//   int pardApduLen
//      The length of the APDU array, not the length of the encoded packet
//   char* parpaszOperationObjectName
//      Operation object name i.e. the identifier of the operation
//      which is represented by the session identifier.
//   UM3ClassIdentifier parUM3ClassIdentifier
//      Object which has the attribute to be set with the given value
//   UM3ObjectName parUM3ObjectName
//      Object name of the above 'parUM3ClassIdentifier' object
//   UM3ClassIdentifier parUM3AttributeClassIdentifier
//      UM3 class identifier of the attribute object which is contained by
//      the object 'parUM3ClassIdentifier' and 'parUM3ObjectName'
//   UM3ObjectName parUM3AttributeObjectName
//      Object name of the attribute
//   AnyType parValue
//      Attribute value which has to be set as the new value of the
//      attribute identified with 'parUM3AttributeClassIdentifier' and
//      'parUM3AttributeObjectName' parameters
//   int pardValueLength
//      The length of the value, usually the length of the character array,
```

```
//
// returns:
//   1
//      success
//  -1
//      Failed because the given parpachApdu array has not been allocated
//      properly.
//  -2
//      Wrong function was called. You should call EncodeSetObjectAttribute-
//      Value() function instead of this.
//
int
// EncodeSetObjectAttributeValueWithValueLength(
EncodeSetObjectAttributeValue(
                        char*           parpachApdu,
                        int             pardApduLen,
                        char*           parpaszOperationObjectName,
                        UM3ClassIdentifier parUM3ClassIdentifier,
                        UM3ObjectName     parUM3ObjectName,
                        UM3ClassIdentifier
                               parUM3AttributeClassIdentifier,
                        UM3ObjectName
                               parUM3AttributeObjectName,
                        AnyType  parValue,
                        int     pardValueLength)
{
  char* pachPseudo = NULL;
  pachPseudo = parpachApdu;

  memset(pachPseudo, 0x00, pardApduLen);

  // T field
  unsigned int dTValue = SET_OBJ_ATTR_VALUE_CID;

  memcpy(pachPseudo, &dTValue, SIZEOF_UM3_T);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_T);

  pachPseudo += SIZEOF_UM3_T;

  // NL field
  unsigned short int dNLen =
    _um3GetObjectNameLength(parpaszOperationObjectName);
  memcpy(pachPseudo, &dNLen, SIZEOF_UM3_NL);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_NL);

  pachPseudo += SIZEOF_UM3_NL;

  // N field
  memcpy(pachPseudo, parpaszOperationObjectName, dNLen);
  pachPseudo += dNLen;

  // Here, we need to enter into the nested packet generation mode
  // L field

  // get UM3ClassIdentifier type parameter length
  int dTempLenOne = GetUM3ClassIdentifierApduLength("parUM3ClassIdentifier");

  // get UM3ObjectName type parameter length
  int dTempLenTwo = GetUM3ObjectNameApduLength("parUM3ObjectName",
                                       parUM3ObjectName);

  // get UM3AttributeClassIdentifier type parameter length
  int dTempLenThree =
    GetUM3ClassIdentifierApduLength("parUM3AttributeClassIdentifier");

  // get UM3AttributeObjectName type parameter length
```

```
int dTempLenFour =
   GetUM3ObjectNameApduLength("parUM3AttributeObjectName",
                         parUM3AttributeObjectName);

// get AnyType type value parameter length
// check the type and ...

int dTempLenFive;

if (parUM3AttributeClassIdentifier == UM3_BIT_STR_CID) {
   int* dTemp = (int*)parValue;
   dTempLenFive =  GetUM3BitStrApduLength("parValue", *dTemp);
}
else if (parUM3AttributeClassIdentifier == UM3_OCTET_STR_CID) {
   int* dTemp = (int*)parValue;
   dTempLenFive = GetUM3OctetStrApduLength("parValue", *dTemp);
}

int dTempLenAll = dTempLenOne + dTempLenTwo + dTempLenThree + dTempLenFour + dTempLenFive;

memcpy(pachPseudo, &dTempLenAll, SIZEOF_UM3_L);
_um3ByteSwap(pachPseudo, SIZEOF_UM3_L);

pachPseudo += SIZEOF_UM3_L;

// V field (There are 4 parameters to be encoded.)

// parUM3ClassIdentifier
EncodeUM3ClassIdentifier(pachPseudo,
                     dTempLenOne,
                     "parUM3ClassIdentifier",
                     parUM3ClassIdentifier);
pachPseudo += dTempLenOne;

// parUM3ObjectName
EncodeUM3ObjectName(pachPseudo,
                 dTempLenTwo,
                 "parUM3ObjectName",
                 parUM3ObjectName);
pachPseudo += dTempLenTwo;

// parUM3AttributeClassIdentifier
EncodeUM3ClassIdentifier(pachPseudo,
                     dTempLenThree,
                     "parUM3AttributeClassIdentifier",
                     parUM3AttributeClassIdentifier);
pachPseudo += dTempLenThree;

// parUM3AttributeObjectName
EncodeUM3ObjectName(pachPseudo,
                 dTempLenFour,
                 "parUM3AttributeObjectName",
                 parUM3AttributeObjectName);
pachPseudo += dTempLenFour;

// AnyType parValue
if (parUM3AttributeClassIdentifier == UM3_BIT_STR_CID) {
   EncodeUM3BitString(pachPseudo,
                   dTempLenFive,
                   "parValue",
                   (char*)parValue,
                   pardValueLength);
}
else if (parUM3AttributeClassIdentifier == UM3_OCTET_STR_CID) {
   EncodeUM3OctetString(pachPseudo,
                     dTempLenFive,
```

```
                      "parValue",
                      (char*)parValue,
                      pardValueLength);
  }
  else {
    return (-2);
  }

  return (1);
}

// EncodeOperationResponseFail()
//
// descriptions:
//  Encodes the UM3 operation, OperationResponse(), with respect to the
//  given operation result. In this case it is a FALSE, as a parameter of
//  the function.
//
//   In general, there are two kinds of categories of AnyType parAnyValue.
//
//   1) UM3Boolean parResult == TRUE
//
//      If the operation that received was UM3-GET service operation,
//      then the value of the AnyType parAnyValue parameter will be an object
//      of the value which the UM3-GET service operation would receive.
//
//      If the operation that received was UM3-SET kinds of service
//      operation, then the value of AnyType parAnyValue will be checked
//      by the receiver of the OperationResponse APDU.
//
//   2) UM3Boolean parResult == FALSE
//
//      If the operation that received was UM3-GET service operation,
//      then the reason is returned with an UM3Enumerated type object.
//      When the operation that received falls into the category of
//      UM3-SET service operation, the same kind of rule is applied to
//      the AnyType parAnyValue parameter.
//
// parameters:
//   char* parpachApdu
//      The packet stream to be encoded into UM3 SER
//   int pardApduLen
//      The length of the parpachApdu packet
//   UM3ObjectName parpaszOperationObjectName
//      The session identifier that received from the previous
//      operation.
//   UM3Boolean parResult
//      TRUE - success, FALSE - failed
//   UM3ErrorCode parErrorCode
//      Error code in UM3Enumerated type
//
// returns:
//   1
//      success
//  -1
//      failed.
//
int
EncodeOperationResponseFail(char*       parpachApdu,
                      int         pardApduLen,
                      UM3ObjectName parpaszOperationObjectName,
                      UM3Boolean   parResult,
                      UM3OperationErrorCode
                                  parErrorCode)
{
  char* pachPseudo = NULL;
```

```
  pachPseudo = parpachApdu;

  memset(pachPseudo, 0x00, pardApduLen);

  // T field
  unsigned int dTValue = OPERATION_RESPONSE_CID;

  memcpy(pachPseudo, &dTValue, SIZEOF_UM3_T);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_T);

  pachPseudo += SIZEOF_UM3_T;

  // NL field
  unsigned short int dNLen =
    _um3GetObjectNameLength(parpaszOperationObjectName);
  memcpy(pachPseudo, &dNLen, SIZEOF_UM3_NL);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_NL);

  pachPseudo += SIZEOF_UM3_NL;

  // N field
  memcpy(pachPseudo, parpaszOperationObjectName, dNLen);
  pachPseudo += dNLen;

  // Here, we need to enter into the nested packet generation mode
  // L field

  // get UM3ClassIdentifier type parameter length
  int dTempLenAll;
  int dTempLenOne = GetUM3BooleanApduLength("parResult");

  // get UM3ObjectName type parameter length
  int dTempLenTwo = GetUM3EnumeratedApduLength("parErrorCode");

  // get AnyType type value parameter length
  // check the type and ...

  dTempLenAll = dTempLenOne + dTempLenTwo;

  memcpy(pachPseudo, &dTempLenAll, SIZEOF_UM3_L);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_L);

  pachPseudo += SIZEOF_UM3_L;

  // V field (There are 4 parameters to be encoded.)
  EncodeUM3Boolean(pachPseudo,
                dTempLenOne,
                "parResult",
                0);
  pachPseudo += dTempLenOne;

  EncodeUM3Enumerated(pachPseudo,
                  dTempLenTwo,
                  "parErrorCode",
                  parErrorCode);
  pachPseudo += dTempLenTwo;

  return (1);
}

// CalOperationResponseFailApduLen()
//
// descriptions:
//   Calculate the whole length of the OperationResponse() APDU when
//   it transfers the information of operation result as failed.
//
```

```
//   length =
//         T(4) + NL(2) + N(?) + L(2) + V(
//                                    T(4) + NL(2) + N(?) + L(2) + V(1)
//                                    T(4) + NL(2) + N(?) + L(2) + V(4)
//                                  )
//
//   There are 3 unknown length and 11 fixed length fields.
//   Thus, the whole length of the APDU for 'OperationResponse()' in the case of a
//   failed situation is as follows;
//
//   length = 8 + 9 + 12 + ...
//
// about parameters:
//   Of course, we could make the function header as follows;
//
//   'CalOperationResponseFailApduLen(char* parpaszOperationObjectName);
//
//   In other words, the name of the parameters are constant and the length
//   of the two types, UM3Boolean and UM3OperationErrorCode(UM3Enumerated)
//   are also constants. So we don't have to pass the values through the
//   these two parameters.
//
//   But, sometimes it is more important to prevent human errors and
//   that is the simple reason why we leave the unnecessary parameters.
//
// parameters:
//   char* parpaszOperationObjectName
//      The parpaszOperationObjectName is the session identifier which has
//      been generated by the requesting side, usually the manager side
//   UM3ClassIdentifier parUM3ClassIdentifier
//      The class identifier which contains the attribute object.
//   UM3ObjectName parUM3ObjectName
//      The object name of the object which has the name of parUM3Object-
//      Name value
//   UM3ClassIdentifier parUM3AttributeClassIdentifier
//      The attribute class identifier
//   UM3ObjectName parUM3AttributeObjectName
//      The attribute object name
//
// returns:
//   int
//      The length of the whole operation APDU.
//
int
CalOperationResponseFailApduLen(char*     parpaszOperationObjectName,
                          UM3Boolean parResult,
                          UM3OperationErrorCode
                                   parErrorCode)
{
  int dTempLenOne = _um3GetObjectNameLength(parpaszOperationObjectName);

  // get UM3Boolean type parameter length
  dTempLenOne += GetUM3BooleanApduLength("parResult");

  // get UM3OperationErrorCode(UM3Enuerated) type parameter length
  dTempLenOne += GetUM3EnumeratedApduLength("parErrorCode");

  dTempLenOne += 29;

  return (dTempLenOne);
}

// EncodeOperationResponseSuccess()
//
// descriptions:
//   Encodes the OperationResponse() UM3 operation with the value of
```

```
//   success and AnyType values. Usually, the success to the Set-like
//   operation brings various kinds of results which are to be
//   returned to the calling environment.
//
//   There are two encoding function to the OperationResponse()
//   operations. One is the encoding function for the following types;
//
//      UM3BitString and UM3OctetString
//
//   The other is to encode the UM3 primitive types except the two
//   types listed above.
//
// parameters:
//   char* parpachApdu
//      The packet stream to be filled with the encoded content of the
//      operation.
//   int pardApduLen
//      The length of the APDU array, not the length of the encoded packet.
//   UM3ObjectName parpaszOperationObjectName
//      Operation object name i.e. the identifier of the operation
//      which is represented by the session identifier.
//   UM3Boolean parResult
//      The value to this parameter is a constant, 1, due to the function
//      is to encode the successful operation result.
//   UM3ClassIdentifier parAnyValueType
//      As its name indicates, the parameter to show the UM3 class type of
//      AnyType parAnyValue parameter
//   AnyType parAnyValue
//      AnyType value means ANY DEFINED BY UM3 ASN.1 module, and it includes
//      all the UM3 primitive types.
//
// returns:
//   1
//      success
//  -1
//      Failed because the given parpachApdu array has not been allocated
//      properly.
int
EncodeOperationResponseSuccess(char*       parpachApdu,
                           int         pardApduLen,
                           UM3ObjectName parpaszOperationObjectName,
                           UM3Boolean   parResult,
                           UM3ClassIdentifier
                                       parAnyValueType,
                           AnyType      parAnyValue)
{
  char* pachPseudo = NULL;
  pachPseudo = parpachApdu;

  memset(pachPseudo, 0x00, pardApduLen);

  // T field
  unsigned int dTValue = OPERATION_RESPONSE_CID;

  memcpy(pachPseudo, &dTValue, SIZEOF_UM3_T);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_T);

  pachPseudo += SIZEOF_UM3_T;

  // NL field
  unsigned short int dNLen =
     _um3GetObjectNameLength(parpaszOperationObjectName);
  memcpy(pachPseudo, &dNLen, SIZEOF_UM3_NL);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_NL);

  pachPseudo += SIZEOF_UM3_NL;
```

```
// N field
memcpy(pachPseudo, parpaszOperationObjectName, dNLen);
pachPseudo += dNLen;

// Here, we need to enter into the nested packet generation mode
// L field

// get UM3ClassIdentifier type parameter length
int dTempLenAll;
int dTempLenOne = GetUM3BooleanApduLength("parResult");

// get AnyType type value parameter length
// check the type and ...

int dTempLenFive;

if (parAnyValueType == UM3_INTEGER16_CID) {
   dTempLenFive = GetUM3Integer16ApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_INTEGER32_CID) {
   dTempLenFive = GetUM3Integer32ApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_INTEGER64_CID) {
   dTempLenFive = GetUM3Integer64ApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_UINTEGER16_CID) {
   dTempLenFive = GetUM3UInteger16ApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_UINTEGER32_CID) {
   dTempLenFive = GetUM3UInteger32ApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_UINTEGER64_CID) {
   dTempLenFive = GetUM3UInteger64ApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_CHARACTER_STR_CID) {
   dTempLenFive = GetUM3CharStrApduLength("parAnyValue", (char*)parAnyValue);
}
else if (parAnyValueType == UM3_BOOLEAN_CID) {
   dTempLenFive = GetUM3BooleanApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_ENUMERATED_CID) {
   dTempLenFive = GetUM3EnumeratedApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_REAL_CID) {
   dTempLenFive = GetUM3RealApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_NULL_CID) {
}
else if (parAnyValueType == UM3_DATETIME_CID) {
   dTempLenFive = GetUM3DateTimeApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_CLASS_IDENTIFIER_CID) {
   dTempLenFive = GetUM3ClassIdentifierApduLength("parAnyValue");
}
else if (parAnyValueType == UM3_OBJECT_NAME_CID) {
   dTempLenFive =
      GetUM3ObjectNameApduLength("parAnyValue", (char*)parAnyValue);
}

dTempLenAll = dTempLenOne + dTempLenFive;

memcpy(pachPseudo, &dTempLenAll, SIZEOF_UM3_L);
_um3ByteSwap(pachPseudo, SIZEOF_UM3_L);

pachPseudo += SIZEOF_UM3_L;
```

```
// V field (There are 4 parameters to be encoded.)

// parResult
EncodeUM3Boolean(pachPseudo,
              dTempLenOne,
              "parResult",
              parResult);
pachPseudo += dTempLenOne;

// AnyType parValue
if (parAnyValueType == UM3_INTEGER16_CID) {
   short int* pdTemp = (short int*)parAnyValue;
   EncodeUM3Integer16(pachPseudo,
                   dTempLenFive,
                   "parAnyValue",
                   *pdTemp);
}
else if (parAnyValueType == UM3_INTEGER32_CID) {
   int* pdTemp = (int *)parAnyValue;
   EncodeUM3Integer16(pachPseudo,
                   dTempLenFive,
                   "parAnyValue",
                   *pdTemp);
}
else if (parAnyValueType == UM3_INTEGER64_CID) {
   long long int* pdTemp = (long long int *)parAnyValue;
   EncodeUM3Integer64(pachPseudo,
                   dTempLenFive,
                   "parAnyValue",
                   *pdTemp);
}
else if (parAnyValueType == UM3_UINTEGER16_CID) {
   unsigned short int* pdTemp = (unsigned short int*)parAnyValue;
   EncodeUM3Integer16(pachPseudo,
                   dTempLenFive,
                   "parAnyValue",
                   *pdTemp);
}
else if (parAnyValueType == UM3_UINTEGER32_CID) {
   unsigned int* pdTemp = (unsigned int *)parAnyValue;
   EncodeUM3Integer16(pachPseudo,
                   dTempLenFive,
                   "parAnyValue",
                   *pdTemp);
}
else if (parAnyValueType == UM3_UINTEGER64_CID) {
   unsigned long long int* pdTemp = (unsigned long long int *)parAnyValue;
   EncodeUM3Integer64(pachPseudo,
                   dTempLenFive,
                   "parAnyValue",
                   *pdTemp);
}
else if (parAnyValueType == UM3_CHARACTER_STR_CID) {
   EncodeUM3CharacterString(pachPseudo,
                         dTempLenFive,
                         "parAnyValue",
                         (char*)parAnyValue);
}
else if (parAnyValueType == UM3_BOOLEAN_CID) {
   int *dTemp = (int *)parAnyValue;
   EncodeUM3Boolean(pachPseudo,
                 dTempLenFive,
                 "parAnyValue",
                 *dTemp);
}
```

```
   else if (parAnyValueType == UM3_ENUMERATED_CID) {
      int *dTemp = (int *)parAnyValue;
      EncodeUM3Enumerated(pachPseudo,
                          dTempLenFive,
                          "parAnyValue",
                          *dTemp);
   }
   else if (parAnyValueType == UM3_REAL_CID) {
      double *dTemp = (double *)parAnyValue;
      EncodeUM3Real(pachPseudo,
                    dTempLenFive,
                    "parAnyValue",
                    *dTemp);
   }
   else if (parAnyValueType == UM3_NULL_CID) {
   }
   else if (parAnyValueType == UM3_DATETIME_CID) {
      int *dTemp = (int *)parAnyValue;
      EncodeUM3DateTime(pachPseudo,
                        dTempLenFive,
                        "parAnyValue",
                        *dTemp);
   }
   else if (parAnyValueType == UM3_CLASS_IDENTIFIER_CID) {
      unsigned int * pdTemp = (unsigned int *)parAnyValue;
      EncodeUM3ClassIdentifier(pachPseudo,
                               dTempLenFive,
                               "parAnyValue",
                               *pdTemp);
   }
   else if (parAnyValueType == UM3_OBJECT_NAME_CID) {
      EncodeUM3ObjectName(pachPseudo,
                          dTempLenFive,
                          "parAnyValue",
                          (char*)parAnyValue);
   }
   else {
      return (-2);
   }

   return (1);
}

// EnocdeOperationResponseSuccess()
//
// alias: overloaded
//    EncodeOperationResponseSuccessWithValueLength()
//
// descriptions:
//    Provides the same functionality with the EncodeOperationResponse-
//    Success() function, but this function concerns the encoding of
//    the following UM3 primitive types only;
//
//       UM3BitString
//       UM3OctetString
//
//    You should remember that the C++ language supports the function
//    overloading, which have different parameters with the same
//    function name. Change the name of the function if you are going to
//    compile this source file with C compiler.
//
// parameters:
//    char* parpachApdu
//       The packet stream to be filled with the encoded content of the
//       operation.
//    int pardApduLen
```

```
//      The length of the APDU array, not the length of the encoded packet.
//   UM3ObjectName parpaszOperationObjectName
//      Operation object name i.e. the identifier of the operation
//      which is represented by the session identifier.
//   UM3Boolean parResult
//      The result of the previous operation, success or fail.
//      In this function's parameter, it is a constant value that always
//      have the value of TRUE or 1.
//   UM3ClassIdentifier parAnyValueType
//      Represent the type of 'AnyType parAnyValue' parameter
//   AnyType parAnyValue
//      Attribute value which has to be set as the new value of the
//      attribute identified with 'parAnyValueType' parameter.
//   int parAnyValueLength
//      The length of the value, usually the length of the character array,
//
// returns:
//   1
//      success
//  -1
//      Failed because the given parpachApdu array has not been allocated
//      properly.
//  -2
//      Wrong function was called. You should call EncodeSetObjectAttribute-
//      Value() function instead of this.
//
int
EncodeOperationResponseSuccess(char*       parpachApdu,
                           int         pardApduLen,
                           UM3ObjectName parpaszOperationObjectName,
                           UM3Boolean   parResult,
                           UM3ClassIdentifier
                                       parAnyValueType,
                           AnyType      parAnyValue,
                           int         parAnyValueLength)
{
  char* pachPseudo = NULL;
  pachPseudo = parpachApdu;

  memset(pachPseudo, 0x00, pardApduLen);

  // T field
  unsigned int dTValue = OPERATION_RESPONSE_CID;

  memcpy(pachPseudo, &dTValue, SIZEOF_UM3_T);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_T);

  pachPseudo += SIZEOF_UM3_T;

  // NL field
  unsigned short int dNLen =
    _um3GetObjectNameLength(parpaszOperationObjectName);
  memcpy(pachPseudo, &dNLen, SIZEOF_UM3_NL);
  _um3ByteSwap(pachPseudo, SIZEOF_UM3_NL);

  pachPseudo += SIZEOF_UM3_NL;

  // N field
  memcpy(pachPseudo, parpaszOperationObjectName, dNLen);
  pachPseudo += dNLen;

  // Here, we need to enter into the nested packet generation mode
  // L field

  // get UM3Booleantype parameter length
  int dTempLenOne = GetUM3BooleanApduLength("parResult");
```

```
   // get AnyType type value parameter length
   // check the type and ...

   // get UM3BitString or UM3OctetString type parameter length
   int dTempLenTwo;
   if (parAnyValueType == UM3_BIT_STR_CID) {
      int* dTemp = (int*)parAnyValueLength;
      dTempLenTwo =  GetUM3BitStrApduLength("parAnyValue", *dTemp);
   }
   else if (parAnyValueType == UM3_OCTET_STR_CID) {
      int* dTemp = (int*)parAnyValueLength;
      dTempLenTwo = GetUM3OctetStrApduLength("parAnyValue", *dTemp);
   }

   int dTempLenAll = dTempLenOne + dTempLenTwo;

   memcpy(pachPseudo, &dTempLenAll, SIZEOF_UM3_L);
   _um3ByteSwap(pachPseudo, SIZEOF_UM3_L);

   pachPseudo += SIZEOF_UM3_L;

   // V field (There are 2 parameters to be encoded.)

   // parUM3ClassIdentifier
   EncodeUM3Boolean(pachPseudo,
                 dTempLenOne,
                 "parResult",
                 1);
   pachPseudo += dTempLenOne;

   // AnyType parAnyValue
   if (parAnyValueType == UM3_BIT_STR_CID) {
      EncodeUM3BitString(pachPseudo,
                     dTempLenTwo,
                     "parAnyValue",
                     (char*)parAnyValue,
                     parAnyValueLength);
   }
   else if (parAnyValueType == UM3_OCTET_STR_CID) {
      EncodeUM3OctetString(pachPseudo,
                       dTempLenTwo,
                       "parAnyValue",
                       (char*)parAnyValue,
                       parAnyValueLength);
   }
   else {
      return (-2);
   }

   return (1);
}

// CalOperationResponseSuccessApduLen()
//
// descriptions:
//    Calculate the total length of the byte array to hold the UM3 SER
//    encoded APDU. There are two kinds of 'CalOperationResponseSuccess-
//    ApduLen()' functions. One is the function for UM3BitString and
//    UM3OctetString types. The other is the function for the
//    rest of the UM3 Primitive types.
//
//    length = T(4) + NL(2) + N(x) + L(2) + V (
//                              T(4) + NL(2) + N(x) + L(2) + V(1)
//                              T(4) + NL(2) + N(x) + L(2) + V(x) )
//
```

```
//   The length of V(x) in the third line of above equation is varied
//   from 1 to several 10s of bytes with respect to the types in Any-
//   Type.
//
// parameters:
//   char* parpaszOperationObjectName
//      The session identification number that has been extracted from
//      the previous request.
//   UM3Boolean parResult
//      The value of parResult is always 0xFF because this function encodes
//      the successful operation response. The size of the bytes
//      is also the constant value of 1.
//      Thus it may be unnecessary to give the value to this function,
//      but not to make the function error prone, we generously provide
//      the value to it.
//      'parResult' is the name of the parameter, and it is also a constant
//      string.
//
//   UM3ClassIdentifier parAnyValueType
//      To encode the given UM3 type which is represented as
//      AnyType parAnyValue, we should know what the exact type is.
//      UM3ClassIdentifier parAnyValueType gives the exact type
//      information of AnyType parAnyValue parameter.
//   AnyType parAnyValue
//      The function brings the information of UM3 primitive type value
//      which has been type casted as AnyType. Two types, UM3Character-
//      string and UM3ObjectName types, need to know the length of the
//      given string to calculate the total bytes of the type.
//      The rest of the UM3 primitive types do not require the par –
//      AnyValue.
//
// returns:
//   1
//      success
//  -1
//      failed
//  -2
//      You are trying to call an improper function or calling mistakenly.
//      You should call another function which has the 'int parAnyValue-
//      Length' as its last parameter.
//
int
CalOperationResponseSuccessApduLen(char*     parpaszOperationObjectName,
                            UM3Boolean parResult,
                            UM3ClassIdentifier
                                    parAnyValueType,
                            AnyType   parAnyValue)
{
  int dTempLenOne = _um3GetObjectNameLength(parpaszOperationObjectName);

  // get UM3Boolean type parameter length
  dTempLenOne += GetUM3BooleanApduLength("parResult");

  // find out the length of the AnyType with the given 'parAnyValueType'
  //
  if (parAnyValueType == UM3_INTEGER16_CID) {
    dTempLenOne += GetUM3Integer16ApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_INTEGER32_CID) {
    dTempLenOne += GetUM3Integer32ApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_INTEGER64_CID) {
    dTempLenOne += GetUM3Integer64ApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_UINTEGER16_CID) {
    dTempLenOne += GetUM3UInteger16ApduLength("parAnyValue");
```

```
  }
  else if (parAnyValueType == UM3_UINTEGER32_CID) {
    dTempLenOne += GetUM3UInteger32ApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_UINTEGER64_CID) {
    dTempLenOne += GetUM3UInteger64ApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_CHARACTER_STR_CID) {
    dTempLenOne += GetUM3CharStrApduLength("parAnyValue", (char*)parAnyValue);
  }
  else if (parAnyValueType == UM3_BOOLEAN_CID) {
    dTempLenOne += GetUM3BooleanApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_ENUMERATED_CID) {
    dTempLenOne += GetUM3EnumeratedApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_REAL_CID) {
    dTempLenOne += GetUM3RealApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_NULL_CID) {
  }
  else if (parAnyValueType == UM3_DATETIME_CID) {
    dTempLenOne += GetUM3DateTimeApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_CLASS_IDENTIFIER_CID) {
    dTempLenOne += GetUM3ClassIdentifierApduLength("parAnyValue");
  }
  else if (parAnyValueType == UM3_OBJECT_NAME_CID) {
    dTempLenOne +=
        GetUM3ObjectNameApduLength("parAnyValue", (char*)parAnyValue);
  }
  else {
    return (-2);
  }

  dTempLenOne += 8;

  return (dTempLenOne);
}

// CalOperationResponseSuccessApduLen()
//
// descriptions:
//   Calculate the total length of the byte array to hold the UM3 SER
//   encoded APDU. There are two kinds of 'CalOperationResponseSuccess-
//   ApduLen()' functions. One is the function for UM3BitString and
//   UM3OctetString types. The other is the function for the
//   rest of the UM3 Primitive types. This function provide the cal-
//   culation of the types UM3BitString and UM3OctetString. Those two
//   types values need to know the length of the byte string
//   because they are not the null terminated string type.
//
//   length = T(4) + NL(2) + N(x) + L(2) + V (
//                           T(4) + NL(2) + N(x) + L(2) + V(1)
//                           T(4) + NL(2) + N(x) + L(2) + V(x) )
//
//   The length of V(x) in the third line of the above equation is varied
//   from 1 to several 10s of bytes with respect to the types in Any-
//   Type.
//
// parameters:
//   char* parpaszOperationObjectName
//      The session identification number that has been extracted from
//      the previous request
//   UM3Boolean parResult
//      The value of parResult is always 0xFF because this function encodes
```

```
//      the successful operation response. The size of the bytes
//      is also the constant value of 1.
//      Thus, it may be unnecessary to give the value to this function,
//      but not to make the function error prone, we generously provide
//      the value to it.
//      'parResult' is the name of the parameter, and it is also a constant
//      string.
//
//   UM3ClassIdentifier parAnyValueType
//      To encode the given UM3 type which is represented as
//      AnyType parAnyValue, we should know what the exact type is.
//      UM3ClassIdentifier parAnyValueType gives the exact type
//      information for AnyType parAnyValue parameter.
//   AnyType parAnyValue
//      The function brings the information of UM3 primitive type value
//      which has been type casted as AnyType. Two types, UM3Character-
//      string and UM3ObjectName types, need to know the length of the
//      given string to calculate the total bytes of the type.
//      The rest of the UM3 primitive types do not require the par –
//      AnyValue.
//   int parAnyValueLength
//      The length of the bytes, not the total APDU length.
//
// returns:
//   1
//      success
//  -1
//      failed
//  -2
//      You are trying to call an improper function or calling mistakenly.
//      You should call another function which has the 'int parAnyValue-
//      Length' as its last parameter.
//
int
CalOperationResponseSuccessApduLen(char*     parpaszOperationObjectName,
                            UM3Boolean parResult,
                            UM3ClassIdentifier
                                    parAnyValueType,
                            AnyType    parAnyValue,
                            int        parAnyValueLength)
{
  int dTempLenOne = _um3GetObjectNameLength(parpaszOperationObjectName);

  // get UM3Boolean type parameter length
  dTempLenOne += GetUM3BooleanApduLength("parResult");

  // find out the length of the AnyType with the given 'parAnyValueType'
  //
  if (parAnyValueType == UM3_BIT_STR_CID) {
    dTempLenOne += GetUM3BitStrApduLength("parAnyValue", parAnyValueLength);
  }
  else if (parAnyValueType == UM3_OCTET_STR_CID) {
    dTempLenOne += GetUM3OctetStrApduLength("parAnyValue", parAnyValueLength);
  }
  else {
    return (-2);
  }

  dTempLenOne += 8;

  return (dTempLenOne);
}

/*
// EncodeSetObjectValue()
//
```

```
// descriptions:
//
// parameters:
//
// returns:
//
int
EncodeSetObjectValue()
{

}

// EncodeSetObjectAttributeValue()
//
// descriptions:
//
// parameters:
//
// returns:
//
int
EnocdeSetObjectAttributeValue()
{

}

// DecodeUM3OperationType()
//
// descriptions:
//
// parameters:
//
// returns:
//
unsigned int
DecodeUM3OperationT(char* parpachApdu)
{

}

// DecodeUM3OperationNL(char* parpachApdu)
//
// descriptions:
//
// parameters:
//
// returns:
//
unsigned short int
DecodeUM3OperationNL(char* parpachApdu)
{

}

// DecodeUM3OperationN()
//
// descriptions:
//   parpaszObjectName is the session number
//
// parameters:
//
// returns:
//
int
DecodeUM3OperationN(char* parpachApdu,
                char* parpaszObjectName)
```

```
{

}

// DecodeUM3OperationL()
//
// descriptions:
//
// parameters:
//
// returns:
//
unsigned short int
DecodeUM3OperationL(char* parpachApdu)
{

}

// DecodeUM3OperationV()
//
// descriptions:
//
// parameters:
//
// returns:
//
int
DecodeUM3OperationV(char* parpachApdu,
                 void* parpValue)
{

}
*/

// PrintOutUM3OperationApdu()
//
// descriptions:
//
// parameters:
//
// returns:
//
int
PrintOutUM3OperationApdu(char* parpachApdu)
{
   char* pachPseudo = NULL;

   pachPseudo = parpachApdu;

   // T field
   unsigned int dType;

   memcpy(&dType, pachPseudo, SIZEOF_UM3_T);
   _um3ByteSwap((char*)&dType, SIZEOF_UM3_T);

   unsigned short int dsL;

   if (dType == GET_OBJ_ATTR_VALUE_CID) {
      printf("\n");
      printf("T (GetObjectAttributeValue()) {\n");
      printf("   %02X %02X %02X %02X \n",
         *(pachPseudo + 0),
         *(pachPseudo + 1),
         *(pachPseudo + 2),
         *(pachPseudo + 3)
      );
```

```
    printf("}\n");
    pachPseudo += SIZEOF_UM3_T;

    pachPseudo += _printOutUM3ApduN(pachPseudo, &dsL);

    //_printOutGetObjectAttributeValueApdu(pachPseudo);

    printf("V {..... \n");

    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    PrintOutUM3ApduN(pachPseudo);

    printf(".....}\n");
  }
  else if (dType == SET_OBJ_ATTR_VALUE_CID) {
    printf("\n");
    printf("T (SetObjectAttributeValue()) {\n");
    printf("  %02X %02X %02X %02X \n",
      *(pachPseudo + 0),
      *(pachPseudo + 1),
      *(pachPseudo + 2),
      *(pachPseudo + 3)
    );
    printf("}\n");
    pachPseudo += SIZEOF_UM3_T;

    pachPseudo += _printOutUM3ApduN(pachPseudo, &dsL);

    //_printOutGetObjectAttributeValueApdu(pachPseudo);

    printf("V {..... \n");

    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    PrintOutUM3ApduN(pachPseudo);

    printf(".....}\n");
  }
  else if (dType == OPERATION_RESPONSE_CID) {
    printf("\n");
    printf("T (OperationResponse()) {\n");
    printf("  %02X %02X %02X %02X \n",
      *(pachPseudo + 0),
      *(pachPseudo + 1),
      *(pachPseudo + 2),
      *(pachPseudo + 3)
    );
    printf("}\n");
    pachPseudo += SIZEOF_UM3_T;

    pachPseudo += _printOutUM3ApduN(pachPseudo, &dsL);

    //_printOutGetObjectAttributeValueApdu(pachPseudo);

    printf("V {..... \n");

    pachPseudo += PrintOutUM3ApduN(pachPseudo);
    PrintOutUM3ApduN(pachPseudo);

    printf(".....}\n");
  }
```

```
}

// _printOutGetObjectAttributeValueApdu()
//
// descriptions:
//
// parameters:
//
// returns:
//
int
_printOutGetObjectAttributeValueApdu(char* parpachApdu)
{
   //

}

// _calParameterApduLength()
//
// descriptions:
//
// parameters:
//
// returns:
//
int
_calParameterApduLength()
{

}
```

Encoder.h

```
/*
   Encoder.h

   (c)2009 KT Corporation

   WunBae Jeon (ubjeon@kt.com)
*/

#ifndef _ENCODER_H
#define _ENCODER_H

// UM3 APDU
#define SIZEOF_UM3_T                  4
#define SIZEOF_UM3_NL                 2
#define SIZEOF_UM3_L                  2

// class identifiers
#define UM3_INTEGER16_CID    11
#define UM3_INTEGER32_CID    12
#define UM3_INTEGER64_CID    13

#define UM3_UINTEGER16_CID   14
#define UM3_UINTEGER32_CID   15
#define UM3_UINTEGER64_CID   16

#define UM3_CHARACTER_STR_CID         17
#define UM3_BIT_STR_CID               18
#define UM3_OCTET_STR_CID    19
```

```
#define UM3_BOOLEAN_CID               20
#define UM3_ENUMERATED_CID   21
#define UM3_REAL_CID                  22

#define UM3_NULL_CID                  23
#define UM3_DATETIME_CID     24

#define UM3_CLASS_IDENTIFIER_CID      25
#define UM3_OBJECT_NAME_CID           26

//temporary
#define GATEWAY_CID          10000
#define ANALOG_SENSOR_CID    10001

// class type size
#define SIZEOF_UM3_INTEGER16 2
#define SIZEOF_UM3_INTEGER32 4
#define SIZEOF_UM3_INTEGER64 8

#define SIZEOF_UM3_UINTEGER16         2
#define SIZEOF_UM3_UINTEGER32         4
#define SIZEOF_UM3_UINTEGER64         8

//#define SIZEOF_UM3_CHARACTER_STR    VARIABLE
//#define SIZEOF_UM3_BIT_STR          VARIABLE
//#define SIZEOF_UM3_OCTET_STR                VARIABLE

#define SIZEOF_UM3_ENUMERATED         4
#define SIZEOF_UM3_BOOLEAN   1
#define SIZEOF_UM3_REAL               8

#define SIZEOF_UM3_NULL               1
#define SIZEOF_UM3_DATETIME  4

#define SIZEOF_UM3_CLASS_IDENTIFIER   4
//#define SIZEOF_UM3_OBJECT_NANE      VARIABLE


// type definitions
typedef short int            UM3Integer16;
typedef int                           UM3Integer32;
typedef long long int                 UM3Integer64;

typedef unsigned short int   UM3UnsignedInteger16;
typedef unsigned int                  UM3UnsignedInteger32;
typedef unsigned long long int        UM3UnsignedInteger64;

typedef char*                         UM3CharacterString;
typedef char*                         UM3BitString;
typedef char*                         UM3OctetString;

//typedef enum                        UM3Enumerated;
typedef bool                          UM3Boolean;
typedef double                        UM3Real;
typedef int                           UM3DateTime;

typedef char*                         UM3CharacterString;
typedef char*                         UM3BitString;
typedef char*                         UM3OctetString;

typedef unsigned int                  UM3ClassIdentifier;
typedef char*                UM3ObjectName;

typedef void*                         AnyType;
```

```
// function definitions

int EncodeUM3Integer16(char* parpachApdu,
                  int   pardApduLen,
                  char* parpaszObjectName,
                  short int   pardValue);
int GetUM3Integer16ApduLength(char* parpaszObjectName);

int EncodeUM3Integer32(char* parpachApdu,
                  int   pardApduLen,
                  char* parpaszObjectName,
                  int   pardValue);
int GetUM3Integer32ApduLength(char* parpaszObjectName);

int EncodeUM3Integer64(char* parpachApdu,
                  int   pardApduLen,
                  char* parpaszObjectName,
                  long long int   pardValue);
int GetUM3Integer64ApduLength(char* parpaszObjectName);

int EncodeUM3UInteger16(char* parpachApdu,
                   int   pardApduLen,
                   char* parpaszObjectName,
                   unsigned short int   pardValue);
int GetUM3UInteger16ApduLength(char* parpaszObjectName);

int EncodeUM3UInteger32(char* parpachApdu,
                   int   pardApduLen,
                   char* parpaszObjectName,
                   unsigned int   pardValue);
int GetUM3UInteger32ApduLength(char* parpaszObjectName);

int EncodeUM3UInteger64(char* parpachApdu,
                   int   pardApduLen,
                   char* parpaszObjectName,
                   unsigned long long int   pardValue);
int GetUM3UInteger64ApduLength(char* parpaszObjectName);

int _encodeUM3Integer(char* parpachApdu,
                 int   pardApduLen,
                 char* parpaszObjectName,
                 void* pardValue,
                 int   pardClassIdentifier,
                 int   pardSize);
int _getUM3IntegerApduLength(char* parpaszObjectName,
                      int   pardClassIdentifier);

int EncodeUM3Enumerated(char* parpachApdu,
                   int   pardApduLen,
                   char* parpaszObjectName,
                   int   pardValue);
int GetUM3EnumeratedApduLength(char* parpaszObjectName);

int EncodeUM3Boolean(char* parpachApdu,
                int   pardApduLen,
                char* parpaszObjectName,
                int   pardValue);
int GetUM3BooleanApduLength(char* parpaszObjectName);

int EncodeUM3Real(char*  parpachApdu,
              int    pardApduLen,
              char*  parpaszObjectName,
              double pardValue);
int GetUM3RealApduLength(char* parpaszObjectName);

int EncodeUM3CharacterString(char*  parpachApdu,
```

```
                        int   pardApduLen,
                        char*  parpaszObjectName,
                        char*  parpaszValue);
int GetUM3CharStrApduLength(char* parpaszObjectName,
                       char* parpaszValue);

int EncodeUM3BitString(char*  parpachApdu,
                   int   pardApduLen,
                   char*  parpaszObjectName,
                   char*  parpachValue,
                   int   pardValueLen);
int GetUM3BitStrApduLength(char* parpaszObjectName,
                      int  pardOctetStringLength);

int EncodeUM3OctetString(char*  parpachApdu,
                    int   pardApduLen,
                    char*  parpaszObjectName,
                    char*  parpachValue,
                    int   pardValueLen);
int GetUM3OctetStrApduLength(char* parpaszObjectName,
                        int  pardOctetStringLength);

int EncodeUM3ClassIdentifier(char* parpachApdu,
                        int  pardApduLen,
                        char* parpaszObjectName,
                        unsigned int pardUM3ClassIdentifier);
int GetUM3ClassIdentifierApduLength(char* parpaszObjectName);

int EncodeUM3ObjectName(char* parpachApdu,
                    int  pardApduLen,
                    char* parpaszObjectName,
                    char* parpaszObjectNameValue);
int GetUM3ObjectNameApduLength(char* parpaszObjectName,
                          char* parpaszObjectNameValue);

int EncodeUM3Null(char* parpachApdu,
               int  pardApduLen,
               char* parpaszObjectName);
int GetUM3NullApduLength(char* parpaszObjectName);

int EncodeUM3DateTime(char* parpachApdu,
                  int  pardApduLen,
                  char* parpaszObjectName,
                  int  pardValue);
int GetUM3DateTimeApduLength(char* parpaszObjectName);

// decode
unsigned int DecodeUM3Type(char* parpachApdu);
unsigned short int DecodeUM3ObjectNameLength(char* parpachApdu);
int DecodeUM3ObjectName(char* parpachApdu,
                    char* parpaszObjectName);
unsigned short int DecodeUM3ValueLength(char* parpachApdu);
int DecodeUM3Value(char* parpachApdu, void* parpValue);


// utilities
int
PrintOutUM3Apdu(char* parpachUM3Apdu);

int
_printOutUM3Apdu(char* _parpachUM3Apdu,
              unsigned short int* papdL);

int
PrintOutUM3ApduN(char* parpachUM3Apdu);
```

```
int
_printOutUM3ApduN(char* _parpachUM3Apdu,
               unsigned short int* papdL);

int
_um3ByteSwap(char* parpachBytes,
          int   pardSize);
unsigned short int
_um3GetObjectNameLength(char* parpaszObjectName);



#endif
```

Operation.h

```
/*
   OperationEncoder.h

   (c)2009 KT Corporation

   WunBae Jeon (ubejon@kt.com)
*/
#ifndef _OPERATION_ENCODER_H
#define _OPERATION_ENCODER_H

#include <Encoder.h>

#define GET_OBJ_VALUE_CID    2001
#define GET_OBJ_ATTR_VALUE_CID        2002

#define SET_OBJ_VALUE_CID    2003
#define SET_OBJ_ATTR_VALUE_CID        2004

#define OPERATION_RESPONSE_CID        2005


// error code
typedef enum _operation_error_code {
   NotFound = 0,
   OutOfService,
   InvalidClassId,
   InvalidObjectName,
   Busy
} UM3OperationErrorCode;


// function definitions
int
EncodeGetObjectAttributeValue(char*           parpachApdu,
                         int             pardApduLen,
                         char*           parpaszOperationObjectName,
                         UM3ClassIdentifier parUM3ClassIdentifier,
                         UM3ObjectName     parUM3ObjectName,
                         UM3ClassIdentifier
                                 parUM3AttributeClassIdentifier,
                         UM3ObjectName
                                 parUM3AttributeObjectName);
int
CalGetObjectAttributeValueApduLen(char* parpaszOperationObjectName,
                            UM3ClassIdentifier parUM3ClassIdentifier,
                            UM3ObjectName     parUM3ObjectName,
```

```
                            UM3ClassIdentifier
                                parUM3AttributeClassIdentifier,
                            UM3ObjectName
                                parUM3AttributeObjectName);

int
EncodeSetObjectAttributeValue(char*           parpachApdu,
                          int             pardApduLen,
                          char*           parpaszOperationObjectName,
                          UM3ClassIdentifier parUM3ClassIdentifier,
                          UM3ObjectName     parUM3ObjectName,
                          UM3ClassIdentifier
                                 parUM3AttributeClassIdentifier,
                          UM3ObjectName
                                 parUM3AttributeObjectName,
                          AnyType  parValue);
int
//EncodeSetObjectAttributeValueWithValueLength(
EncodeSetObjectAttributeValue(char*           parpachApdu,
                          int             pardApduLen,
                          char*           parpaszOperationObjectName,
                          UM3ClassIdentifier parUM3ClassIdentifier,
                          UM3ObjectName     parUM3ObjectName,
                          UM3ClassIdentifier
                                 parUM3AttributeClassIdentifier,
                          UM3ObjectName
                                 parUM3AttributeObjectName,
                          AnyType  parValue,
                          int     pardValueLength);
int
CalSetObjectAttributeValueApduLen(char* parpaszOperationObjectName,
                             UM3ClassIdentifier parUM3ClassIdentifier,
                             UM3ObjectName     parUM3ObjectName,
                             UM3ClassIdentifier
                                 parUM3AttributeClassIdentifier,
                             UM3ObjectName
                                 parUM3AttributeObjectName,
                             AnyType
                                 parAnyValue);
int
CalSetObjectAttributeValueApduLen(char* parpaszOperationObjectName,
                             UM3ClassIdentifier parUM3ClassIdentifier,
                             UM3ObjectName     parUM3ObjectName,
                             UM3ClassIdentifier
                                 parUM3AttributeClassIdentifier,
                             UM3ObjectName
                                 parUM3AttributeObjectName,
                             AnyType
                                 parAnyValue,
                             int  pardValueLength);

/*
int
EncodeOperationResponse(UM3Boolean   parResult,
                    UM3ObjectName parUM3ObjectName,
                    AnyType      parAnyValue);
*/

//
int
EncodeOperationResponseFail(char*       parpachApdu,
                        int         pardApduLen,
                        UM3ObjectName parpaszOperationObjectName,
                        UM3Boolean   parResult,
                        UM3OperationErrorCode
                                    parErrorCode);
```

```
int
CalOperationResponseFailApduLen(char*     parpaszOperationObjectName,
                           UM3Boolean parResult,
                           UM3OperationErrorCode
                                    parErrorCode);

//
int
EncodeOperationResponseSuccess(char*       parpachApdu,
                           int         pardApduLen,
                           UM3ObjectName parpaszOperationObjectName,
                           UM3Boolean   parResult,
                           UM3ClassIdentifier
                                      parAnyValueType,
                           AnyType      parAnyValue);

int
EncodeOperationResponseSuccess(char*       parpachApdu,
                           int         pardApduLen,
                           UM3ObjectName parpaszOperationObjectName,
                           UM3Boolean   parResult,
                           UM3ClassIdentifier
                                     parAnyValueType,
                           AnyType      parAnyValue,
                           int         parAnyValueLength);

int
CalOperationResponseSuccessApduLen(char*     parpaszOperationObjectName,
                              UM3Boolean parResult,
                              UM3ClassIdentifier
                                       parAnyValueType,
                              AnyType   parAnyValue);
int
CalOperationResponseSuccessApduLen(char*     parpaszOperationObjectName,
                              UM3Boolean parResult,
                              UM3ClassIdentifier
                                       parAnyValueType,
                              AnyType   parAnyValue,
                              int       parAnyValueLength);


int
PrintOutUM3OperationApdu(char* parpachApdu);


#endif
```

Operation.h

```
/*
   Operation.h

   (c)2009 KT Corporation

   WunBae Jeon (ubjeon@kt.com)
*/

#ifndef _OPERATION_H_
#define _OPERATION_H_

#include <Encoder.h>
```

```
int SetObjectAttributeValue(int sd);
int GetObjectAttributeValue(int sd);

int
SetObjectAttributeValue(int             sd,
                    char*           parpaszUM3OperationObjectName,
                    UM3ClassIdentifier parpaszObjectClassIdentifier,
                    UM3ObjectName     parpaszObjectObjectName,
                    UM3ClassIdentifier parpaszAttributeClassIdentifier,
                    UM3ObjectName     parpaszAttributeObjectName,
                    AnyType          parpAnyTypeValue);

int
GetObjectAttributeValue(int             sd,
                    char*           parpaszUM3OperationObjectName,
                    UM3ClassIdentifier parpaszObjectClassIdentifier,
                    UM3ObjectName     parpaszObjectObjectName,
                    UM3ClassIdentifier parpaszAttributeClassIdentifier,
                    UM3ObjectName     parpaszAttributeObjectName);

int
OperationResponseSuccess(char*           parpachApdu,
                     int             pardApduLen,
                     char*           parpaszOperationObjectName,
                     UM3Boolean       parResult,
                     UM3ClassIdentifier parAnyValueType,
                     AnyType          parAnyValue);


#endif
```

이 곳은 본 권고안의 마지막 페이지 입니다.