(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
H04L 29/06 (2006.01)

(45) 공고일자 2006년07월28일
(11) 등록번호 10-0606025
(24) 등록일자 2006년07월20일

(21) 출원번호 10-2004-0094795
(22) 출원일자 2004년11월18일

(65) 공개번호 10-2006-0055243
(43) 공개일자 2006년05월23일

(73) 특허권자 삼성전자주식회사
경기도 수원시 영통구 매탄동 416

(72) 발명자 송봉규
경기 수원시 영통구 영통동 청명마을4단지아파트 405동 703호

최승필
경기 용인시 풍덕천2동 주공1단지아파트 106동 1501호

조원창
경기 용인시 풍덕천2동 용인3차 동보아파트 101동 504호

(74) 대리인 이건주

심사관 : 이동환

(54) 간이 망 관리 프로토콜 기반의 망 관리 장치 및 방법

요약

본 발명은 네트워크 장비를 관리하는 장치 및 방법에 관한 것으로, 특히 간이 망 관리 프로토콜(SNMP)을 사용하는 통신 장비를 관리하기 위한 망 관리 장치 및 방법에 관한 것이다.

컴파일 타임에 개발자가 응용프로그램을 통해 SNMP 인터페이스 헤더 파일을 작성하면, 추출부가 상기 인터페이스 헤더 파일을 근거로 MIB 파일과 객체 식별자 정보를 생성한다. 런 타임에는 매니저가 SNMP 요청을 하면, 에이전트가 상기 SNMP 요청 메시지에 포함된 OID정보를 객체 식별자 정보 처리부로 전송하고, 상기 객체 식별자 정보 처리부는 객체 식별자 정보 저장부를 참조하여 GMS 정보를 상기 에이전트에게 전달한다.

상기 에이전트는 상기 GMS 정보를 근거로 응용프로그램과 GMS 요청/응답 과정을 수행한다.

대표도

도 5

색인어

SNMP, OID, MIB

명세서

도면의 간단한 설명

도 1은 일반적인 SNMP의 매니저(manager)와 SNMP 에이전트(agent)간의 구성 및 제어 동작을 설명하기 위한 도면,

도 2는 종래 기술에 따른 SNMP 에이전트 개발 방법의 흐름도,

도 3은 종래 기술에 따른 상기 매니저와 응용프로그램들 간의 인터페이스 방법을 도시한 도면,

도 4는 본 발명에 따른 SNMP 에이전트의 블록 구성도,

도 5는 상기 도 4에 도시된 상기 SNMP 에이전트의 블록구성도 및 본 발명의 컴파일(Compile) 시간과 런(Run) 시간을 포함한 전체 동작흐름을 나타낸 도면,

도 6은 상기 SNMP 에이전트의 컴파일 동작 수행방식을 도시한 도면,

도 7은 본 발명의 실시 예에 따른 컴파일 타임 시 SNMP 에이전트내의 동작 흐름도,

도 8은 상기 SNMP 에이전트가 런 타임 시 상기 도 6의 컴파일 시에 생성된 OIDInfo 데이터 파일을 이용하여 수행하는 과정

도 9는 본 발명의 실시 예에 따른 런 타임 시 SNMP 에이전트내의 동작 흐름도,

도 10은 에이전트와 응용프로그램간의 통신을 위한 GMS PDU 구성을 도시한 도면,

도 11은 상기 에이전트가 Get/GetNext/GetFirst 메시지를 상기 응용프로그램에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면,

도 12는 상기 에이전트가 PreSet/Set 메시지를 상기 응용프로그램에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면,

도 13은 상기 에이전트가 GetBulk 메시지를 상기 응용프로그램에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면,

도 14는 상기 에이전트가 Notification(Trap) 메시지를 상기 응용프로그램에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면,

도 15a는 상기 객체 식별자 정보 파일을 구성하는 항목들을 개략적으로 도시한 도면,

도 15b는 상기 도 15a에 도시된 각 항목들을 구성하는 정보를 나타내는 테이블을 도시한 도면,

도 16은 상기 객체 식별자 정보 저장부(500)의 구조를 대략적으로 묘사해 놓은 도면,

도 17은 본 발명의 실시 예에 따른 GMS 정보를 찾는 방법을 도시한 도면,

도 18은 본 발명의 실시 예에 따른 상기 에이전트(400)가 노티피케이션 정보를 찾는 방법을 도시한 도면.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 네트워크 장비를 관리하는 장치 및 방법에 관한 것으로, 특히 간이 망 관리 프로토콜(Simple Network Management Protocol : 이하 SNMP라 함)을 사용하는 통신 장비를 관리하기 위한 망 관리 장치 및 방법에 관한 것이다.

지난 수년간 네트워크의 급속한 성장과 다양한 이질적인 시스템의 등장으로 네트워크를 통합적으로 관리하기 어려워졌으며, 네트워크가 대규모화되면서 많은 분야에서 네트워크 장비에 대한 관리가 필수 불가결한 요소가 되었다.

그러므로, 네트워크 관리자들은 다양한 네트워크 환경에서 포괄적으로 관리할 수 있는 네트워크 프레임워크(Frame work)를 요구하게 되었다. 이러한 요구로 인하여 세계 최대의 인터넷 표준 단체인 인터넷 공학 추진 팀(Internet Engineering Task Force : 이하 IETF라 함)은 인터넷 기반의 네트워크 장비를 관리하니 위한 표준으로 비교적 단순한 프로토콜인 SNMP를 채택하게 되었다.

일반적으로 SNMP를 적용하는 시스템에서는 관리 시스템을 매니저(manager)라고 통칭하며, 관리 대상을 에이전트(agent)라 통칭한다. 상기 매니저와 에이전트를 연결하는 관리 정보의 전달망은 TCP/IP(Transmission Contol Protocol/Internet Protocol) 방식에 준거한 것이며, SNMP를 사용한 통신은 매니저와 에이전트 사이에서 관리 정보 베이스(Management Information Base : 이하 MIB라 함)를 기초로 관리 정보 검색, 관리 정보 연속 검색, 관리 정보 바꿔 쓰기, 예외 동작의 통지 등의 명령을 사용하여 이루어진다.

상기 SNMP 에이전트는 관리 대상 장비에 존재하는 소프트웨어 모듈(software module)로서 MIB에 대한 정보를 갖고 있으며, 그 정보를 SNMP 프로토콜을 사용하여 상기 SNMP 매니저로 전달한다.

SNMP를 이용하여 관리되어야 할 특정한 정보(information), 자원(resource) 등을 객체(Object)라 하며, 이런 객체들을 모아놓은 집합체를 상기 MIB라하며, 상기 MIB 형식은 SNMP의 일부로서 정의되어 있고, 각 객체들은 ASN.1(Abstract Syntax Notation One)을 사용해서 정의되어진다.

상기 SNMP 에이전트는 네트워크 장비 기능에 관련된 파라미터(parameter)들로 구성된 MIB를 관리한다. 그러면, 상기 SNMP 매니저는 상기 SNMP 에이전트들이 제공하는 MIB 중에서 특정 값을 얻어와서 그 장비의 상태를 파악하거나 그 값을 변경한다.

상술한바와 같이 일반적으로 SNMP를 이용하여 네트워크를 관리한다는 것은 관리 대상인 장비들이 제공하는 MIB중에서 특정 값을 얻어와서 그 장비의 상태를 파악하거나 그 값을 변경함을 의미한다.

SNMP는 관리 방법의 용이성과 TCP/IP를 사용하는 다양한 종류의 장비에서 개발이 가능하며, 다양한 RFC(Request For Comment)를 통해서 관리 범위를 지정하거나 확장이 쉽게 가능하며 프로토콜의 구성이 간단하다. 또한, 구현을 쉽게 할 수 있는 이유로 현재 많은 관리 프로토콜 중에 SNMP가 가장 널리 사용되고 있다.

도 1은 일반적인 SNMP의 매니저(manager)(100)와 SNMP 에이전트(agent)(102)간의 구성 및 제어 동작을 설명하기 위한 도면이다.

먼저, 도 1에 도시된 SNMP 매니저(100)와 SNMP 에이전트간(102)에 송/수신되는 명령어에 관해 설명한다.

-GetRequest : 오브젝트(object)값을 읽기(read) 위한 요청 신호.

-GetNextRequest : 현재 오브젝트의 다음 오브젝트 값을 읽기 위한 요청 신호.

-GetResponse : Request에 대한 응답 신호.

-SetRequest : 오브젝트 값을 기록(write)하기 위한 신호.

-Trap : 예외 상황 알림.

상기 도 1의 SNMP 매니저(100)와 SNMP 에이전트(102)는 상기에 서술한 메시지를 이용하여 서로 통신을 수행한다.

이하 도 2 및 도 3을 통해 종래 기술에 따른 에이전트의 개발 방법 및 응용 프로그램과의 인터페이스 방법에 대해 살펴보기로 하겠다.

도 2는 종래 기술에 따른 SNMP 에이전트(102) 개발 방법의 흐름도이다.

200단계에서 네트워크 관리자는 상기 SNMP 에이전트(102)를 개발하기 위해 MIB에 대한 정의를 하고, 이에 해당하는 응용 프로그램과 인터페이스에 사용되는 구조를 정의한다.

상기 200단계에서 정의된 MIB를 근거로 202단계에서 MIB 파일을 생성한다. 204단계에서 상기 네트워크 관리자는 상기 생성된 MIB 파일을 코딩 및 컴파일하여 SNMP 에이전트(102)를 생성한다.

206단계에서 상기 네트워크 관리자는 MIB 또는 인터페이스의 수정 사항이 있는지 여부를 검사하여, 수정 사항이 있다면, 208단계로 진행하여 관리 항목을 정의한다.

상기 208단계에서 정의된 관리 항목을 근거로 210단계에서 상기 네트워크 관리자는 MIB 파일을 생성하고, 212단계에서 상기 생성된 MIB 파일을 근거로 상기 에이전트(102)를 재코딩 및 재컴파일을 한다.

도 3은 종래 기술에 따른 상기 매니저(100)와 응용프로그램(304)들 간의 인터페이스 방법을 도시한 도면이다.

상기 매니저(100)가 상기 응용프로그램(304)들과의 인터페이스를 하기 위해서는 각각의 구조에 대한 정보와 목적지에 대한 정보를 상기 에이전트(102)가 알고 있어야 한다. 그리고, 일반적인 SNMP 에이전트(102) 개발을 위한 툴(Tool)을 사용한 경우, 관리 항목에 해당하는 객체(Object)(302)들에서 해당하는 응용프로그램(304)에서 사용하는 구조를 이용하여 각각 구현해야 한다.

또한, 일반적인 SNMP 에이전트(102)의 개발 방법은 장비의 특성을 반영할 수 있도록 MIB에 대한 설계가 이루어지고 MIB에 설계된 내용에 따라서 SNMP 에이전트(102)의 개발 범위, 장비 내에서 관리 기능을 수행하는 응용 프로그램(304)들의 역할, SNMP 에이전트(102)와 응용 프로그램(304)들간의 인터페이스 방법 등이 결정된다.

현재 일반적으로 SNMP 프로토콜을 효과적으로 개발하기 위해서 일반적인 툴(Tool)을 사용하는데, 이러한 툴을 이용한 SNMP 에이전트(102)의 개발은 SNMP 에이전트(102)가 데이터를 갖고 있는 경우에는 개발이 쉽지만, 응용 프로그램(304)과의 인터페이스를 통해서 MIB 객체 값을 얻어 오는 경우 각각의 MIB 객체마다 다른 구조를 사용하기 때문에 SNMP 에이전트(102)의 개발이 어렵고 복잡해지게 된다.

또한, SNMP 매니저(100)는 SNMP 에이전트(102)를 통해서 장비내의 다른 응용 프로그램(304)들이 관리하는 관리항목에 접근을 하게 된다. 이 관리 항목은 MIB로 표현이 되며, 각각 장비의 특성에 따라 다르게 된다. 이런 MIB를 개발 초기에 완벽히 정의한다는 것은 사실상 불가능한 일이다.

종래의 기술은 이런 MIB를 기준으로 SNMP 에이전트(102)가 개발이 되기 때문에 각각의 장비마다 서로 다른 SNMP 에이전트(102)를 개발해야 한다. 또한 같은 장비 내에서도 관리되어야 할 항목들의 변경, 추가, 삭제는 수시로 발생할 수 있으며, 이런 변경 사항이 발생할 경우에 SNMP 에이전트(102)의 수정이 필요하게 된다.

상기와 같은 환경에서 MIB가 변경, 추가, 삭제될 때마다 SNMP 에이전트(102)를 다시 수정하고 재컴파일(Recompile)하는 것은 개발하는데 많은 시간과 노력이 필요하게 된다.

발명이 이루고자 하는 기술적 과제

따라서 본 발명의 목적은 네트워크 관리 시스템에서 관리 정보 베이스 파일과 객체 식별자 정보 파일을 실시간으로 생성하는 간이 망 관리 프로토콜 기반의 망 관리 장치 및 방법을 제공함에 있다.

본 발명은 다른 목적은 네트워크 관리 시스템을 구성하는 네트워크에서 발생한 변동사항에 맞게 상기 장비들의 동작을 제어하기 위한 간이 망 관리 프로토콜 관리 장치 및 방법을 제공함에 있다.

본 발명의 또 다른 목적은 네트워크 관리 시스템을 구성하는 장비들을 표준 방식에 의해 동작하게 하기 위한 간이 망 관리 프로토콜 관리 장치 및 방법을 제공함에 있다.

상기한 목적들을 달성하기 위한 본 발명에 따른 방법은, 간이 망 관리 프로토콜(SNMP)을 사용하여 통신 장비를 관리하기 위해 컴파일 타임시에 상기 통신 장비를 관리하기 위한 관리 정보 베이스(MIB) 파일과 상기 간이 망 관리 프로토콜 에이전트와 응용프로그램 간의 통신을 위한 객체 식별자 정보 파일을 생성하기 위한 방법에 있어서, 변경된 관리 항목을 입력받은 응용프로그램이 간이 망 관리 프로토콜 인터페이스 헤더 파일을 작성하는 과정과, 상기 작성된 간이 망 관리 프로토콜 인터페이스 헤더 파일을 읽어와서 관리 정보 베이스 파일과 객체 식별자 정보 파일을 생성하는 과정과, 상기 생성된 관리 정보 베이스 파일과 상기 객체 식별자 정보 파일을 저장하는 과정을 포함한다.

상기한 목적들을 달성하기 위한 본 발명의 장치는, 간이 망 관리 프로토콜(SNMP)을 사용하여 통신 장비를 관리하기 위해 컴파일 타임시에 상기 통신 장비를 관리하기 위한 관리 정보 베이스(MIB) 파일과 상기 간이 망 관리 프로토콜 에이전트와 응용프로그램 간의 통신을 위한 객체 식별자 정보 파일을 생성하기 위한 장치에 있어서, 상기 간이 망 관리 프로토콜 장비의 관리를 위해 응용프로그램이 작성한 헤더 파일을 저장하는 헤더 파일 저장부와, 상기 헤더 파일 저장부에서 상기 헤더 파일을 읽어서 상기 관리 정보 베이스 파일과 상기 간이 망 관리 프로토콜 에이전트와 상기 응용프로그램간의 메시지 교환을 위한 상기 객체 식별자 정보 파일을 작성하는 추출부와, 상기 추출부에서 작성한 상기 관리 정보 베이스 파일을 저장하는 관리 정보 베이스 파일 저장부와, 상기 추출부에서 작성한 상기 객체 식별자 정보 파일을 저장하는 객체 식별자 정보 저장부를 포함한다.

발명의 구성 및 작용

이하 본 발명의 실시 예를 첨부한 도면을 참조하여 상세히 설명하기로 하겠다. 하기에서 본 발명을 설명함에 있어, 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다.

본 발명에 있어서, SNMP 에이전트는 도 4와 같이 크게 4부분으로 구성된다.

도 4는 본 발명에 따른 SNMP 에이전트(102)의 블록 구성도이다. 상기 SNMP 에이전트(102)는 네트워크 엘리먼트(Element)에서 SNMP을 지원하는 기능을 수행한다.

상기 SNMP 에이전트(102)는 SNMP 메시지를 처리하고 응용프로그램들과의 인터페이스를 담당하는 에이전트(400)와 상기 응용프로그램과 상기 에이전트(400)간의 통신을 하기 위해 필요한 GMS(General Messsage Service)메시지를 만들기 위한 정보를 갖고 있는 라이브러리(Library)인 객체 식별자 정보(Object Identifier Information :이하 OIDInfo)를 처리하는 객체 식별자 정보 처리부(402)로 구성되어 있다.

또한, GMS 메시지와 MIB를 만들기 위한 정보를 도시되지 않은 객체 식별자 정보 저장부에 전달하는 추출부(Extractor)(404)와 응용프로그램과의 통신을 위해 사용되는 SNMP 인터페이스 헤더 파일과 상기 응용프로그램(504)이 관리해야 할 정보와 GMS 정보를 저장하는 헤더 파일 저장부(406)로 구성된다.

상기 객체 식별자 정보(OIDInfo) 처리부는 크게 나누어 두 가지의 기능을 제공한다. 첫째가 SNMP 에이전트(102)에게 응용프로그램(504)에 대한 GMS 정보와 구조(Structure)에 대한 정보를 제공한다. 둘째로는 상기 GMS 정보를 근거로 다시 SNMP 정보를 제공한다.

도 5는 상기 도 4에 도시된 상기 SNMP 에이전트의 블록구성도 및 본 발명의 컴파일(Compile) 시간과 런(Run) 시간을 포함한 전체 동작흐름을 나타낸 도면으로서, 하기에서 설명하도록 하겠다.

먼저, 상기 에이전트(400)는 SNMP 프로토콜 처리와 실제 관리 항목을 유지 관리하는 응용 프로그램과 GMS 메시지 송/수신 역할을 담당한다. 첫 번째로 망 관리자가 이용하는 매니저(100)로부터 미리 정해진 다양한 포맷의 PDU(Protocol Data Unit)을 받으면, 상기 객체 정보 처리부(402)로부터 메시지 ID, 포트 번호, 메시지 구조 등의 메시지 정보를 알아내서 응용프로그램(504)에게 해당 요청을 전달하고, 두 번째로 상기 응용프로그램(504)으로부터 안내(Notification : Trap) 메시지를 수신하면 상기 객체 식별자 정보 처리부(402)로부터 객체 식별자(Object IDentifier : 이하 OID라 함)정보를 얻어서 매니저(100)에게 트랩(Trap) PDU를 보내는 역할을 한다.

상기 객체 식별자 정보 처리부(402)는 관리 항목에 접근하기 위해서 응용프로그램(504)과 메시지 송수신을 위해 필요한 정보를 저장하고 있는 객체 식별자 정보 파일을 객체 식별자 정보 저장부로부터 읽어오고, 검색할 수 있게 해주는 역할을 한다. 상기 객체 식별자 정보 저장부(500)는 OIDInfo 라이브러리 APIs와 OIDInfo 데이터 파일로 구성되며 상기 추출부(404)가 생성한 데이터 파일을 제공받는다.

상기 추출부(404)는 상기 응용프로그램(504)이 관리 항목으로 정의한 SNMP 인터페이스 헤더 파일을 갖고 MIB 파일을 생성하고 상기 객체 정보 처리부(402)에서 필요한 객체 식별자 정보 파일을 생성하여 상기 객체 식별자 정보 저장부에 저장하는 역할을 한다.

상기 SNMP 인터페이스 헤더 파일(506)은 상기 응용 프로그램(504)에서 관리하는 정보를 얻기 위해, 메시지 ID, 포트번호, 메시지 구조 등이 정의되어있는 파일로 각 응용프로그램(504)에서 정의되어 헤더 파일 저장부(406)에 저장된다.

하기 도 6 및 도 7을 참조하여 상기 SNMP 에이전트(102)에서 컴파일 타임시 행해지는 동작에 대하여 설명하기로 하겠다.

또한, 응용프로그램(504)에 전달되는 GMS 메시지에 대한 정보를 갖고 있는 객체 식별자 정보(OIDInfo) 데이터 파일을 만들기 위한 기본 구조와 이 구조를 이용한 객체 식별자 정보 데이터 생성과정에 대해서 설명을 하겠다.

상기 설명된 방법을 이용하여 상기 추출부(404)는 헤더 파일 저장부(406)로부터 데이터를 생성하는 역할을 수행하며, 상기 에이전트(400)는 객체 식별자 정보 파일을 이용하여 필요한 정보를 탐색할 수 있는 OIDInfo 라이브러리를 이용하여 응용프로그램(504)에게 전달되는 GMS 메시지의 정보를 이용한다.

그러면, 상기 도 6을 참조하여 상기 SNMP 에이전트(102)의 컴파일 동작을 보다 상세히 설명하기로 한다. 상기 도 6에서 컴파일 시 추출부(404)에 의해서 데이터 파일을 생성하는 과정을 보여준다.

먼저, 상기 SNMP 에이전트(102)의 개발자가 응용프로그램(504)을 통해 관리해야할 정보 등을 정의한 SNMP 인터페이스 헤더 파일(506)을 헤더 파일 저장부(406)에 정의한다. 그리고, 상기 추출부(404)가 상기 관리해야할 정보를 갖고 있는 상기 헤더 파일 저장부(406)에서 상기 SNMP 인터페이스 헤더 파일(506)을 읽어서 관리 항목이 정의된 MIB 파일을 만들어 MIB 파일 저장부(502)에 저장한다. 또한, 상기 추출부(404)는 상기 헤더 파일 저장부(406)에서 상기 SNMP 인터페이스 헤더 파일(506)을 읽어서 GMS 메시지를 생성하기 위한 OIDInfo 데이터 파일을 만들어 객체 식별자 정보 저장부(402)에 저장한다.

도 7은 본 발명의 실시 예에 따른 컴파일 타임 시 SNMP 에이전트(102)내의 동작 흐름도이다.

700단계에서 상기 SNMP 에이전트(102)의 관리 항목의 변경이 요구되었다면, 702단계로 진행하여 상기 개발자가 입력한 정보를 근거로 응용프로그램(504)은 변경된 관리 항목에 따라 SNMP 인터페이스 헤더 파일(506)을 작성하여 헤더 파일 저장부(406)에 저장한다.

704단계에서 상기 추출부(404)가 상기 SNMP 인터페이스 헤더 파일을 읽어와서 MIB와 OID 정보 데이터 파일을 작성한다. 상기 704단계에서 작성된 MIB와 OID 정보 파일을 706단계에서 상기 MIB은 MIB 파일 저장부(502)에 상기 작성된 OID 정보 데이터 파일은 객체 식별자 정보 저장부(500)에 저장한다.

도 8은 상기 SNMP 에이전트가 런 타임 시 상기 도 6의 컴파일 시에 생성된 OIDInfo 데이터 파일을 이용하여 수행하는 과정을 보여준다.

상기 도 8의 에이전트(400)와 응용프로그램(504)간의 통신은 GMS(General Message Service : 객체 지향 메시지 서비스)인터페이스를 이용하며 상기 GMS 메시지의 페이로드(Payload)(1006)에는 에이전트(400)와 응용프로그램(504)간의 SNMP 통신을 지원하기 위한 메시지의 타입과 메시지의 페이로드에 반복되는 스트럭처의 개수를 나타내는 EM_Interface_header(1002)가 존재하며, 하기에서 설명하기로 하겠다.

먼저, 상기 매니저(100)가 상기 에이전트(400)에게 SNMP 메시지를 전송하면, 상기 에이전트(400)는 상기 SNMP 메시지 내의 OID 정보를 객체 식별자 정보 처리부(402)로 전송한다. 상기 에이전트(400)가 OID 정보를 상기 객체 식별자 정보 처리부(402)로 전송하는 이유는, 상기 에이전트(400)가 상기 응용 프로그램(504)과 통신을 수행하기 위해선 GMS 정보가 필요하기 때문이다.

상기 객체 식별자 정보 처리부(402)는 수신된 상기 OID 메시지를 근거로 객체 식별자 정보 처리부(500)로부터 GMS 정보를 읽어서 상기 에이전트(400)로 전송한다. 그러면, 상기 에이전트(400)는 상기 GMS 정보를 근거로 상기 응용프로그램(504)과 통신을 수행하고, 상기 응용프로그램(504)가 응답한 결과를 근거로 상기 매니저(100)에게 상기 SNMP 요청에 대한 응답을 한다.

그러면, 이하 도 9를 참조하여 상기 SNMP 에이전트(102)내부 블록들의 런 타임시 동작에 대해 살펴보기로 하겠다.

도 9는 본 발명의 실시 예에 따른 런 타임시 SNMP 에이전트(102)내의 동작 흐름도이다.

900단계에서 에이전트(400)는 상기 매니저(100)로부터 SNMP 요청이 있는지 여부를 검사한다. 상기 900단계에서 요청이 있다면 902단계로 진행하여 상기 에이전트(400)는 상기 매니저(100)로부터 수신된 SNMP 메시지에 포함된 OID를 객체 식별자 정보 처리부(402)에 전달한다.

904단계에서 상기 객체 식별자 정보 처리부(402)는 수신된 상기 OID 정보를 근거로 해당되는 GMS 정보를 상기 객체 식별자 정보 저장부(500)에서 읽어서 상기 에이전트(400)에게 전달한다.

906단계에서 상기 GMS 정보를 수신한 상기 에이전트(400)는 상기 GMS 정보를 근거로 상기 응용프로그램(504)에게 GMS를 요청한다.

908단계에서 상기 응용프로그램(504)은 상기 에이전트(400)가 요구한 GMS 요청에 대한 응답을 하며, 910단계에서 상기 에이전트(400)는 상기 응용프로그램(504)이 응답한 GMS 정보를 근거로 상기 매니저(400)에게 SNMP 응답 메시지를 전송한다.

도 10은 상기 에이전트(400)와 상기 응용프로그램(504)간의 통신을 위한 GMS PDU 구성을 도시한 도면이다. GMS 헤더(1000)는 SNMP를 사용하는 모든 소프트웨어간의 통신을 하기 위해 필요한 필드이다. GMS 페이로드(1006)에는 상기 에이전트(400)와 상기 응용프로그램(504)간의 SNMP 통신을 지원하기 위한 특별한 헤더 파일인 EM_Interface_header(1002)는 4개의 필드(하기 도 11에서 설명하기로 함)로 구성되어 있으며, 상기 에이전트(400)와 상기 응용프로그램(404)간의 SNMP를 지원하기 위해서 사용된다. 스트럭쳐(Structure)(1004)는 SNMP 통신시 필요한 필드 값이다.

도 11내지 도 14는 상기 에이전트(400)와 상기 응용프로그램(504)간의 메시지 종류에 따른 PDU 메시지 구성 및 구성 필드들을 설명한 것으로 하기에서 설명하기로 하겠다.

먼저, 설명에 앞서서 각각의 필드들의 구성 및 역할들에 대해 간략히 살펴보기로 하겠다.

GMS 헤더(GMS Hdr)(1000)와 EM_Interface_header(1002)와 스트럭쳐(Structure)(1004)를 한 개의 열(row)(1014)이라고 칭한다.

상기 EM_Interface_header(1002)는 상술한 바대로 4개의 필드로 구성되어 있으며 하기에서 상세히 설명하기로 하겠다. 먼저, msgType(1006)은 상기 에이전트(102)와 상기 응용프로그램(504)간의 메시지 타입을 나타내고, rowCount(1008)는 멀티 로 테이블(multiRowTable)을 지원하는 테이블에 대해서 Get, Set, Get-Next 할때 Payload에 반복되는 스트럭쳐(1004)의 개수를 나타낸다. 또한, 모든 테이블에 대해서 Get-Bulk를 할 때 SNMP Get-Bulk의 Max-Repeat 값과 같은 용도로 사용된다. response(1010)는 발생할 수 있는 에러를 전달하는데 사용되며, structId(1012)는 응용프로그램(504) 내부에서 추가적으로 사용하는 메시지 ID를 나타낸다. 또한, PLD(Programable Loading Data)를 위해서는 Relation ID로 사용한다.

추가적으로, 상기 에이전트(400)가 상기 응용프로그램(504)로부터 받은 response(1010)를 정상적으로 동작시키기 위해서 상기 응용프로그램(504)은 상기 에이전트(400)로부터 받은 GMS 메시지의 헤더에서 transactionId(1016)와 bsmId(1018)를 복사하여 응답 메시지를 만든 후에 상기 에이전트(400)에게 전달한다.

도 11은 상기 에이전트(400)가 Get/GetNext/GetFirst Request 메시지를 상기 응용프로그램(504)에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면이다.

상기 에이전트(400)가 하나의 열(row)에 해당하는 메모리를 할당하고 상기 매니저(100)로부터 받은 인덱스 값을 GMS 페이로드(Payload)(1006)에 넣은 후에 응용프로그램(504)에게 전달한다.

상기 응용프로그램(504)은 해당하는 테이블에서 상기 에이전트(400)로부터 받은 인덱스에 해당하는 열을 상기 메모리에 복사하고 상기 에이전트(400)에게 응답을 보낸다. 이때 MsgId는 미리 Structure를 정의하였을 때의 RespondId로 설정한다. 만약에 Fail이 발생했을 때는 EM_Interface_header(1002)의 response(1010)에 해당하는 에러 값을 넣어서 전달한다. 이때 GMS 페이로드(1006)를 그대로 다시 보내게 된다.

GetNextRequest 일 경우는 해당하는 열의 다음 열을 찾아서 전달하면 된다.

도 12는 상기 에이전트(400)가 PreSet/SetRequest 메시지를 상기 응용프로그램(504)에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면이다.

먼저, SetRequset 메시지는 상기 에이전트(400)가 상기 응용프로그램(504)에게 PreSetRequset 메시지를 보내게 되는데 상기 PreSetRequest은 상기 GetRequset과 같은 동작을 수행하게 되며, 상기 PresetRequest을 받은 응용프로그램(504)은 해당 열을 고정하게 된다. 상기 응용프로그램(504)으로부터 PreSetRequest의 응답을 받은 에이전트(400)는 SetRequest을 수행할 어트리뷰트(Attribute)의 값을 바꿔서 다시 응용프로그램(504)으로 SetRequest을 보내게 된다.

상기 응용프로그램(504)으로부터 정상적인 응답을 받으면 상기 에이전트(400)는 상기 매니저(100)로 Set 메시지에 대한 응답을 보내게 된다. 만약에 상기 응용프로그램(504)으로부터 Fail을 받으면 상기 에이전트(400)도 상기 매니저(100)로 Fail을 알리고 상기 에이전트(400)가 시간 초과가 발생하면, 이 Set 패킷을 폐기한다.

SetRequest일 경우에는 상기 에이전트(400)가 요청하는 GMS 페이로드(1006)에는 열에 해당하는 메모리가 할당되어서 전달되지만 응용프로그램이 응답을 보낼 때는 EM_Interface_header(1002)만을 보낸다.

도 13은 상기 에이전트(400)가 여러개의 객체(Objest)값을 읽기 위한 GetBulkRequest 메시지를 상기 응용프로그램(504)에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면이다.

상기 GetBulkRequest일 경우, 해당하는 열로부터 EM_Interface_header(1002)의 rowCount(Maxrepeat)(1008)의 숫자만큼을 메모리에 생성해서 전달하며, 만약에 rowCount가 "0"이면 전체 테이블을 전달한다.

또한 rowCount(1008)가 "3"으로 상기 응용프로그램(504)에게 요청되었다면, 요청하는 GMS의 페이로드(1006)는 시작을 나타내는 Index를 보내기 위해서 하나의 Row에 해당하는 메모리만이 할당된다. 상기 응용프로그램(504)은 이 메모리를 삭제하고 해당하는 rowCount(1008)의 숫자만큼의 메모리를 할당하고 정보를 복사해서 상기 에이전트(400)로 보내게 된다. 이 메모리에 대해서는 상기 에이전트(400)가 삭제한다.

도 14는 상기 에이전트(400)가 Notification(Trap) 메시지를 상기 응용프로그램(504)에게 전송할 경우 그 절차와 그 응답에 관한 동작을 도시한 도면이다.

노티피케이션(Notification)을 처리하기 위해서는 미리 약속된 스트럭쳐(Structure)가 정의가 되어 있어야 하며, 유일한 Message Id를 갖는다. 응용프로그램(504)은 EM_Interface_header(1002)의 msgType을 EM_NOTIFICATION로 설정하고 미리 약속되어진 MsgId(1400)의 값을 채운 후에 페이로드에 Structure를 채워서 에이전트(400)에게 전달한다.

이하에서 객체 식별자 정보 저장부(500)에 저장되는 개체 식별자 정보 데이터 파일에 대해 알아보기로 하겠다.

도 15a 와 도 15b는 객체 식별자 정보(OIDInfo) 데이터 파일을 상기 객체 식별자 저장부(500)의 도시되지 않은 메모리에 매핑한 모습을 대략적으로 나타낸 것이다.

도 15a는 상기 객체 식별자 정보 파일을 구성하는 항목들을 개략적으로 도시한 도면이며, 각각의 항목들에 대해서 하기의 도 15b를 참조하여 설명하기로 하겠다.

도 15b는 상기 도 15a에 도시된 각 항목들을 구성하는 정보를 나타내는 테이블을 도시한 도면이다.

먼저, 객체 식별자 정보 헤더(OIDInfoHdr)(1500)은 객체 식별자 정보(OIDInfo)의 가장 기초적인 정보를 저장하는 스트럭쳐(Structure) 이다. 이 스트럭쳐에는 상기 객체 식별자 정보의 버전과 생성 날짜, 디폴트(Default)로 처리하는 객체 식별자(OID), 그리고 객체 식별자 트리(OID Tree)의 각 노드를 나타내는 스트럭쳐인 OIDTreeInfo(1502)의 가장 상위 노드를 가리키는 오프셋(Offset) 값과 메시지(Message) ID를 이용하여 노티피케이션(Notification)의 정보를 찾기 위해서 사용되는 NotiInfoHdr(1508)를 가리키는 오프셋(Offset) 값을 포함하고 있다.

상기 OIDInfoHdr(1500)를 구성하는 각 필드에 대한 자세한 설명은 하기의 <표 1>고 같다

[표 1]

필드 이름	설명
버전(Version)	객체 식별자 정보(OIDInfo) 파일의 버전을 나타낸다.
날짜(Date)	객체 식별자 정보(OIDInfo) 파일의 생성날짜를 나타낸다.
DefaultOID	모든 객체 식별자(OID)가 기본적으로 포함되는 Default OID를 나타낸다. 객체 식별자 정보 내부에서 탐색 할 때 이 부분을 제외하기 위한 목적에서 사용된다.
OIDTreeInfoOffset	객체 식별자 트리 정보(OIDTreeInfo)의 가장 상위 객체 식별자 트리 정보를 가리키는 오프셋(Offset)값이다.
NotiInfoHdrOffset	메시지 ID를 이용해서 노티피케이션(Notification) 정보를 찾기위해서 사용되는 NotiInfoHdr 중에서 가장 상위의 NotiInfoHdr를 가리키는 오프셋 값이다.

다음으로, 객체 식별자 트리 정보(OIDTreeInfo)(1502)는 트리의 노드 정보를 갖고 있으며 트리 탐색을 위해서 사용된다. OIDTreeInfo(1502)는 그룹(Group)에 대한 표현은 물론이고 스칼라 객체(Scalar Object)와 테이블 객체(Table Object)까지 OIDTreeInfo(1502)의 하나의 노드로 표현된다. 이 스트럭쳐는 MIB의 객체 식별자를 나타내는 ObjectID와 상기 노드의 타입(Type), 그리고 네 개의 오프셋(Offset)으로 구성되어 있다.

상기 ObjectID는 전체 OID를 나타내는 것이 아니라 MIB에서 현재 노드의 객체 식별자이다. 노드 타입(NodeType)은 현재의 노드가 그룹 오브젝트(Group Object)를 나타내는지 스칼라 오브젝트(Scalar Object)를 나타내는지 테이블 오브젝트(Table Object)를 나타내는지 구분하기 위해 사용된다. 그리고, 상기 OIDTreeInfo 에 포함되어 있는 오프셋은 상위 OIDTreeInfo를 가리키는 오프셋과 GMS wjdqhdhk 스트럭쳐에 대한 정보를 갖고 있는 GMSInfo를 가리키는 Offset으로 구성되어 있다.

상기 OIDTreeInfo(1502)를 구성하고 있는 각 필드에 대한 설명은 하기의 <표 2>와 같다.

[표 2]

필드 이름	설명
objectId	해당 트리 노드의 Object ID를 나타낸다. 이 값은 전체 OID 스트링(String)을 나타내는 것이 아니라 해당 노드의 객체 식별자를 나타내는 인티저(Integer)이다.
nodeType	트리 노드가 스칼라, 테이블, 그룹인지를 구분하기 위해 사용한다.
upOIDTreeInfoOffset	트리에서 자신의 상위 OIDTreeInfo를 가리키는 오프셋 값이다.
nextOIDTreeInfoOffset	OID Tree에서 같은 레벨(Level)의 다음 OIDTreeInfo를 가리키는 Offset값이다.
gmsInfoOffset	노드 타입이 테이블이나 스칼라 일 경우에 GMS 정보를 갖고 있는 오프셋 값이다. 만약에 nodeType 이 스칼라나 테이블이고 이 값이 0 인 경우나, nodeType이 그룹이고 이 값이 0 이 아니면, 이것은 추출부에서 객체 식별자 정보 파일을 잘못 생성한 것임을 의미한다.

GMS 정보(GMSInfo)(1504)는 응용프로그램(504)에게 메시지를 전달하기 위한 GMS 헤더 정보와 페이로드 정보를 포함하고 있다. 이 스트럭쳐에는 structType에 따라서 그 의미가 달라지는 필드와 공통적으로 사용되는 필드로 구분할 수 있다. 대략적인 필드의 설명으로는, 객체 식별자(OID)는 이 스트럭쳐를 이용해서 생성된 MIB 테이블이나 스칼라를 포함하는

그룹에 해당하는 OID를 나타내며, 공통적인 default OID(enterprise OID)를 포함하고 있다. structname은 스트럭쳐를 정의할 때 만든 이름을 나타낸다. structType은 6가지 서로 다른 스트럭쳐의 타입을 나타내기 위해서 사용되며, payloadType은 이 스트럭쳐를 이용해서 여러 개의 열(Row)을 Set, Get, Get-Next 등이 가능한지를 나타내는 필드이다.

requestMsgId, responseMsgId, portNumber 는 GMS 헤더 정보를 나타내며, numberOfIndex 는 스트럭쳐의 인덱스(Index)의 개수를 나타낸다. numberOfField는 MIB의 테이블의 어트리뷰트(Attribute) 개수가 아니라 스트럭쳐의 전체 필드의 개수를 나타내며, payloadSize는 GMS 페이로드의 크기를 계산하기 위한 스트럭쳐의 전체 크기를 나타낸다. pldRelationId는 PLD 의 릴레이션(Relation ID)를 나타내며, masterTableOffset은 strucType이 서브-테이블일 경우에 마스터 테이블(Master Table)을 가리키는 Offset이다. nextGMSInfoOffset 는 추후에 사용하기 위해서 남겨놓은 필드이며 지금은 용도가 정해지지 않았다. 마지막으로 firstGMSAttInfoOffset은 스트럭쳐의 각각의 필드의 정보를 나타내는 GMSAttInfo의 첫 번째를 나타내는 Offset이다.

상기 GMSInfo(1504)의 각 필드들은 structType에 따라서 서로 다른 용도로 사용되며, requestMsgID, responseMsgId, portNumber 는 GMS Header 정보로 공통적으로 사용되며, nextGMSInfoOffset와 firstGMSAttInfoOffset도 모든 structType에서 사용되는 용도가 같다. 다음으로 하기의 <표 3>에서 상기 structType에 대한 설명과 상기 structType에 따라 달라지는 각 필드의 용도를 설명하기로 하겠다.

[표 3]

타입(Type)	설명
SType_Scalar (scalar)	스칼라 들을 모아 놓은 스트럭쳐를 나타낸다.
SType_StaticTable (static table)	이 테이블은 응용프로그램에서 데이터를 관리하는 스테틱 테이블을 나타낸다.
SType_AgentDyTable (agent dynamic table)	이 테이블은 에이전트에서 데이터를 관리하는 다이내믹 테이블을 나타낸다.
SType_AppDyTable (application dynamic table)	이 테이블은 응용프로그램에서 데이터를 관리하는 다이내믹 테이블을 나타낸다.
SType_PLD (pld)	PLD 테이블을 나타내며, Get, Get-Next, Get-Bulk 는 RSI를 통해서 에이전트가 데이터를 얻어오며 Set 만을 응용프로그램에게 전송한다.
SType_SubTable (sub table, array)	배열의 사용에 의해서 확장된 GMSInfo를 나타낸다.

이하에서 상기 structType에 따른 GMSInfo(1504) 각 필드의 용도를 설명하기로 하겠다.

먼저, 상기 structType이 스칼라(Scalar)일 경우에 상기 GMS(1504)의 각 필드의 용도는 하기의 <표 4>를 참조하여 설명하기로 하겠다.

[표 4]

필드 이름	설명	Etc
OID	Scalar Node들을 포함하는 Group까지의 OID를 String으로 나타낸다.	
structName	SNMP 인터페이스 Header File에 정의된 Structure의 이름을 String으로 나타낸다.	structType에 관계없이 의미가 같다.
structType	GMSInfo가 어떤 Structure를 표현하는 것인지를 나타내는 Field이다. Sturcture Type은 Scalar가 된다. (structType = SType_Scalar)	
payloadType	이 Structure를 이용해서 여러 개의 Row를 동시에 Set, Get, Get-Next등이 가능한지를 나타내는 Field이다. 이 경우 undo기능을 Application에서 제공하는 경우만 사용이 가능하다. single row, multi row 두 가지 Type이 있다.	structType에 관계없이 의미가 같다.
requestMsgId	Agent가 Application에게 요청을 할 때, GMS Header에 들어가는 Request Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
responseMsgId	Application이 Agent에게 응답을 줄 때, GMS Header에 들어가는 Response Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
portNumber	Application이 Agent으로부터 메시지를 받는 Port Number를 나타낸다.	structType에 관계없이 의미가 같다.
numberOfIndex	이 Structure의 Index의 개수, 이 경우 0가 된다. Scalar들을 모아놓은 Structure인 경우에는 Index가 존재하지 않는다.	

numberOfField	SNMP 인터페이스 Header File에 정의된 Structure의 Field의 개수를 나타낸다. numberOfField 는 index field 의 갯수을 포함한다. 만약에 아래와 같이 structure가 정의가 되어있다면,
	struct {
	int A; // index
	int B; // index
	int C;
	int D[3];
	int E[4];
	} AA;	

	위의 경우 numberOfField는 5 이다.	
payloadSize	Structure의 전체 크기를 나타낸다	
pldRelationId	사용하지 않으며 값은 0이다.	
masterTableOffset	사용하지 않으며 값은 0이다.	
nextGMSInfoOffset	아직 용도가 정해지지 않은 Field이다. 추후에 사용할 예정이다.	structType에 관계없이 의미가 같다.
firstGMSAttInfoOffset	Structure의 각각의 Field의 정보를 나타내는 GMSAttInfo의 첫번째를 나타내는 Offset이다.	structType에 관계없이 의미가 같다.

다음은, 상기 structType이 static-table일 경우에 상기 GMS(1504)의 각 필드의 용도는 하기의 <표 5>를 참조하여 설명하기로 하겠다.

[표 5]

필드 이름	설명	Etc
OID	MIB의 Table Entry까지의 OID를 String으로 나타낸다.	
structName	SNMP 인터페이스 Header File에 정의된 Structure의 이름을 String으로 나타낸다.	structType에 관계없이 의미가 같다.
structType	GMSInfo가 어떤 Structure를 표현하는 것인지를 나타내는 Field이다. Sturcture Type은 Static Table이 된다. (structType = SType_StaticTable)	
payloadType	이 Structure를 이용해서 여러 개의 Row를 동시에 Set, Get, Get-Next등이 가능한지를 나타내는 Field이다. 이 경우 undo기능을 Application에서 제공하는 경우만 사용이 가능하다. single row, multi row 두 가지 Type이 있다.	structType에 관계없이 의미가 같다.
requestMsgId	Agent가 Application에게 요청을 할 때, GMS Header에 들어가는 Request Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
responseMsgId	Application이 Agent에게 응답을 줄 때, GMS Header에 들어가는 Response Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
portNumber	Application이 Agent으로부터 메시지를 받는 Port Number.	structType에 관계없이 의미가 같다.
numberOfIndex	Structure의 Index의 개수를 나타낸다.	
numberOfField	SNMP 인터페이스 Header File에 정의된 Structure의 Field의 개수를 나타낸다. 만약에 이 Structure가 배열을 포함하고 있는 경우, 배열을 나타내는 Field의 개수도 포함된다. (MIB에는 배열을 나타내는 Field는 나타나지 않는다.)	
payloadSize	Structure의 전체 크기를 나타낸다.	
pldRelationId	사용하지 않으며 값은 0이다.	
masterTableOffset	사용하지 않으며 값은 0이다.	
nextGMSInfoOffset	아직 용도가 정해지지 않은 Field이다. 추후에 사용할 예정이다.	structType에 관계없이 의미가 같다.
firstGMSAttInfoOffset	Structure의 각각의 Field의 정보를 나타내는 GMSAttInfo의 첫번째를 나타내는 Offset이다.	structType에 관계없이 의미가 같다.

다음은, 상기 structType이 agent-dynamic-table일 경우에 상기 GMS(1504)의 각 필드의 용도는 하기의 <표 6>를 참조하여 설명하기로 하겠다.

[표 6]

필드 이름	설명	Etc
OID	MIB의 Table Entry까지의 OID를 String으로 나타낸다.	
structName	SNMP 인터페이스 Header File에 정의된 Structure의 이름을 String으로 나타낸다.	structType에 관계없이 의미가 같다.
structType	GMSInfo가 어떤 Structure를 표현하는 것인지를 나타내는 Field이다. Sturcture Type은 Agent Dynamic Table이 된다. (structType = SType_AgentDyTable)	
payloadType	이 Structure를 이용해서 여러 개의 Row를 동시에 Set, Get, Get-Next등이 가능한지를 나타내는 Field이다. 이 경우 undo기능을 Application에서 제공하는 경우만 사용이 가능하다. single row, multi row 두 가지 Type이 있다.	structType에 관계없이 의미가 같다.
requestMsgId	Agent가 Application에게 요청을 할 때, GMS Header에 들어가는 Request Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
responseMsgId	Application이 Agent에게 응답을 줄 때, GMS Header에 들어가는 Response Message ID 를 나타낸다.	structType에 관계없이 의미가 같다.
portNumber	Application이 Agent으로부터 메시지를 받는 Port Number 를 나타낸다.	structType에 관계없이 의미가 같다.
numberOfIndex	Structure의 Index의 개수	
numberOfField	SNMP 인터페이스 Header File에 정의된 Structure의 Field의 개수를 나타낸다. RowStatus에 해당하는 Field는 포함하지 않는다.(MIB에는 RowStatus가 존재하지만 Structure를 정의할 때는 RowStatus에 해당하는 Field를 정의하지 않으며 OIDInfo의 GMSAttInfo도 존재하지 않는다. RowStatus관리는 Agent에서 한다.)	
payloadSize	Structure의 전체 크기를 나타낸다.	
pldRelationId	사용하지 않으며 값은 0이다.	
masterTableOffset	사용하지 않으며 값은 0이다.	
nextGMSInfoOffset	아직 용도가 정해지지 않은 Field이다. 추후에 사용할 예정이다.	structType에 관계없이 의미가 같다.
firstGMSAttInfoOffset	Structure의 각각의 Field의 정보를 나타내는 GMSAttInfo의 첫번째를 나타내는 Offset이다.	structType에 관계없이 의미가 같다.

다음은, 상기 structType이 application-dynamic-table일 경우에 상기 GMS(1504)의 각 필드의 용도는 하기의 <표 7>를 참조하여 설명하기로 하겠다.

[표 7]

필드 이름	설명	Etc
OID	MIB의 Table Entry까지의 OID를 String으로 나타낸다.	
structName	SNMP 인터페이스 Header File에 정의된 Structure의 이름을 String으로 나타낸다.	structType에 관계없이 의미가 같다.
structType	GMSInfo가 어떤 Structure를 표현하는 것인지를 나타내는 Field이다. structType은 Application Dynamic Table이다.	
payloadType	이 Structure를 이용해서 여러 개의 Row를 동시에 Set, Get, Get-Next등이 가능한지를 나타내는 Field이다. 이 경우 undo기능을 Application에서 제공하는 경우만 사용이 가능하다. single row, multi row 두 가지 Type이 있다.	structType에 관계없이 의미가 같다.
requestMsgId	Agent가 Application에게 요청을 할 때, GMS Header에 들어가는 Request Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
responseMsgId	Application이 Agent에게 응답을 줄 때, GMS Header에 들어가는 Response Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
portNumber	Application이 Agent으로부터 메시지를 받는 Port Number를 나타낸다.	structType에 관계없이 의미가 같다.
numberOfIndex	Structure의 Index의 개수를 나타낸다.	
numberOfField	SNMP 인터페이스 Header File에 정의된 Structure의 Field의 개수를 나타낸다. RowStatus에 해당하는 Field가 Sturcture에 포함이 되어있으므로 application-dynamic-table인 경우에는 Field의 개수에 포함되며 OIDInfo의 GMSAttInfo도 존재한다. (RowStatus관리는 Application에서 한다.)	
payloadSize	Structure의 전체 크기를 나타낸다.	
pldRelationId	사용하지 않으며 값은 0이다.	
masterTableOffset	사용하지 않으며 값은 0이다.	
nextGMSInfoOffset	아직 용도가 정해지지 않은 Field이다. 추후에 사용할 예정이다.	structType에 관계없이 의미가 같다.
firstGMSAttInfoOffset	Structure의 각각의 Field의 정보를 나타내는 GMSAttInfo의 첫번째를 나타내는 Offset이다.	structType에 관계없이 의미가 같다.

다음은, 상기 structType이 pld일 경우에 상기 GMS(1504)의 각 필드의 용도는 하기의 <표 8>를 참조하여 설명하기로 하겠다.

[표 8]

필드 이름	설명	Etc
OID	MIB의 Table Entry까지의 OID를 String으로 나타낸다.	
structName	SNMP 인터페이스 Header File에 정의된 Structure의 이름을 String으로 나타낸다.	structType에 관계없이 의미가 같다.
structType	GMSInfo가 어떤 Structure를 표현하는 것인지를 나타내는 Field이다. structType은 PLD이다.	
payloadType	이 Structure를 이용해서 여러 개의 Row를 동시에 Set, Get, Get-Next등이 가능한지를 나타내는 Field이다. 이 경우 undo기능을 Application에서 제공하는 경우만 사용이 가능하다. single row, multi row 두 가지 Type이 있다.	structType에 관계없이 의미가 같다.
requestMsgId	Agent가 Application에게 요청을 할 때, GMS Header에 들어가는 Request Message ID를 나타낸다..	structType에 관계없이 의미가 같다.
responseMsgId	Application가 Agent에게 응답을 줄 때, GMS Header에 들어가는 Response Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
portNumber	Application이 Agent으로부터 메시지를 받는 Port Number를 나타낸다.	structType에 관계없이 의미가 같다.
numberOfIndex	Structure의 Index의 개수를 나타낸다.	
numberOfField	Structure의 Field의 개수를 나타낸다. 만약에 이 Structure가 배열을 포함하고 있는 경우, 배열을 나타내는 Field의 개수도 포함된다. (MIB에는 배열을 나타내는 Field는 나타나지 않는다.) 3.3절의 예를 보면, structure의 field는 3개 이므로 numberOfField는 3이다.	
payloadSize	Structure의 전체 크기를 나타낸다.	
pldRelationId	PLD의 Relation ID를 나타낸다.	
masterTableOffset	사용하지 않으며 값은 0이다.	
nextGMSInfoOffset	아직 용도가 정해지지 않은 Field이다. 추후에 사용할 예정이다.	structType에 관계없이 의미가 같다.
FirstGMSAttInfoOffset	Structure의 각각의 Field의 정보를 나타내는 GMSAttInfo의 첫번째를 나타내는 Offset이다.	structType에 관계없이 의미가 같다.

다음은, 상기 structType이 sub-table일 경우에 상기 GMS(1504)의 각 필드의 용도는 하기의 <표 9>를 참조하여 설명하기로 하겠다.

[표 9]

필드 이름	설명	Etc
OID	MIB의 Table Entry까지의 OID를 String으로 나타낸다.	
structName	SNMP 인터페이스 Header File에 정의된 Structure의 이름을 String으로 나타낸다.	structType에 관계없이 의미가 같다.
structType	GMSInfo가 어떤 Structure를 표현하는 것인지를 나타내는 Field이다. structType은 scalar, static-table,agent-dynamic-table,application-dyanmic-table, pld, sub-table 의 6가지가 존재한다.	
payloadType	이 Structure를 이용해서 여러 개의 Row를 동시에 Set, Get, Get-Next등이 가능한지를 나타내는 Field이다. 이 경우 undo기능을 Application에서 제공하는 경우만 사용이 가능하다. single row, multi row 두 가지 Type이 있다.	structType에 관계없이 의미가 같다.
requestMsgId	Agent가 Application에게 요청을 할 때, GMS Header에 들어가는 Request Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
responseMsgId	Application가 Agent에게 응답을 줄 때, GMS Header에 들어가는 Response Message ID를 나타낸다.	structType에 관계없이 의미가 같다.
portNumber	Application이 Agent으로부터 메시지를 받는 Port Number를 나타낸다.	structType에 관계없이 의미가 같다.
numberOfIndex	MIB에 나타나는 Master Table의 Index를 제외하고 Sub Table의 Index의 개수를 나타낸다. 또한 이 값은 배열의 차수를 나타낸다.	
numberOfField	이 경우 배열의 사용에 의해서 확장된 경우이므로 확장된 Table의 Index와 Value를 나타내는 Field의 전체 개수를 나타낸다. 3.3절의 예를 참고하면 이 값은 2이다.	
payloadSize	배열을 정의한 Type의 Size를 나타낸다.	
pldRelationId	사용하지 않으며 값은 0이다.	
masterTableOffset	이 Sub Table이 나타내는 배열을 포함하고 있는 원래의 Master Table을 가리킨다.	
nextGMSInfoOffset	아직 용도가 정해지지 않은 Field이다. 추후에 사용할 예정이다.	structType에 관계없이 의미가 같다.
firstGMSAttInfoOffset	Structure의 각각의 Field의 정보를 나타내는 GMSAttInfo의 첫번째를 나타내는 Offset이다.	structType에 관계없이 의미가 같다.

페이로드(payload)에 들어가는 각각의 필드에 대한 정보는 firstGMSAttInfoOffset 이 가리키는 GMSAttInfo들을 순서대로 찾아가면 알 수 있다.

지금까지 GMSInfo(1504)에 대한 각각의 필드와 스트럭쳐들의 구조를 살펴보았다. 이제는 GMSAttInfo(1506) 스트럭쳐에 대해 살펴보기로 하겠다. 상기 GMSAssInfo(1506)는 GMS의 페이로드에 들어가는 스트럭쳐의 각각의 필드의 정보를 나타내는 스트럭쳐이다. 스트럭쳐 각가의 필드를 나타내는 GMSAttInfo(1506)는 오프셋으로 서로 연결이 되어 있으며 마지막 GMSAttInfo의 오프셋 값은 "0"이다.

상기 GMSAttInfo(1506)의 순서는 인덱스를 나타내는 GMSAttInfo가 먼저 순서대로 나오며, 만약에 이 스트럭쳐가 배열을 포함하고 있는 경우에는 배열을 나타내는 GMSAttInfo는 제일 마지막에 나오게 된다.

인덱스들은 Object ID 순서대로 정렬이 되어 있으며, 밸류(Value)들도 Object ID 순서대로 정렬이 되어 있다.
GMSAttInfo 들은 GMSInfo의 structType과 GMSAttInfo의 fieldType에 따라서 각각의 필드의 의미가 다르게 된다. 먼저 GMSAttInfo의 타입을 결정하는 fieldType에 대해서 살펴본 후에 fieldType에 따라서 달라지는 각 Field의 용도를 하기의 <표 10>을 참조하여 살펴보기로 하겠다.

[표 10]

타입	설명	Etc
FType_ScalarValue (scalar value)	이 GMSAttInfo가 Scalar Value를 나타내고 있다.	
FType_TableValue (table value)	이 GMSAttInfo가 Table의 Attribute Value를 나타내고 있다.	
FType_TableAndIndex (table and index)	이 GMSAttInfo가 Table의 Index Value를 나타내고 있다.	
FType_TableAndArray (table and array)	이 GMSAttInfo가 Structure를 정의할 때 배열로 정의된 Field를 나타내고 있다.	
FType_ArrayAndIndex (array and index)	이 GMSAttInfo는 상위의 GMSInfo의 structType이 sub table일 경우에 index field를 나타낸다. 만약에 sub table이 아닌데 array and index Field Type이 나온다면 Extractor가 OIDInfo File를 잘못 구성한 것이다.	

structType 이 sub-table이 아니고 fieldType이 scalar value, table value, table and index일 경우에 상기 GMSAttInfo (1506)의 각각의 필드에 대해 하기의 <표 11>에 나타내었다.

[표 11]

필드 이름	설명	Etc
objectId	이 Scalar 나 Table의 Attribute의 Object ID를 나타낸다.	
attName	SNMP 인터페이스 Header File에 정의된 Structure의 Field name을 String으로 표현한다.	
asnType	이 Field가 MIB에 표현되는 Syntax Type을 나타낸다.	
accessType	이 Field가 MIB에 표현되는 Access Type을 나타낸다.	
fieldType	이 Field Type을 나타낸다. (scalar value, table value, table and index)	
attType	Structure에 정의된 C data type을 나타낸다.	
attSize	Field의 Size를 나타낸다.	
startOffSet	GMS Payload에서 이 Field가 들어가는 Start Offset을 나타낸다. 시작은 Structure가 들어가는 부분부터의 Offset값이다.	
subTableOffset	사용하지 않으며 값은 0이다.	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextAttInfoOffset	다음 GMSAttInfo를 가리키는 Offset	fieldType에 관계없이 의미가 같다.

structType 이 sub-table이 아니고 fieldType이 table and array일 경우에 상기 GMSAttInfo(1506)의 각각의 필드에 대해 하기의 <표 12>에 나타내었다.

[표 12]

필드 이름	설명	Etc
objectId	배열을 나타내는 Table로 확장하였기 때문에 사용하지 않는다. 값은 0으로 한다.	
attName	SNMP 인터페이스 Header File에 정의된 Structure의 Field name을 String으로 표현한다.	
asnType	ASN_Null로 고정한다.	
accessType	ACType_NotAccessible으로 고정한다.	
fieldType	이 Field Type을 나타낸다. (table and array)	
attType	Structure에 정의된 C data type을 나타낸다.	
attSize	Field의 Size를 나타낸다. 이 경우에는 배열을 나타내는 Field의 전체 Size를 나타낸다. (예를 들어서, 이 Field가 int a[10][5]; 으로 정의가 되었다면 attSize = 200이 된다.)	
startOffSet	GMS Payload에서 이 Field가 들어가는 Start Offset.	
subTableOffset	배열로 인해서 확장된 GMSInfo를 가리키는 Offset값이다.	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextAttInfoOffset	다음 GMSAttInfo를 가리키는 Offset	fieldType에 관계없이 의미가 같다.

structType 이 sub-table이 아니고 fieldType이 array and index일 경우에 상기 GMSAttInfo(1506)의 각각의 필드에 대해 하기의 <표 13>에 나타내었다.

[표 13]

필드 이름	설명	Etc
objectId	MIB에 나타나는 Object ID.	
attName	MIB생성할 때 만들어지는 Field name.	
asnType	이 경우 항상 ASN_Unsigned32이다. (0부터 값을 갖는다.)	
accessType	ACType_NotAccessible으로 고정한다.	
fieldType	이 Field Type을 나타낸다. (array and index)	
attType	Structure에 정의된 C data type을 나타낸다. 이 경우는 항상 AttType_Int 이다.	
attSize	항상 4이다.(int의 size는 4이므로)	
startOffSet	사용하지 않으며 값은 0이다.	
subTableOffset	사용하지 않으며 값은 0이다.	
maxDimensionValue	이 Index가 갖을 수 있는 최대값을 나타낸다. 즉, 배열의 크기가 결정된다.	
nextAttInfoOffset	다음 GMSAttInfo를 가리키는 Offset	fieldType에 관계없이 의미가 같다.

structType 이 sub-table이 아니고 fieldType이 table value일 경우에 상기 GMSAttInfo(1506)의 각각의 필드에 대해 하기의 <표 14>에 나타내었다.

[표 14]

필드 이름	설명	Etc
objectId	MIB에 나타나는 Object ID.	
attName	MIB생성할 때 만들어지는 Attribute name을 String으로 나타낸다.	
asnType	이 Field가 MIB에 표현되는 Syntax Type을 나타낸다.	
accessType	이 Field가 MIB에 표현되는 Access Type을 나타낸다.	
fieldType	이 Field Type을 나타낸다. (table value)	
attType	Structure에 정의된 C data type을 나타낸다.	
attSize	Field의 Size를 나타낸다	
startOffSet	GMS Payload에서 Array가 시작하는 Start Offset이다. 즉, structType이 sub-table이 아니고 fieldType이 table and array일 경우에 startOffSet값과 같다.	
subTableOffset	사용하지 않으며 값은 0이다.	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextAttInfoOffset	다음 GMSAttInfo를 가리키는 Offset	

상기 도 15의 NotiinfoHdr(1508)는 메시지 ID를 이용하여 노티피케이션 정보를 얻기 위하여 객체 식별자 정보 처리부(402) 내부에서 사용하는 스트럭쳐이다. 상기 노티피케이션 정보를 찾는 방법은 두 개의 메시지 ID를 이용해서 찾는 방법과 하나의 메시지 ID를 이용해서 찾는 방법 두 가지가 존재한다.

NotiInfoHdr(1508)에는 NotiNodeInfo(1510)의 개수를 나타내는 필드와 상기 NotiNodeInfo(1510)의 첫 번째를 가리키는 오프 셋으로 구성된다. 각 필드에 대한 설명은 하기의 <표 15>를 참조한다.

[표 15]

필드 이름	설명	Etc
numberOfNotiNodeInfo	Message ID별로 순서대로 정렬이 되어있는 NotiNodeInfo들의 개수를 나타낸다. 이 Field를 이용해서 해당하는 Message ID를 갖는 NotiNodeInfo를 찾기 위해서 사용된다.	
firstNotiNodeInfoOffset	순서대로 정렬이 되어있는 NotiNodeInfo의 첫번째를 가리키는 Offset값이다.	

상기 도 15의 NotiNodeInfo(1510)는 메시지 ID를 저장하는 필드와 NotiNodeInfo의 타입을 구분하는 notiNodeType와 서브(Sub) NotiInfoHdr를 가리키는 subNotiInfoHdrOffset, 마지막으로, 노티피케이션 정보를 갖고 있는 NotiInfo(1512)를 가리키는 notiInfoOffset으로 구성되어 있다.

NotiNodeInfo(1510)는 메시지 ID를 이용하여 해당 노티피케이션 정보를 찾기 위해서 사용하는 스트럭쳐이다. 상기 NotiNodeInfo(1510)는 상기 OIDInfo 데이터 파일 저장부(500)에 메시지 ID 순서대로 정렬이 되어 있다. 객체 식별자 정보 처리부(402)는 상기 NotiNodeInfo(1510)와 NotiInfoHdr(1508)을 이용해서 해당하는 노티피케이션 정보를 찾을 수 있다.

첫 번째로, 하나의 메시지 ID가 하나의 노티피케이션 정보와 매핑되어 있는 경우, NotiInfoHdr의 numberOfNotiNodeinfo와 firstNotiInfoOffset를 이용해서 이진 탐색 알고리즘을 수행한다. 그러면 해당하는 NotiNodeInfo(1510)를 찾는다. 이는 NotiNodeInfo(15101)가 메시지 ID순서대로 정렬이 되어있기 때문에 가능하다.

상기 NotiNodeInfo(1510)의 notiInfoOffset이 가리키는 NotiInfo(1512) 정보를 이용하여 SNMP 노티피케이션 메시지를 만든다.

두 번째 경우로 두 개의 메시지 ID가 하나의 노티피케이션 정보와 매핑되어있을 경우, 위의 경우와 같은 방법으로 NotiNodeinfo(1510)를 찾는다. 이 NotiNodeinfo(1510)의 notiNodeType은 멀티 노티피케이션 노드가 될 것이며, subNotiInfoHdrOffset 은 서브(Sub) NotiInfoHdr(1508)를 가리킬 것이다.

그러면, 두 번째 메시지 ID를 이용해서 같은 알고리즘을 반복한다. 그러면 해당하는 NotiNodeInfo(1510)를 찾는다. NotiNodeInfo(1510)의 notiinfoOffset이 가리키는 NotiInfo 정보를 이용하여 SNMP 노티피케이션 메시지를 만든다.

상술한 각 필드에 대한 설명은 하기의 <표 16>를 참조한다.

[표 16]

필드 이름	설명	Etc
notiMsgId	Notification 정보를 찾기 위한 Message ID.	
notiNodeType	NotiNodeInfo의 Type을 나타내며, single notification node, multi notification node, sub notification node 의 3가지 Type이 존재한다. single notification node 는 일반적으로 하나의 Message ID가 하나의 Notification으로 Mapping될 때를 나타낸다. multi notification node 는 두개의 Message ID를 갖고 Notification정보와 Mapping가 되는 경우에 첫번째 Message ID를 이용해서 찾은 NotiNodeInfo의 Type이다. sub notification node 은 두 번째 Message ID를 이용해서 찾은 NotiNodeInfo의 Type이다.	
subNotiInfoHdrOffset	notiNodeInfoType이 multi notification node일 때, Sub NotiInfoHdr을 가리키는 Offset이다. notiNodeInfoType이 single notification node이나 sub notification node이면 이 값은 0이다.	
notiInfoOffset	notiNodeInfoType이 single notification node나 multi notification node일 때, Notification정보를 갖고 있는 NotiInfo를 가리키는 Offset이다. notiNodeInfoType이 multi notification node이면 이 값은 0-이다.	

다음으로, NotiInfo(1512)에 대하여 살펴보기로 한다. 상기 NotiInfo(1512)는 노티피케이션의 타입을 구분하는 notiInfoType, 그리고 SNMP 인터페이스 헤더 파일에 정의된 노티피케이션 스트럭쳐의 필드의 개수를 나타내는 numberOfNotiField와 메시지 필드 정보의 첫 번째 오프 셋으로 구성된다.

먼저, 상기 notiInfoType에 대해 하기의 <표 17>을 참조한다.

[표 17]

타입	설명	Etc
NIType_CommonNoti	일반적이 Notification으로 Agent는 Manager에게 Notification을 전달만 한다.	
NIType_InformNoti	Notification을 보내고 Agent는 Inform Message를 기다린다. 만약에 Inform Message가 Manager로부터 오지 않으면 미리 약속된 처리를 Agent는 수행한다.	

이하에서 상기 NotiInfo(1512)를 구성하는 각 필드에 대해 하기의 <표 18>을 참조한다.

[표 18]

필드 이름	설명	Etc
notiOID	MIB에 정의된 Notification의 OID를 나타낸다.	
notiInfoType	Inform Message를 처리하는 Notification인지 아닌지를 나타낸다.	
numberOfNotiField	SNMP 인터페이스 Header File에 정의된 Notification Structure의 Field 개수를 나타낸다.	
firstNotiAttInfoOffset	SNMP 인터페이스 Header File에 정의된 Notification Structure의 Field 정보를 갖고 있는 NotiAttInfo의 첫번째를 가리키는 Offset이다.	

다음으로, 마지막 스트럭쳐인 NotiAttInfo(1514)에 대해 설명하기로 하겠다. NotiAttinfo(1514)는 노티피케이션 메시지 각각의 필드에 대한 정보를 나타낸다. 먼저 notiAttOID는 각 노티피케이션을 보내는 필드의 OID를 나타내는 스트링으로 스칼라는 full OID를 테이블에 대해서는 인덱스를 제외한 값을 나타낸다. notiAttName은 SNMP 인터페이스 Header File에 정의된 필드의 이름을 스트링으로 나타내며, notiASNType은 이 Attribute이 MIB에 표현되는 신택스 형식(Syntax Type)을 나타낸다. notiFieldType은 이 Attribute이 어떤 타입인지 결정되며, 이 타입에 따라서 다른 필드의 사용 용도도 조금씩 변하게 된다. notiAttType은 이 필드의 C 데이터 형식을 나타내며, notiAttSize는 이 필드의 크기를 나타낸다. notiStartOffSet은 Application으로부터 받은 Structure로부터 이 필드가 위치하는 오프셋(Offset)값이며 subNotiAttInfoOffset는 만약에 현재의 필드가 배열을 나타낼 경우에 배열을 표현하는 NotiAttInfo를 가리키는 오프 셋값이다. notiMaxDimensionValue 는 배열의 사용에 의해서 확장된 NotiAttInfo중에 타입이 array and index일 경우, 이 index의 최대값을 나타내며, 이것은 배열의 최대값을 의미한다. 마지막으로 nextNotiAttInfoOffset는 다음 NotiAttInfo(1514)를 가리키는 오프 셋 값이다. NotiAttInfo(1514)가 들어가 있는 순서는 index가 앞쪽으로 위치한다. 만약에 여러 개의 Table로 구성된 Notification이라면 Table단위로 index다음에 value가 오고 다름 index, value가 위치한다.

상기 NotiAttinfo(1514)의 notiFieldType에 따라 각 필드가 의미하는 내용이 달라지는데, 이는 하기의 표들을 참조하기로 하겠다. 먼저, notiFieldType이 스칼라 밸류(Scalar value)인 경우는 하기의 <표 19>와 같다.

[표 19]

필드 이름	설명	Etc
notiAttOID	이 scalar의 전체 OID String을 나타낸다.	
notiAttName	Structure의 Field name을 나타내는 String을 나타낸다.	notiFieldType에 상관없이 의미가 같다.
notiASNType	이 Field의 Syntax Type 을 나타낸다.	
notiFieldType	이 Field의 Field Type (scalar value) 을 나타낸다.	
notiAttType	이 Field의 C Type 을 나타낸다.	
notiAttSize	이 Field의 Size 을 나타낸다.	
notiStartOffSet	Notification Structure에서 이 Field가 시작하는 Offset값이다.	notiFieldType에 상관없이 의미가 같다.
subNotiAttInfoOffset	각 배열을 나타내는 NotiAttInfo를 가리키는 Offset값	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextNotiAttInfoOffset	다음 Field를 가리키는 Offset값.	notiFieldType에 상관없이 의미가 같다.

다음으로, notiFieldType이 table and index인 경우는 하기의 <표 20>을 참조하겠다.

[표 20]

필드 이름	설명	Etc
notiAttOID	Table의 Attribute까지의 OID String을 나타낸다.	
notiAttName	Structure의 Field name을 나타내는 String을 나타낸다.	notiFieldType에 상관없이 의미가 같다.
notiASNType	이 Field의 Syntax Type 을 나타낸다.	
notiFieldType	이 Field의 Field Type (table and index) 을 나타낸다.	
notiAttType	이 Field의 C Type 을 나타낸다.	
notiAttSize	이 Field의 Size 을 나타낸다.	
notiStartOffSet	Notification Structure에서 이 Field가 시작하는 Offset값이다.	notiFieldType에 상관없이 의미가 같다.
subNotiAttInfoOffset	사용하지 않으며 값은 0이다.	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextNotiAttInfoOffset	다음 Field를 가리키는 Offset값.	notiFieldType에 상관없이 의미가 같다.

notiFieldType이 table value인 경우는 하기의 <표 21>을 참조하겠다.

[표 21]

Field Name	Description	Etc
notiAttOID	Table의 Attribute까지의 OID String을 나타낸다.	
notiAttName	Structure의 Field name을 나타내는 String을 나타낸다.	notiFieldType에 상관없이 의미가 같다.
notiASNType	이 Field의 Syntax Type 을 나타낸다.	
notiFieldType	이 Field의 Field Type (table value) 을 나타낸다.	
notiAttType	이 Field의 C Type 을 나타낸다.	
notiAttSize	이 Field의 Size 을 나타낸다.	
notiStartOffSet	Notification Structure에서 이 Field가 시작하는 Offset값이다.	notiFieldType에 상관없이 의미가 같다.
subNotiAttInfoOffset	사용하지 않으며 값은 0이다.	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextNotiAttInfoOffset	다음 Field를 가리키는 Offset값.	notiFieldType에 상관없이 의미가 같다.

notiFieldType이 table and array인 경우는 하기의 <표 22>을 참조하겠다.

[표 22]

필드 이름	설명	Etc
notiAttOID	사용하지 않으며 값은 empty string이다.	
notiAttName	Structure의 Field name을 나타내는 String을 나타낸다.	notiFieldType에 상관없이 의미가 같다.
notiASNType	이 Field의 Syntax Type 을 나타낸다.	
notiFieldType	이 Field의 Field Type (table and array) 을 나타낸다.	
notiAttType	이 Field의 C Type 을 나타낸다.	
notiAttSize	이 Field의 Size 을 나타낸다.	
notiStartOffSet	Notification Structure에서 이 Field가 시작하는 Offset값이다.	notiFieldType에 상관없이 의미가 같다.
subNotiAttInfoOffset	배열 Value를 나타내는 NotiAttInfo를 가리키는 Offset값을 나타낸다.	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextNotiAttInfoOffset	다음 Field를 가리키는 Offset값.	notiFieldType에 상관없이 의미가 같다.

다음으로, 상기 NotiAttInfo(1514)의 subNotiAttInfoOffset이 가리키는 NotiAttInfo의 notiFieldType이 array and index인 경우는 하기의 <표 23>을 참조하겠다.

[표 23]

필드 이름	설명	Etc
notiAttOID	MIB에 나타나는 Table의 Attribute까지의 OID String을 나타낸다. 이 Table은 배열의 사용에 의해서 확장된 Table이다.	
notiAttName	MIB상에 나타나는 Attribute Name을 나타내는 String이다.	
notiASNType	이 경우 항상 ASN_Unsigned32이다. (0부터 값을 갖는다.)	
notiFieldType	이 Field의 Field Type (table value) 을 나타낸다.	
notiAttType	이 Field의 C Type 을 나타낸다. 이 경우는 항상 AttType_Int 이다.	
notiAttSize	이 Field의 Size 을 나타낸다. 이 경우 항상 4이다.	
notiStartOffSet	사용하지 않으며 값은 0이다.	
subNotiAttInfoOffset	사용하지 않으며 값은 0이다.	
maxDimensionValue	이 Index가 갖을 수 있는 최대값을 나타낸다. 즉, 배열의 크기가 결정된다.	
nextNotiAttInfoOffset	사용하지 않으며 값은 0이다.	

상기 NotiAttInfo(1514)의 subNotiAttInfoOffset이 가리키는 NotiAttInfo의 notiFieldType이 table value인 경우는 하기의 <표 24>을 참조하겠다.

[표 24]

필드 이름	설명	Etc
notiAttOID	MIB에 나타나는 Table의 Attribute까지의 OID String을 나타낸다. 이 Table은 배열의 사용에 의해서 확장된 Table이다.	
notiAttName	MIB상에 나타나는 Attribute Name을 나타내는 String이다.	
notiASNType	이 Field의 Syntax Type 을 나타낸다.	
notiFieldType	이 Field의 Field Type (table value) 을 나타낸다.	
notiAttType	이 Field의 C Type 을 나타낸다.	
notiAttSize	이 Field의 Size 을 나타낸다.	
notiStartOffSet	원래의 Notification Structure에서 배열을 나타내는 Field가 시작하는 Offset값이다. notiFieldType이 table and array인 경우의 notiStartOffSet의 값과 같다.	
subNotiAttInfoOffset	사용하지 않으며 값은 0이다.	
maxDimensionValue	사용하지 않으며 값은 0이다.	
nextNotiAttInfoOffset	사용하지 않으며 값은 0이다.	

도 16은 상기 객체 식별자 정보 저장부(500)의 구조를 대략적으로 묘사해 놓은 것이며 OID를 이용해서 Tree를 탐색하기 위한 부분과 Message ID를 이용해서 Notification 정보를 얻어오기 위한 부분으로 나눌 수가 있다. 상기 MIB에 표현되는 하나의 테이블은 하나의 GMS 정보와 매핑된다. Scalar의 경우에는 같은 Group의 밑에 있는 것들을 묶어서 하나의 GMS 정보와 매핑된다. 상기 Scalar 경우에 Group밑에 있는 것들은 하나의 Structure로 정의가 된다는 것이며, index가 존재하지 않는 Structure라는 의미이다.

그러면, 본 발명의 실시 예에 따라 상기 에이전트(400)와 상기 응용프로그램(504)간의 통신을 하기 위해 GMS 정보를 찾는 방법을 도 17을 참조하여 설명하기로 하겠다.

도 17은 본 발명의 실시 예에 따른 GMS 정보를 찾는 방법을 도시한 도면이다. 매니저(100)로부터 SNMP Get, Get-Next, Get-Bulk, Set등의 요청을 받았을 경우 상기 Get, Get-Next, Get-Bulk, Set 메시지에 포함된 객체 식별자(OID)를 이용하여 해당하는 GMS 정보를 찾기 위해서 객체 식별자 정보 처리부(402)는 다음과 같은 절차에 따르게 된다.

먼저, 에이전트(400)로부터 입력받은 OID String을 Integer Type의 Object ID로 분리, 변경한다. 그 후 Object ID를 갖고 트리 탐색 알고리즘을 이용해서 순차적으로 OIDTreeInfo(1502)를 찾아간다.

그러면, 객체 식별자 정보 저장부(500)는 OID와 일치하는 OIDTreeInfo(1502)를 반환하거나, 상기 OID와 일치하는 OIDTreeInfo(1502)의 N다음 OIDTreeInfo를 반환한다. 만일, 해당하는 OIDTreeInfo를 찾게 되면, 이 OIDTreeInfo이 가리키는 GMSInfo(1504)를 상기 객체 식별자 정보 처리부((402)는 반환하게 된다. 그리고, 상기 에이전트(400)는 GMSInfo(1504)에 있는 정보를 이용하여 GMS Header와 Payload를 만들게 된다. GMS Payload(1006)를 만들 때, Payload의 앞에는 OAM Header가 들어가고 다음에 에이전트(400)와 응용프로그램(504)사이에 SNMP를 지원하기 위한 EM Header(1002)가 들어가고 그 다음에 스트럭쳐의 값이 들어가게 된다. 또 스트럭쳐의 필드에 대한 정보는 GMSAttInfo(1506)들을 이용하여 GMS PDU를 완성해서 응용프로그램(504)에게 Message를 전달한다.

본 발명의 실시 예에 따라 상기 에이전트(400)가 노티피케이션 정보를 찾는 방법을 도 18을 참조하여 설명하기로 하겠다.

도 18은 본 발명의 실시 예에 따른 상기 에이전트(400)가 노티피케이션 정보를 찾는 방법을 도시한 도면이다.

상기 에이전트(400)가 상기 응용프로그램(504)로부터 미리 약속이 되어 있는 메시지 ID로 상기 응용프로그램(504)로부터 Trap Message를 받으면 상기 메시지 ID를 이용해서 상기 식별자 정보 처리부(500)가 SNMP Notification정보를 찾는 과정이다.

먼저, 상기 에이전트(400)는 수신된 메시지 ID의 개수가 한 개인지 두 개인지를 검사한다. 상기 에이전트(400)는 EM Header(1002)의 subMsgId 필드를 검사하여 이 값이 0이 아니면 GMS Header(1000)에 있는 Message ID를 첫 번째 메시지 ID로 subMsgId를 두 번째 메시지 ID로 사용한다.

그 후, 첫 번째 메시지 ID를 이용하여 NotiNodeInfo(1510)를 찾는다. 찾는 방법은 NotiInfoHdr(1508)의 NotiNodeInfo (1510)의 개수와 이진 탐색 알고리즘을 이용하여 NotiNodeInfo(1510)을 찾고 만약에 상기 두 번째 메시지를 갖는 경우 NotiNodeInfo(1510)의 subNotiInfoHdrOffset이 가리키는 NotiInfoHdr(1508)의 Sub NotiNodeInfo의 개수와 상기 이진 탐색 알고리즘을 이용하여 NotiNodeInfo(1510)을 찾는다.

NotiNodeInfo(1510)의 notiInfoOffset은 노티피케이션정보를 저장하고 있는 NotiInfo(1512)를 가리키고 있으며 상기 NotiInfo(1512)와 NotiAttInfo(1514)를 이용하여 SNMP 노티피케이션 메시지를 만들어서 상기 매니저(100)로 보낸다.

발명의 효과

본 발명에 의하면, 관리 대상이 바뀌거나 관리항목이 추가, 변경, 삭제가 되어도 추출부로 새로운 객체 식별자 정보 파일만을 생성하면 SNMP 에이전트는 기능의 추가나 새롭게 코딩, 재 컴파일을 할 필요가 없다. 또한 SNMP 에이전트와 응용프로그램간에 사용하는 스트럭쳐가 추가, 변경, 삭제 되어도 관리항목인 MIB의 변경과 객체 식별자 정보 파일의 변경이 자동으로 가능하다. 이로 인하여 SNMP 에이전트의 개발이 용이하며 여러 가지 다른 장비나 관리 대상에 SNMP 에이전트를 그대로 사용이 가능하다.

(57) 청구의 범위

청구항 1.

간이 망 관리 프로토콜(SNMP)을 사용하여 통신 장비를 관리하기 위해 컴파일 타임시에 상기 통신 장비를 관리하기 위한 관리 정보 베이스(MIB) 파일과 상기 간이 망 관리 프로토콜 에이전트와 응용프로그램 간의 통신을 위한 객체 식별자 정보 파일을 생성하기 위한 장치에 있어서,

상기 간이 망 관리 프로토콜 장비의 관리를 위해 응용프로그램이 작성한 헤더 파일을 저장하는 헤더 파일 저장부와,

상기 헤더 파일 저장부에서 상기 헤더 파일을 읽어서 상기 관리 정보 베이스 파일과 상기 간이 망 관리 프로토콜 에이전트와 상기 응용프로그램간의 메시지 교환을 위한 상기 객체 식별자 정보 파일을 작성하는 추출부와,

상기 추출부에서 작성한 상기 관리 정보 베이스 파일을 저장하는 관리 정보 베이스 파일 저장부와,

상기 추출부에서 작성한 상기 객체 식별자 정보 파일을 저장하는 객체 식별자 정보 저장부를 포함함을 특징으로 하는 간이 망 관리 프로토콜 관리 장치.

청구항 2.

제 1항에 있어서, 상기 메시지는 상기 간이 망 관리 프로토콜 에이전트와 상기 응용프로그램간의 통신을 위한 GMS (General Message Service) 메시지임을 특징으로 하는 간이 망 관리 프로토콜 관리 장치.

청구항 3.

간이 망 관리 프로토콜(SNMP)을 사용하여 통신 장비를 관리하기 위해 컴파일 타임시에 상기 통신 장비를 관리하기 위한 관리 정보 베이스(MIB) 파일과 상기 간이 망 관리 프로토콜 에이전트와 응용프로그램 간의 통신을 위한 객체 식별자 정보 파일을 생성하기 위한 방법에 있어서,

변경된 관리 항목을 입력받은 응용프로그램이 간이 망 관리 프로토콜 인터페이스 헤더 파일을 작성하는 과정과,

상기 작성된 간이 망 관리 프로토콜 인터페이스 헤더 파일을 읽어와서 관리 정보 베이스 파일과 객체 식별자 정보 파일을 생성하는 과정과,

상기 생성된 관리 정보 베이스 파일과 상기 객체 식별자 정보 파일을 저장하는 과정을 포함함을 특징으로 하는 간이 망 관리 프로토콜 관리 방법.

청구항 4.

간이 망 관리 프로토콜을 사용하여 장비를 관리하는 간이 망 관리 프로토콜 에이전트가 런 타임시에 외부에서 입력된 간이 망 관리 프로토콜 요청에 응답하기 위한 장치에 있어서,

외부로부터 간이 망 관리 프로토콜 요청 메시지를 수신하여 상기 메시지 내에 포함된 객체 식별자 정보를 객체 식별자 정보 처리부로 전송하고 망 관리를 위한 응용프로그램과 통신을 수행하여 망 관리 프로토콜 응답 메시지를 출력하는 에이전트와,

상기 에이전트가 전송한 객체 식별자 정보를 근거로 객체 식별자 정보 저장부로부터 해당되는 정보를 읽어와서 상기 정보를 상기 에이전트에게 전달하는 객체 식별자 정보 처리부를 포함함을 특징으로 하는 간이 망 관리 프로토콜 관리 장치.

청구항 5.

제 4항에 있어서, 상기 응용프로그램과 상기 에이전트간의 통신은 GMS(General Message Service)메시지를 사용함을 특징으로 하는 간이 망 관리 프로토콜 관리 장치.

청구항 6.

간이 망 관리 프로토콜을 사용하여 장비를 관리하는 간이 망 관리 프로토콜 에이전트가 런 타임시에 외부에서 입력된 간이 망 관리 프로토콜 요청에 응답하기 위한 방법에 있어서,

외부로부터 간이 망 관리 프로토콜 요청 메시지를 수신하는 과정과,

상기 요청 메시지에 포함된 객체 식별자 정보를 객체 식별자 정보 처리부에 전달하는 과정과,

상기 객체 식별자 정보 처리부가 상기 에이전트가 상기 응용프로그램과 통신하기 위해 필요한 GMS 정보를 객체 식별자 정보 저장부에서 읽어와 상기 에이전트에게 전달하는 과정과,

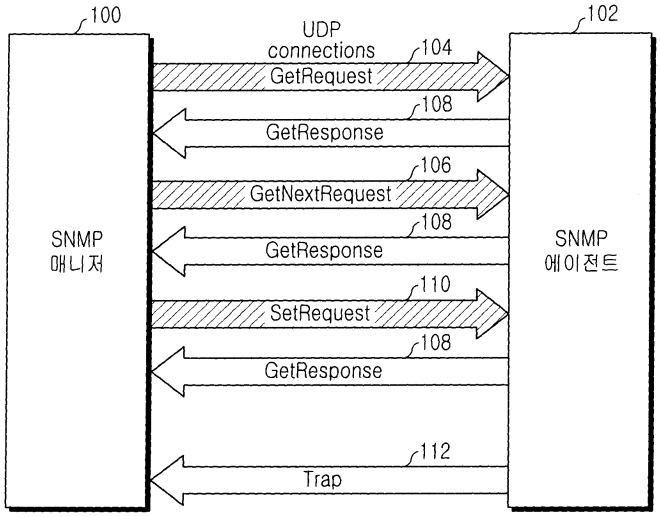
상기 GMS 정보를 근거로 상기 에이전트가 응용프로그램에게 GMS 요청을 하는 과정과,

상기 응용프로그램이 상기 에이전트에게 GMS 응답을 하는 과정과,

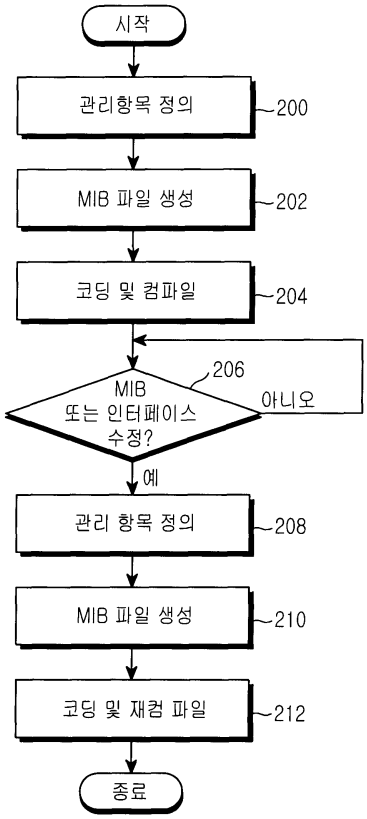
상기 에이전트가 상기 매니저에게 간이 망 관리 프로토콜 요청에 대한 응답 메시지를 출력하는 과정을 포함함을 특징으로 하는 간이 망 관리 프로토콜 관리 방법.

도면

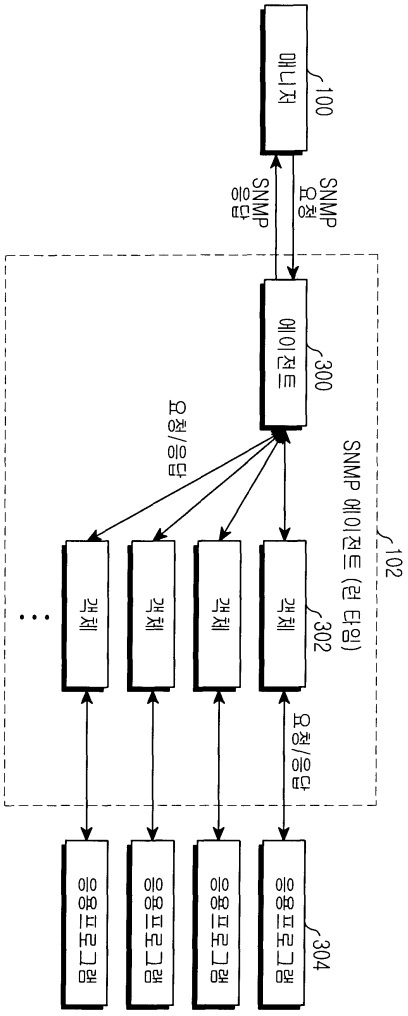
도면1


100
UDP
connections
104
GetRequest
108
GetResponse
106
GetNextRequest
108
GetResponse
110
SetRequest
108
GetResponse
112
Trap
102
SNMP
매니저
SNMP
에이전트

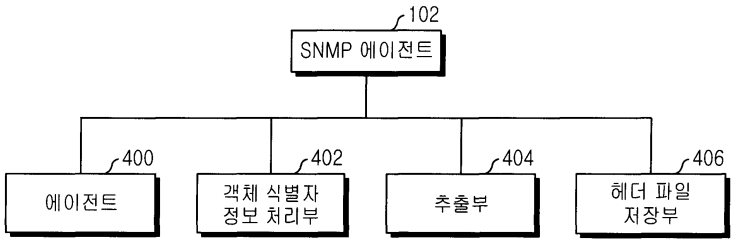
도면2


시작
관리항목 정의
200
MIB 파일 생성
202
코딩 및 컴파일
204
MIB
또는 인터페이스
수정?
206
아니오
예
관리 항목 정의
208
MIB 파일 생성
210
코딩 및 재컴 파일
212
종료

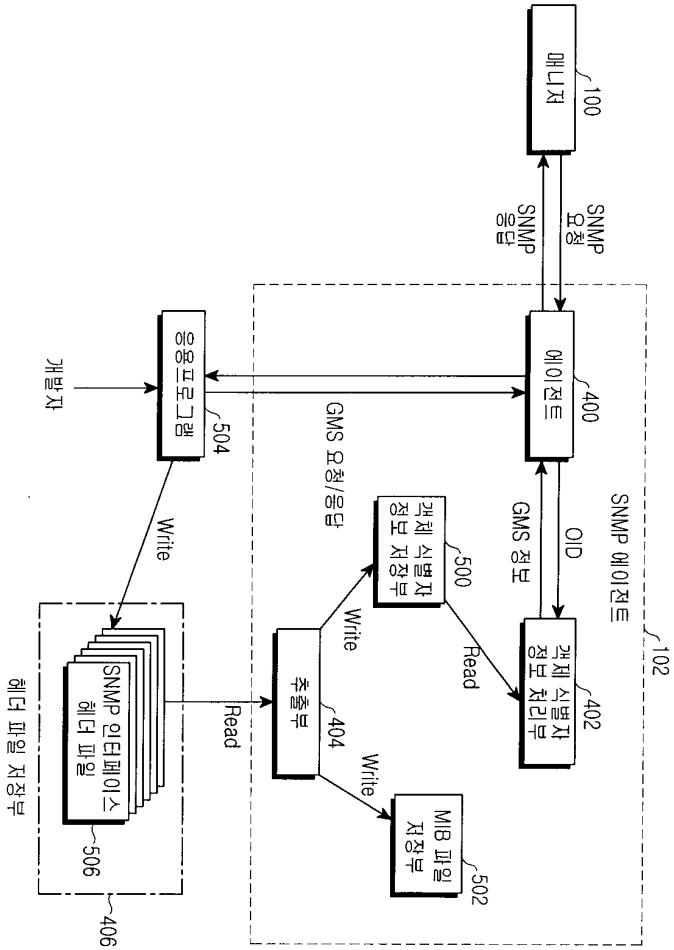
도면3


매니저
100
SNMP 응답
SNMP 요청
에이전트
300
요청/응답
SNMP 에이전트 (런 타임)
102
객체
객체
객체
객체
302
요청/응답
응용프로그램
응용프로그램
응용프로그램
응용프로그램
304

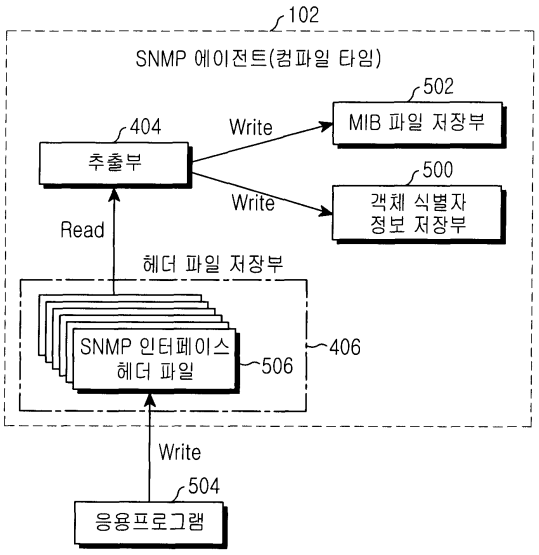
도면4


102
SNMP 에이전트
400
에이전트
402
객체 식별자 정보 처리부
404
추출부
406
헤더 파일 저장부

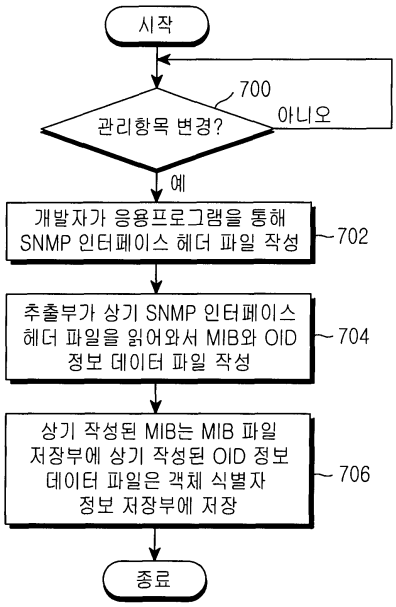
도면5


매니저
100
SNMP 요청
SNMP 응답
SNMP 에이전트
102
에이전트
400
응용프로그램
504
개발자
GMS 요청/응답
GMS 정보
OID
객체 식별자 정보 저장부
500
객체 식별자 정보 처리부
402
Write
Read
추출부
404
MIB 파일 저장부
502
헤더 파일 저장부
SNMP 인터페이스 헤더 파일
506
406

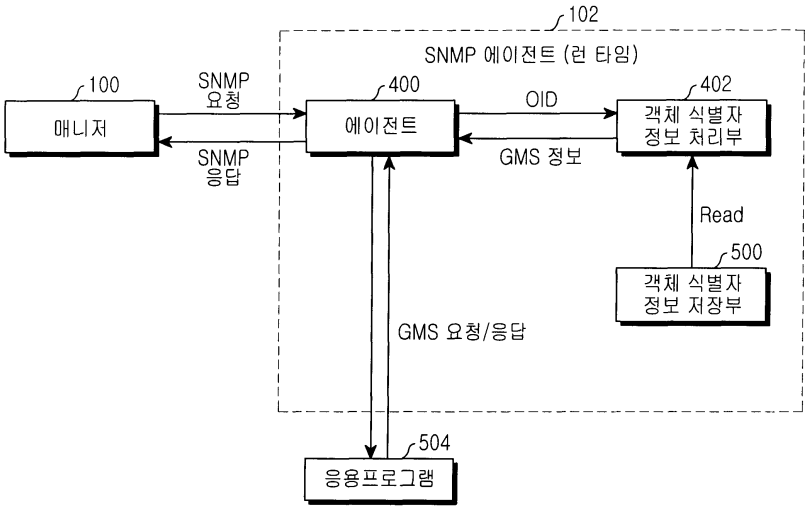
도면6


102
SNMP 에이전트(컴파일 타임)
502
MIB 파일 저장부
404
추출부
Write
500
객체 식별자 정보 저장부
Read
헤더 파일 저장부
SNMP 인터페이스 헤더 파일
506
406
Write
504
응용프로그램

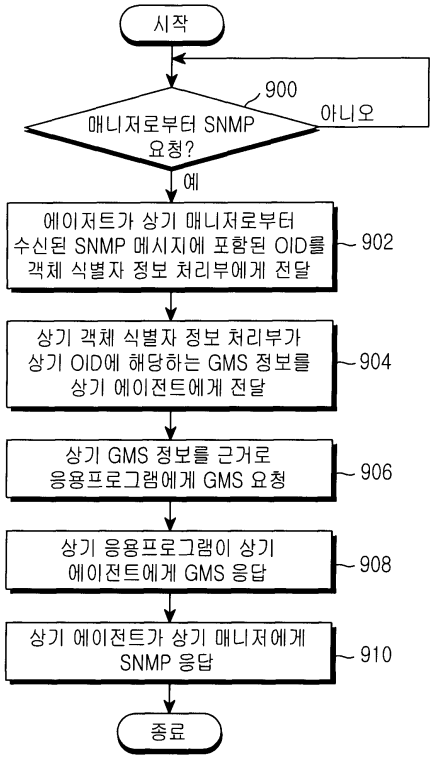
도면7


시작
700
관리항목 변경?
아니오
예
개발자가 응용프로그램을 통해 SNMP 인터페이스 헤더 파일 작성
702
추출부가 상기 SNMP 인터페이스 헤더 파일을 읽어와서 MIB와 OID 정보 데이터 파일 작성
704
상기 작성된 MIB는 MIB 파일 저장부에 상기 작성된 OID 정보 데이터 파일은 객체 식별자 정보 저장부에 저장
706
종료

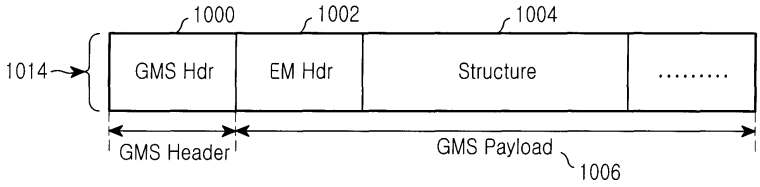
도면8


102
SNMP 에이전트 (런 타임)
100
매니저
SNMP 요청
SNMP 응답
400
에이전트
OID
GMS 정보
402
객체 식별자 정보 처리부
Read
500
객체 식별자 정보 저장부
GMS 요청/응답
504
응용프로그램

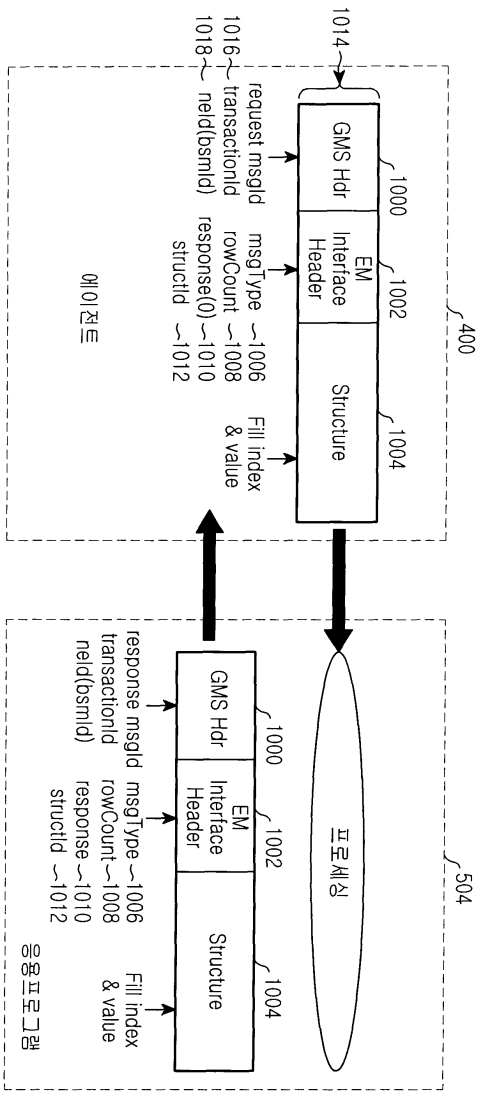
도면9


시작
900
매니저로부터 SNMP 요청?
아니오
예
에이저트가 상기 매니저로부터 수신된 SNMP 메시지에 포함된 OID를 객체 식별자 정보 처리부에게 전달
902
상기 객체 식별자 정보 처리부가 상기 OID에 해당하는 GMS 정보를 상기 에이전트에게 전달
904
상기 GMS 정보를 근거로 응용프로그램에게 GMS 요청
906
상기 응용프로그램이 상기 에이전트에게 GMS 응답
908
상기 에이전트가 상기 매니저에게 SNMP 응답
910
종료

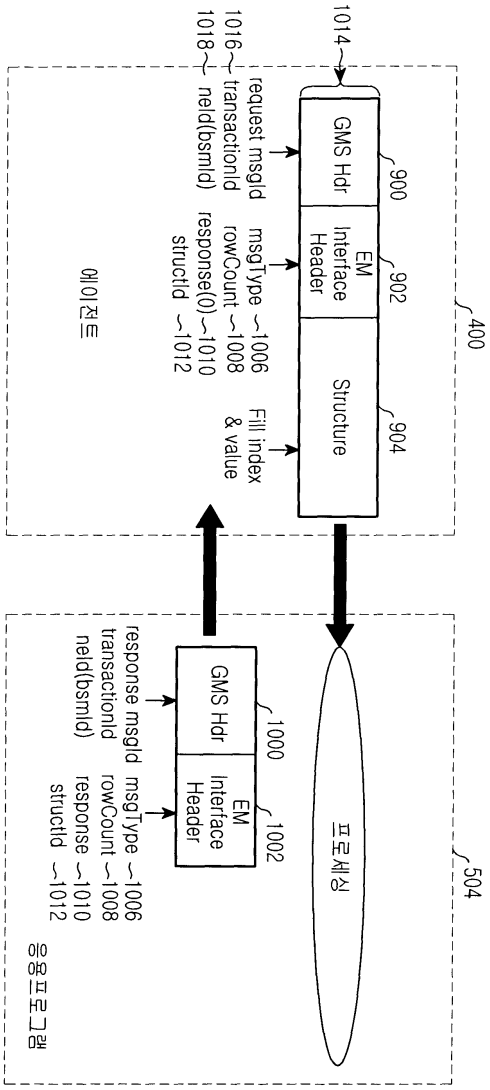
도면10


1000
1002
1004
1014
GMS Hdr
EM Hdr
Structure
.....
GMS Header
GMS Payload
1006

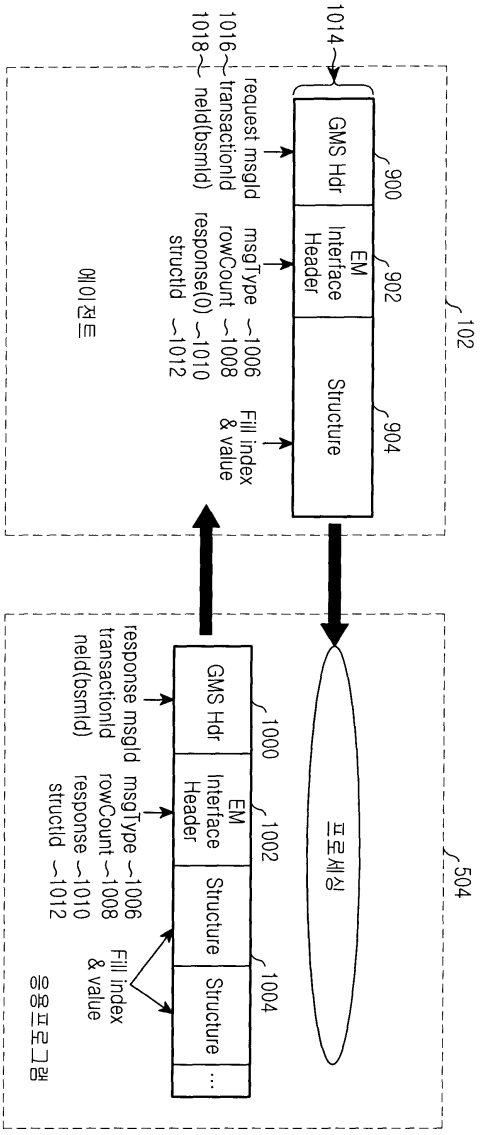
도면11


1014
1016
1018
GMS Hdr
EM Interface Header
Structure
1000
1002
1004
400
request msgId
transactionId
nelId(bsmId)
msgType
rowCount
response(0)
structId
~1006
~1008
~1010
~1012
Fill index & value
에이전트
프로세싱
504
response msgId
transactionId
nelId(bsmId)
msgType
rowCount
response
structId
~1006
~1008
~1010
~1012
Fill index & value
응용프로그램

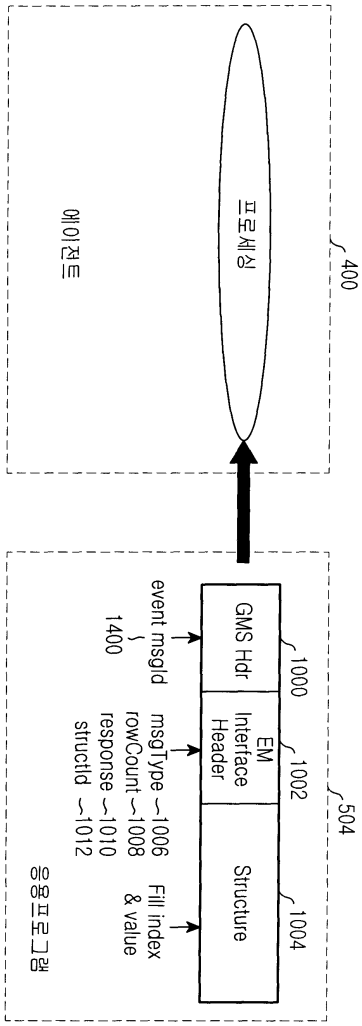
도면12


1014
1016
1018
GMS Hdr
EM Interface Header
Structure
900
902
904
400
request msgId
transactionId
nelId(bsmId)
msgType ~1006
rowCount ~1008
response(0)~1010
structId ~1012
Fill index & value
에이전트
프로세싱
504
GMS Hdr
EM Interface Header
1000
1002
response msgId
transactionId
nelId(bsmId)
msgType ~1006
rowCount ~1008
response ~1010
structId ~1012
응용프로그램

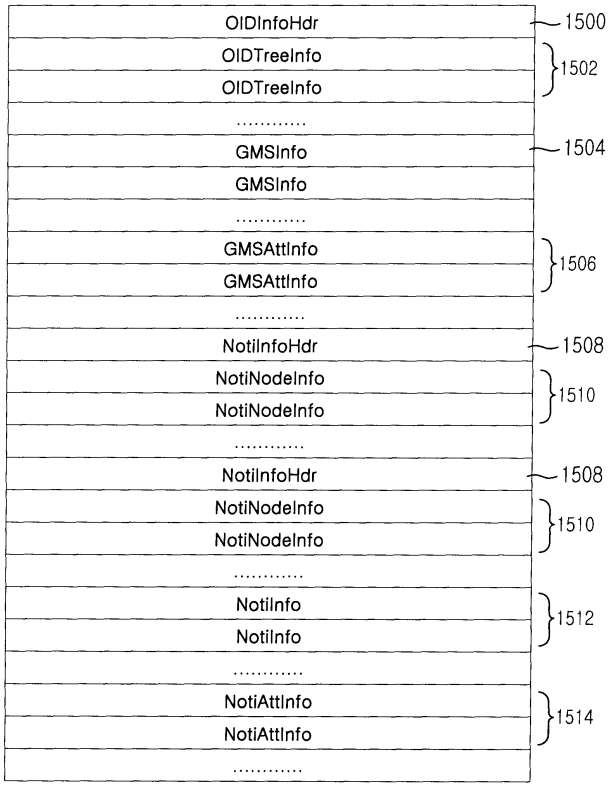
도면13


1014
1016
1018
request msgId
transactionId
nelId(bsmId)
GMS Hdr
EM Interface Header
Structure
900
902
904
102
msgType ~1006
rowCount ~1008
response(0) ~1010
structId ~1012
Fill index & value
에이전트
504
프로세싱
GMS Hdr
EM Interface Header
Structure
Structure
...
1000
1002
1004
response msgId
transactionId
nelId(bsmId)
msgType ~1006
rowCount ~1008
response ~1010
structId ~1012
Fill index & value
응용프로그램

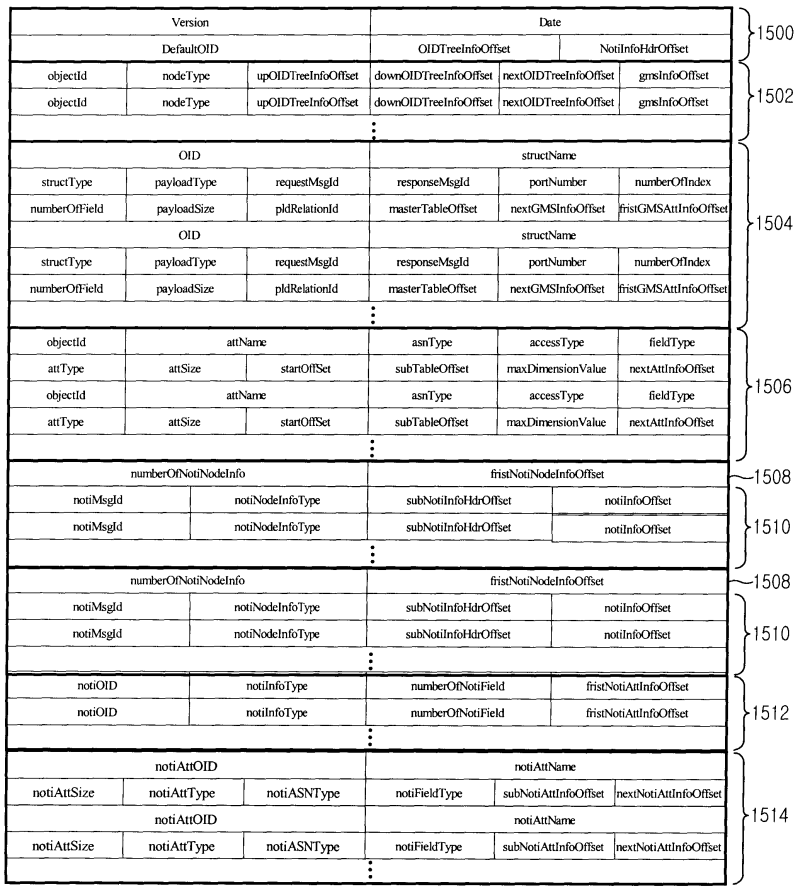
도면14


에이전트
프로세싱
400
GMS Hdr
EM Interface Header
Structure
1000
1002
1004
504
event msgId
1400
msgType ~1006
rowCount ~1008
response ~1010
structId ~1012
Fill index & value
응용프로그램

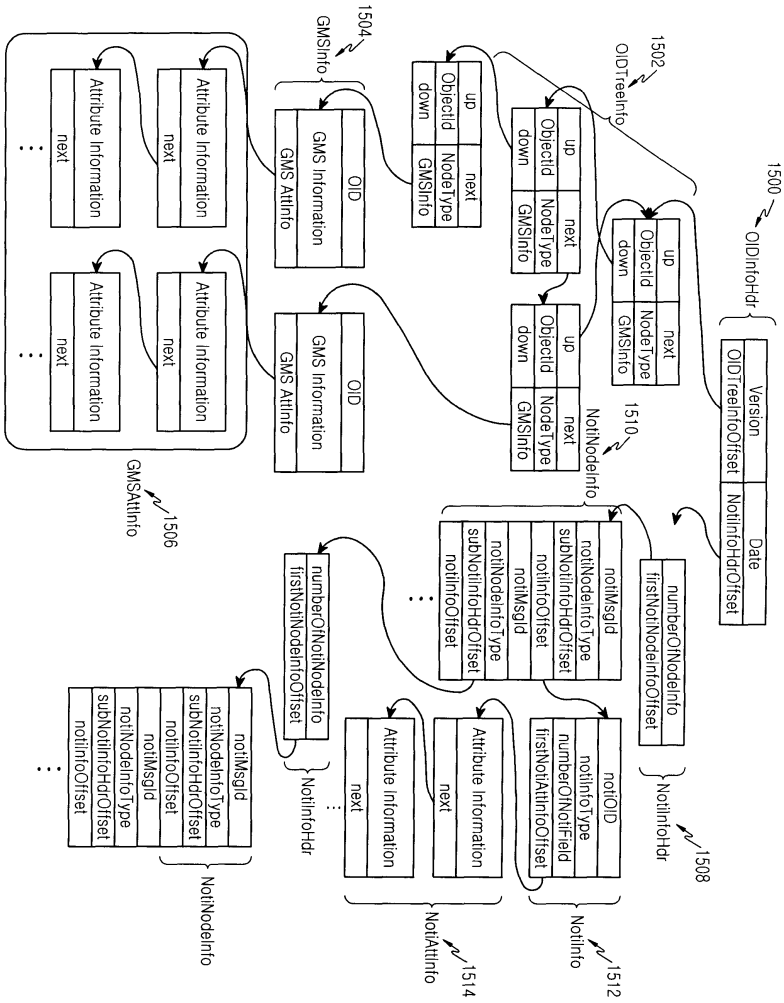
도면15a


OIDInfoHdr
1500
OIDTreeInfo
OIDTreeInfo
1502
GMSInfo
1504
GMSInfo
GMSAttInfo
GMSAttInfo
1506
NotiInfoHdr
1508
NotiNodeInfo
NotiNodeInfo
1510
NotiInfoHdr
1508
NotiNodeInfo
NotiNodeInfo
1510
NotiInfo
NotiInfo
1512
NotiAttInfo
NotiAttInfo
1514

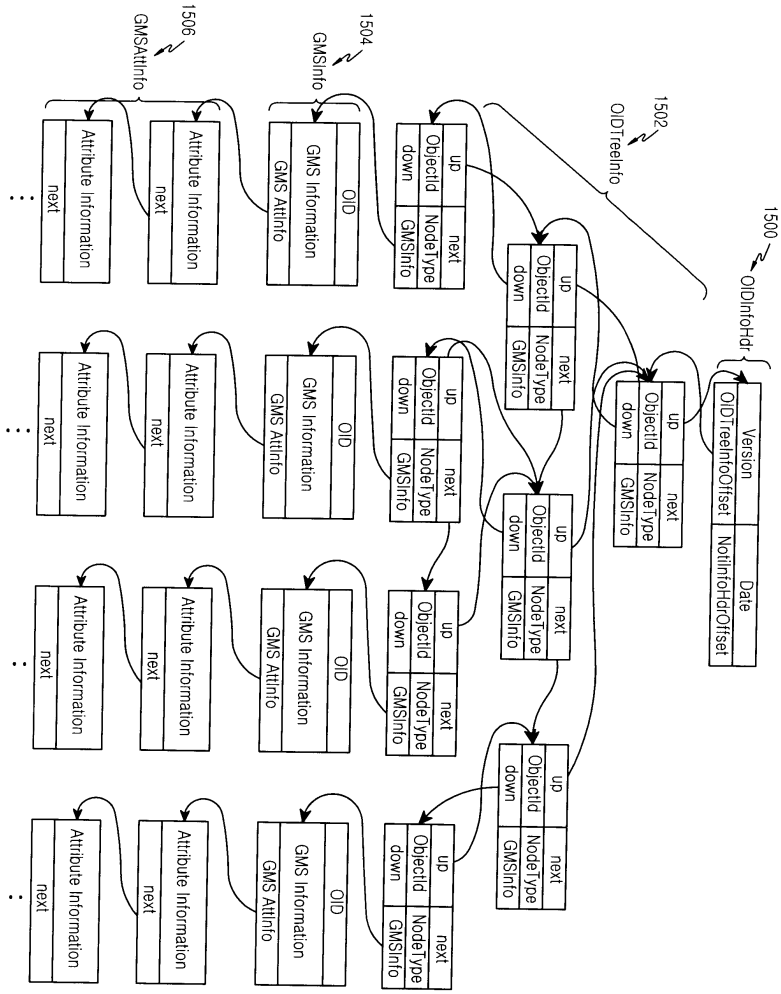
도면15b


Version
Date
1500
DefaultOID
OIDTreeInfoOffset
NotiInfoHdrOffset
objectId
nodeType
upOIDTreeInfoOffset
downOIDTreeInfoOffset
nextOIDTreeInfoOffset
gmsInfoOffset
1502
OID
structName
structType
payloadType
requestMsgId
responseMsgId
portNumber
numberOfIndex
numberOfField
payloadSize
pIdRelationId
masterTableOffset
nextGMSInfoOffset
fristGMSAttInfoOffset
1504
attName
asnType
accessType
fieldType
attType
attSize
startOffSet
subTableOffset
maxDimensionValue
nextAttInfoOffset
1506
numberOfNotiNodeInfo
fristNotiNodeInfoOffset
1508
notiMsgId
notiNodeInfoType
subNotiInfoHdrOffset
notiInfoOffset
1510
notiOID
notiInfoType
numberOfNotiField
fristNotiAttInfoOffset
1512
notiAttOID
notiAttName
notiAttSize
notiAttType
notiASNType
notiFieldType
subNotiAttInfoOffset
nextNotiAttInfoOffset
1514

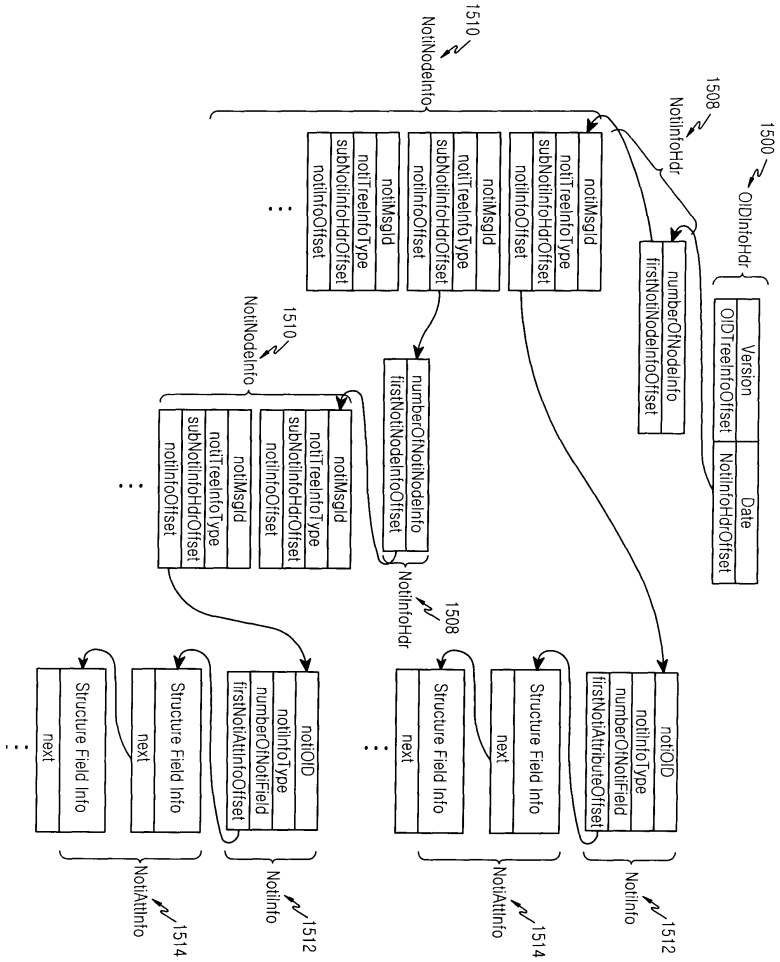
도면16


1500
OIDInfoHdr
Version
OIDTreeInfoOffset
NotiInfoHdrOffset
Date
1502
OIDTreeInfo
ObjectId
NodeType
GMSInfo
up
down
next
1504
GMSInfo
GMS Information
GMS AttInfo
OID
Attribute Information
1506
GMSAttInfo
1508
NotiInfoHdr
numberOfNodeInfo
firstNotNodeInfoOffset
1510
NotiNodeInfo
notiMsgId
notiNodeInfoType
subNotiInfoHdrOffset
notiInfoOffset
subNotiInfoHdrOffset
notiInfoOffset
numberOfNotiNodeInfo
firstNotiNodeInfoOffset
1512
Noti Info
notiOID
notiInfoType
numberOfNotiField
firstNotiAttInfoOffset
1514
NotiAttInfo

도면17


1500
OIDInfoHdr
Version
OIDTreeInfoOffset
NotiInfoHdrOffset
Date
1502
OIDTreeInfo
ObjectId
up
down
NodeType
next
GMSInfo
1504
GMSInfo
OID
GMS Information
GMS AttInfo
1506
GMSAttInfo
Attribute Information
next

도면18


1500
OIDInfoHdr
Version
Date
OIDTreeInfoOffset
NotiInfoHdrOffset
1508
NotiInfoHdr
numberOfNodeInfo
firstNotiNodeInfoOffset
1510
NotiNodeInfo
notiMsgId
notiTreeInfoType
subNotiInfoHdrOffset
notiInfoOffset
numberOfNotiNodeInfo
firstNotiNodeInfoOffset
notiOID
notiInfoType
numberOfNotiField
firstNotiAttributeOffset
Structure Field Info
next
1512
NotiInfo
1514
NotiAttInfo