(기술보고서)

# 안정적인 시스템 부트를 위한 펌웨어 분석

**KISTI 국가슈퍼컴퓨팅본부 슈퍼컴퓨터개발센터**

**2018. 4. 2.**

한국과학기술정보연구원

# 저자소개


## 정현미 (Hyun Mi Jung)
Korea Institute of Science and Technology Information
Center for Development of Supercomputing System
Senior Researcher
Email : hmjung@kisti.re.kr

## 구경모 (Kyungmo Koo)
Korea Institute of Science and Technology Information
Supercomputing Center
Researcher
Email : kookm@kisti.re.kr

## 김상완 (Sangwan Kim)
Korea Institute of Science and Technology Information
Supercomputing Center
Senior Researcher
Email : sangwan@kisti.re.kr

## 차광호 (Kwangho Cha)
Korea Institute of Science and Technology Information
Center for Development of Supercomputing System
Senior Researcher
Email : khocha@kisti.re.kr

# 목 차

# 제 1 장 연구 배경 및 범위

## 1. 연구 배경

■ 안정적인 시스템 부트(Boot)를 위해서는 UEFI (Unified Extensible Firmware Interface)에 대한

- UEFI 아키텍쳐 표준 규격 조사, 단계별 (SEC - PEI – DXE - BDS - TSL - RT - AL)기능 분석 및 UEFI 펌웨어 개발 환경 구축이 필요함
- 안정적인 시스템 구동 및 POST (Power On Self Test)를 위한 UEFI 단계별 펌웨어 요구사항 도출이 필요함

■ 하드웨어 제어 및 상태 모니터링을 위한 BMC(Baseboard Management Controller) 펌웨어 요구사항 도출 및 분석을 위하여

- 최신 BMC 칩 규격 조사, PCH (Platform Controller Hub)와의 연동 인터페이스 분석 및 BMC 펌웨어 개발 환경 구축
- 원격 하드웨어 제어, 전력/온도/습도/팬속도 등 센서 데이터 모니터링 등을 위한 소프트웨어 계층별 BMC 펌웨어 요구사항 도출 및 분석
- 하드웨어 제어 및 상태 모니터링 지원을 위한 BMC 펌웨어 소프트웨어 계층별 구조 및 IPMI (Intelligent Platform Management Interface) 스펙 분석 등 이 필요함

## 2. 연구 범위

본 기술 조사·분석 문서는

■ 안정적인 시스템 부트를 위한 UEFI 펌웨어 요구사항 도출 및 기능 분석을 위하여

- UEFI 특징 분석

- UEFI 부팅 단계 분석

■ 하드웨어 제어 및 상태 모니터링을 위한 BMC 펌웨어 요구사항 도출 및 분석을 위하여

- BMC 구성요소 및 필수요구사항 분석
  - BMC 하드웨어 구성 요소 분석
  - BMC 구동 안정성을 위한 펌웨어 구성 요소
  - 모듈 패키지 기반 펌웨어
  - 펌웨어 운영체제 확장성 및 개선 용이성
  - 하드웨어 제어 및 상태 모니터링 시스템 지원 여부

- 안정적인 BMC 구현을 위한 IPMI 스펙 분석
  - BMC 펌웨어 요구사항을 도출하기 위하여 IPMI 표준 스펙
  - IPMI와 BMC의 Disadvantage와 그에 대한 해결방안
  - BMC 기능별 IPMI 2.0 스펙분석의 예를 포함하고 있음

# 제 2 장 UEFI 펌웨어 요구사항 도출 및 기능 분석

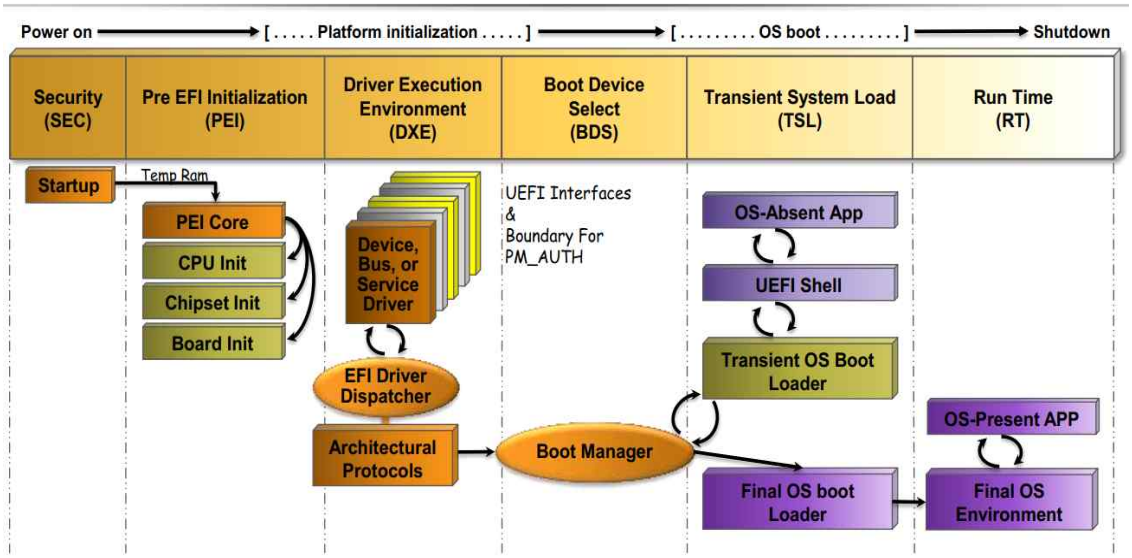
## 1. UEFI 특징 분석

- ■ 하드웨어 시스템 진보에 부응하여 다수의 커뮤니티에 의해 지속적인 업데이트가 가능한 펌웨어가 요구됨
- ■ 기존 BIOS는 16비트 기반이며 1MB의 실행 메모리 제한이 있어 64비트 시스템에 적용하기에 한계가 있음
- ■ 하드웨어 독립적이며 대형 시스템에 적용 가능한 부팅 시스템이 요구됨, 하드웨어 추상화를 통한 하드웨어 Independency
- ■ 개발을 위한 소스 코드의 가독성, 개발 환경 구축의 용이성이 요구됨
- ■ OS 런타임 중에도 접근 가능하도록 서비스를 제공해야 함
- ■ 운영 체제 비정상 종료 이후 재시작 시 충돌 메시지를 NVRAM에 보관하는 기능 등 플랫폼 펌웨어와 운영 체제 사이에 공유되는 비휘발성 데이터의 저장 기능이 요구됨
- ■ 2007년 UEFI 2.1 버전 이후로 2017년 2.7 버전이 릴리즈되어 지속적인 업데이트
- ■ 다양한 그룹으로 나누어져서 버전 관리 및 활동이 진행중, (ASWG, ICWG, PSST, PDST, ABST, USST, UVST 등 다양한 그룹에서 표준화 진행)
- ■ Legacy BIOS는 벤더 의존적인 인터페이스를 이용하였으나, UEFI는 표준 SPEC으로 다양한 시스템에 범용적으로 적용 가능
- ■ 기존 BIOS는 주소 공간이 1MB, Option ROM 공간은 128KB로 제한되는 반면 UEFI 표준 펌웨어는 메모리 제한이 없음
- ■ 운영체제와 펌웨어 간의 인터페이스를 제공하며, 아키텍쳐 독립적이며 모듈화 방식을 도입하여 확장성이 용이, UEFI를 지원하지 않는 OS를 위하여 CSM(Compatibility Support Module) 요소가 존재함
- ■ GPT 파티션 지원 : 레거시 바이오스는 MBR(Master Boot Record) 파티션 방식을 사용하여 32비트 기반으로, 최대 4개의

primary 파티션을 지원하며, 최대 파티션 크기는 2TB로 제한되나 UEFI에서는 GPT(GUID Partition Table) 파티션을 지원하며, 64비트 기반으로 최대 128개의 primary 파티션을 지원함. 최대 파티션의 크기는 9ZB.

■ EDK core : UEFI Spec에 따라 작성된 UEFI 기반 펌웨어 메인 오픈 소스 코드, BIOS 펌웨어 벤더에서는 EDK core를 수정하거나 확장하여 펌웨어를 제작

## 2. UEFI 부팅 단계 분석

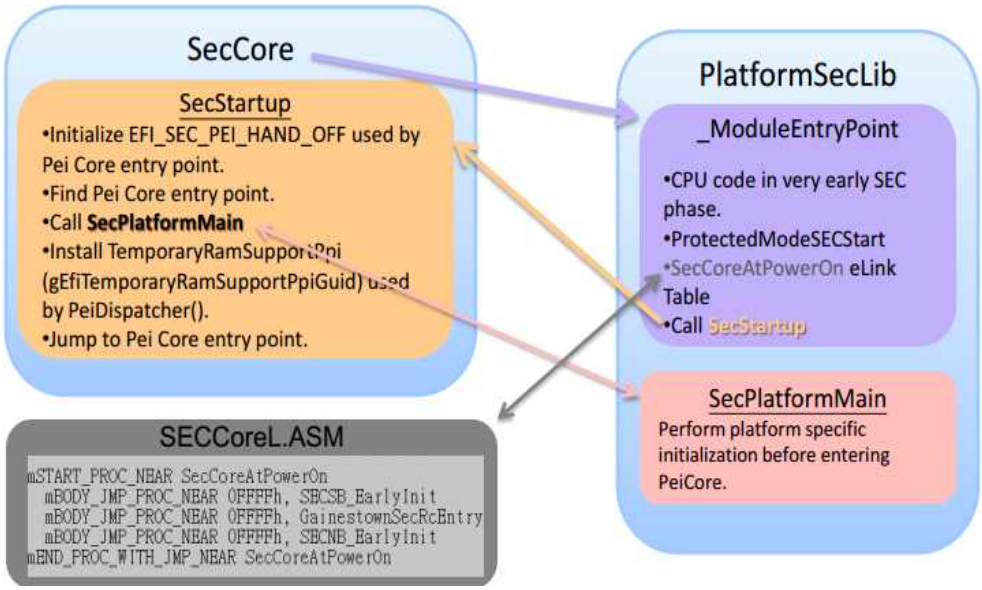



[그림 1] UEFI 부팅 단계 흐름도

■ SEC(Security Phase) 단계

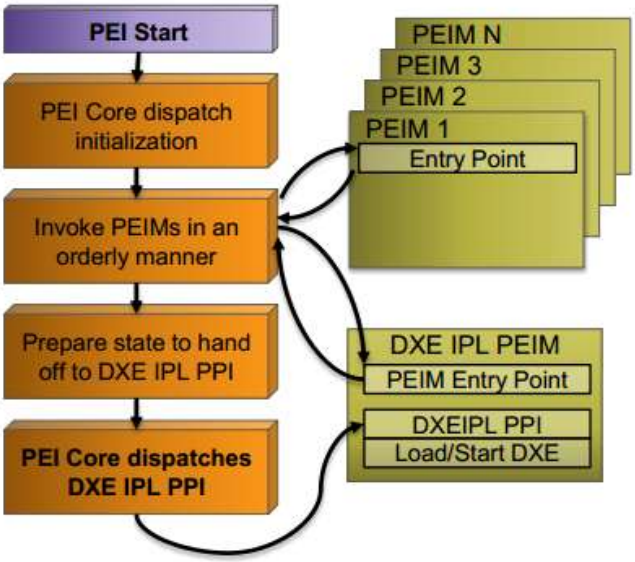
- Platform Initialization (PI) 아키텍처의 첫 번째 단계
- 모든 플랫폼 재시작 이벤트를 처리
- 임시 메모리 영역 생성, CPU cache로 부터 임시 메모리를 초기화, 캐시를 RAM처럼 사용하는 cache-as-ram(CAR)방식
- 시스템 상에서 root of trust로써 동작 : 시스템 제어 코드 초기화
- 마이크로코드를 패치 함

- CPU는 16비트 모드에서 32비트 보호 모드로 전환
- SEC 사우스 브릿지 초기화 : 시스템 인터럽트 중지, PM 및 GPIO 베이스 어드레스 설정, RCBA 활성화, 80 Port 및 기타 LPC 입출력 디코딩
- SEC 노스 브릿지 초기화 : PCI Express 및 MCHBAR (Host Memory Mapped Register Range Base Register) 베이스 어드레스 설정
- 시스템 정보와 SEC Platform 인터페이스들을 PEI 단계로 넘겨준다. (Boot Firmware Volume(BFV)의 크기와 위치, 임시 RAM의 크기와 위치, stack의 크기와 위치)




[그림 2] SEC 접속

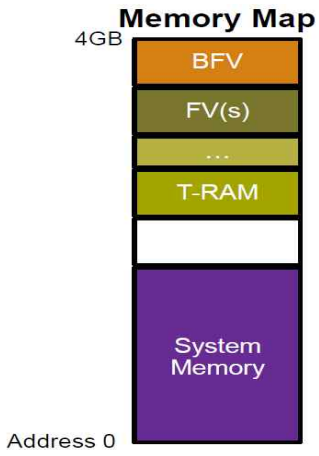
■ PEI (Pre EFI Initialization) 단계
- PI 아키텍처의 두 번째 단계




[그림 3] PEI 개요

- 메모리 초기화 : 메인 메모리를 초기화하기 위한 모듈을 실행, PEI 단계에서 초기화 된 메모리는 재설정 불가
- 부트 모드(boot mode)를 결정 (resume, recovery, normal or fresh boot인지, sleep mode(S3) boot인지를 결정)
- 프로세서와 칩셋 인터페이스를 초기화
- 추가적인 펌웨어 볼륨을 검색
- 디버깅을 위한 인프라를 준비 (SEC 단계에서는 하드웨어적인 디버깅 방식이 사용됨)
- PEI Phase에서만 인식 가능한 플랫폼 특정 자원들을 설정
- PEI 구성 요소 : PEI Core 및 PEI Modules (PEIMs), PEIM-to-PEIM Interface(PPIs), PEI는 Reset -> Start-up -> Support의 단계로 구성
- PEIM : PEIM은 펌웨어의 모듈화 된 chunk로써 PEI 코어에 의해 실행되며 칩셋과 플랫폼을 지원
- PEI Core : PEIMs을 제어하고 PEI 핵심 서비스를 제공하기 위한 PEI의 주 executable binary
- SEC 단계에서 설정된 임시 메모리(CPU Cache)가 재위치 되면 PEI단계는 ROM에서 실행 됨, ROM의 크기 제한으로 인해 집약적인 코드로써 실행

- PEI memory – PEI 메모리 초기화 는 Pre-Permanent Memory, Initial Memory, Memory Map, Post-Permanent Memory로 구성됨
- Memory Map : Memory Map은 BFV, FV(s), T-RAM, System Memory로 구성됨. PEI 코드가 저장되어 있는 장소는 Boot Firmware Volume(BFV)이며 PEI는 시스템 메모리가 초기화되기 이전에 temporary memory인 (T-RAM)을 사용

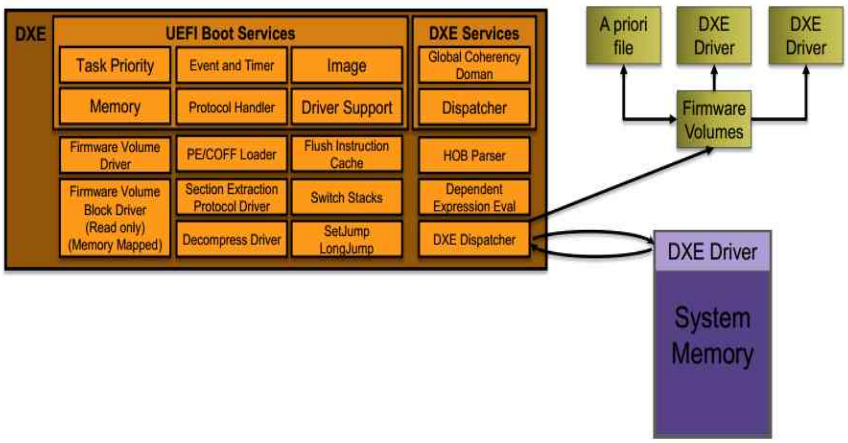



[그림 4] 메모리 맵 구성

- Pre-Permanent Memory : 메모리를 초기화시키고 테스트, FV 위치와 PEI 서비스 테이블의 위치를 파악, 32bit protected mode로 동작하며 프로세서 캐쉬를 램처럼 사용 (실제 시스템 메모리를 사용하지 않음)
- Post-Permanent Memory : HOBs를 설정하여 PEI에서 DXE 단계로 전환시키며 시스템 메모리를 이용 가능
- HOB(Handoff Blocks) : platform configration 정보를 DXE단계로 전달하기 위해 사용
- DXE IPL(DXE Initial Program Load) PEI to DXE Handoff : PEI로부터 DXE 단계로 전환되는 과정을 수행, PEI Core는 DXE IPL PEIM을 이용하여 DXE Foundation을 설정, DEX 단계로 전환이 완료되면 PEI services와 PPIs는 더 이상 사용 불가능, DXE Core를 메모리로 적재시키며 HOB 리스트로부터 FVs를 검색

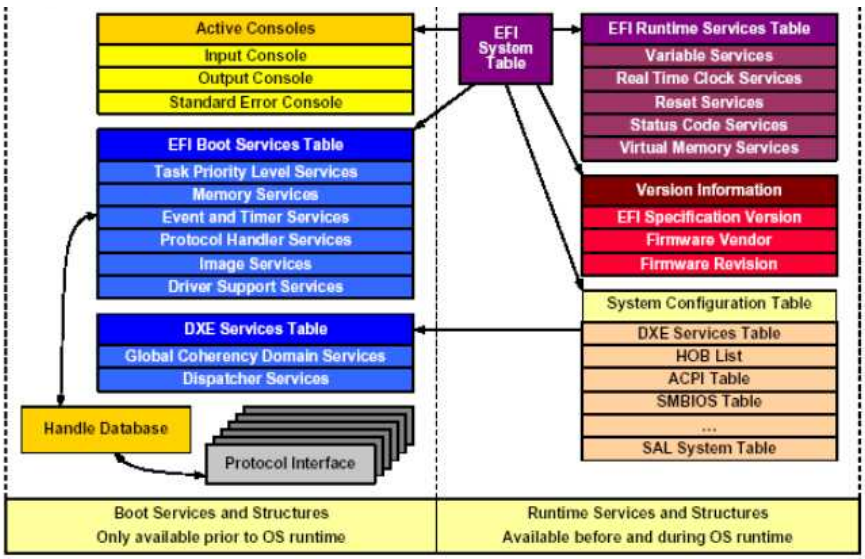
■ DXE (Driver Execution Environment) 단계

- 드라이버를 이용하여 플래폼 초기화 작업을 완료함
- 하드웨어 의존적인 Architectural Protocols을 수행함
- 드라이버를 이용하여 플래폼 초기화 작업을 완료함
- Boot POST 와 런타임 드라이버를 포함함



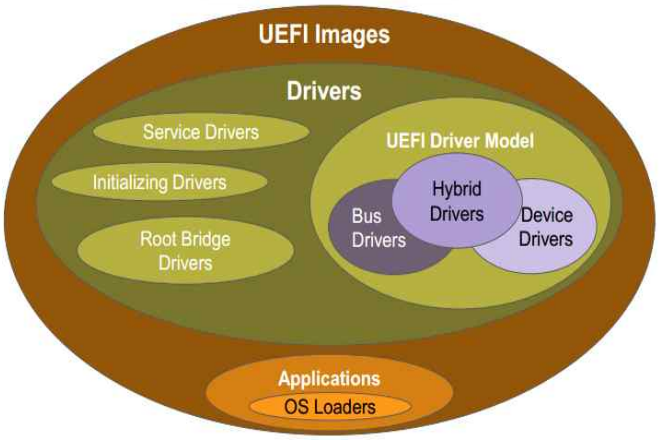

[그림 5] DXE 단계 흐름

- DXE Core : DXE 단계의 주 executable binary, DXE Dispatcher를 이용하여 DXE 드라이버를 관리하고, 부트 서비스, 런타임 서비스 등을 생성 및 관리함




[그림 6] DXE 서비스 - UEFI 시스템 테이블

- DEX Driver : 코어에 의해 로드되며 플랫폼 전반 초기화 작업, 프로토콜 및 기타 서비스 생성 작업을 담당함




[그림 7] DXE 드라이버 개요

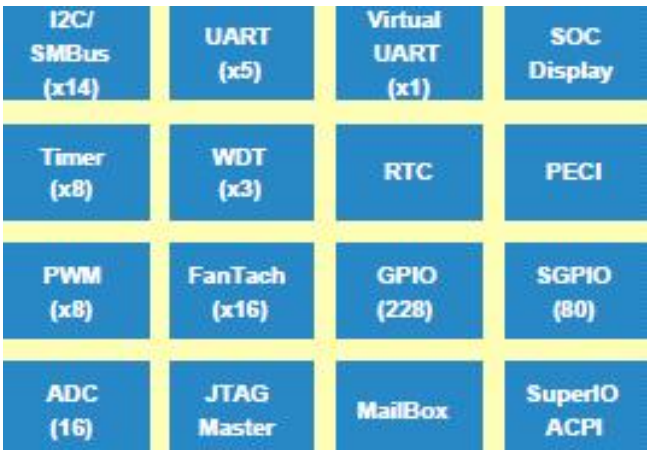
- DXE Architectural Protocols : DXE 단계를 하드웨어로부터 추상화 시키는 역할을 담당, CPU, chipset, board 사양을 Encapsulation
- DXE Dispatcher : 드라이버의 검색과 실행을 담당하는 DXE 코어의 한 부분
- EFI System Table : EFI system table은 모든 EFI service table, configuration table, handle data base, console device를 참조할 수 있는 포인터들을 포함
- DXE Core Initialization : EFI Boot Services Table, DXE Services Table, Memory-Only TPL Services, Memory Services 를 초기화, EFI System Table 및 EFI Runtime Service Table을 할당

■ BDS (Boot Device Select) 단계
- 콘솔 디바이스를 초기화
- 디바이스 드라이버를 로드
- UEFI 어플리케이션 지원
- CSM (compatible support module)을 통하여 기존 레거시 인터페이스를 지원
- 부트 디바이스를 검색함

- OS를 로드함
- 제어권을 OS에 넘김

■ TSL (Transient System Load) 단계
- EFI 어플리케이션으로서 OS 로더가 수행하는 부팅 프로세스의 첫 번째 단계
- OS 부트로더로 넘어가기 위한 임시적인 단계
- 부팅이 실패할 경우 BDS 단계로 다시 복귀함
- 부트 서비스를 종료함

■ RT (Runtime) 단계
- 런타임 서비스
- System Management Interrupts (SMI) 처리




[그림 8] BMC 칩 기본 기능

# 제 3 장 BMC 구성요소 및 필수요구사항 분석

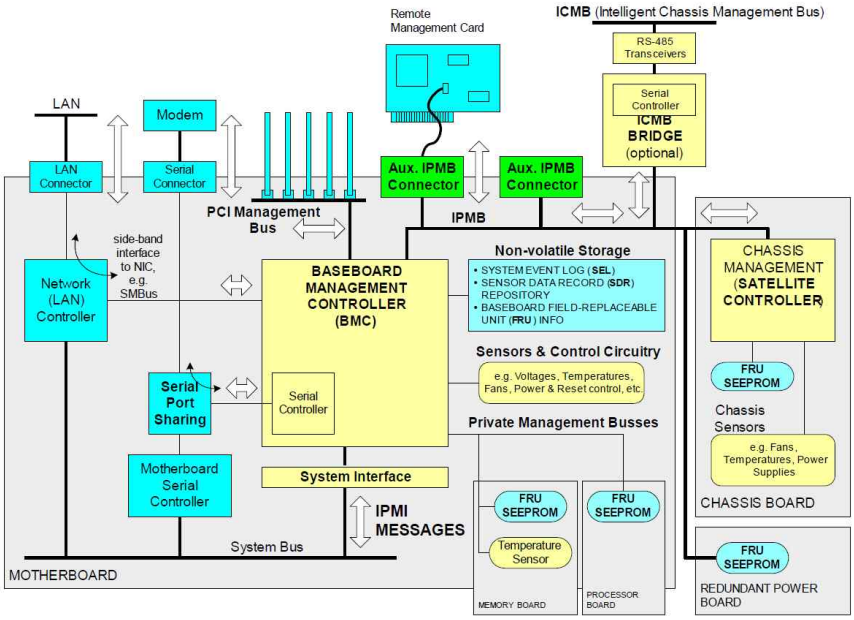
## 1. BMC 개요

■ BMC 기본 역할 :

- IPMI2.0 기준에 따라 센서 모니터링 기능 제공, 시스템 이벤트 로그 기능 제공, 센서 데이터 저장 기능, FRU 기능, IPMI 및 ICMB 기능, SoL 기능, 센서 및 전원 제어 기능
- 센서 모니터링 기능 제공 : 온도, 전원 전압, 팬 속도와 같은 센서가 장착되어 BMC와 연결된 경우 기능 제공
- 시스템 이벤트 로그(System Event Log: SEL) : BMC 센서들의 임계값 설정에 위반되는 경우나 시스템 전원 on/off 요청 등의 이벤트 메시지 내용과 장치 이름 로그 시간 등을 NVRAM에 저장
- 센서 데이터 저장(Sensor Data Repository) 기능 : 시스템에 장착되어 있는 모든 장치들에 관한 데이터 정보를 NVRAM에 저장 (센서 주소, 이름, 형태, 단위, 임계치, 개체 등)
- FRU (Field Repaceable Unit) 기능 : 보통 외부로 연결된 NVRAM 형태, BMC와 인터페이스 하며, 제품 시리얼 번호 및 모델명 등이 포함됨
- IPMB(Intelligent Platform Management Bus) : IPBM는 샤시 내에 이벤트 전달, 모니터링, 제어 및 관리의 확장을 위한 표준화된 버스 및 프로토콜을 제공, 기본적으로 I2C가 제공됨
- ICMB(Intelligent Chasis Magagement Bus) : ICMB는 여러 대 호스트 연결 및 병렬 샤시 연결로 확장할 경우 사용되며 64개의 샤시 까지 확장 가능
- SoL (Serial Over Lan) 기능 : 네트워크를 통한 시리얼 포트의 리다이렉션 기능 제공, OS 부팅 이전 화면(Pre OS Boot Screen) 모니터링 기능 제공 및 BIOS Setup 기능 수행
- 센서 및 전원 제어 기능 : 각 센서 임계치 설정 및 전원 제어 기능

## 2. BMC 하드웨어 컴포넌트

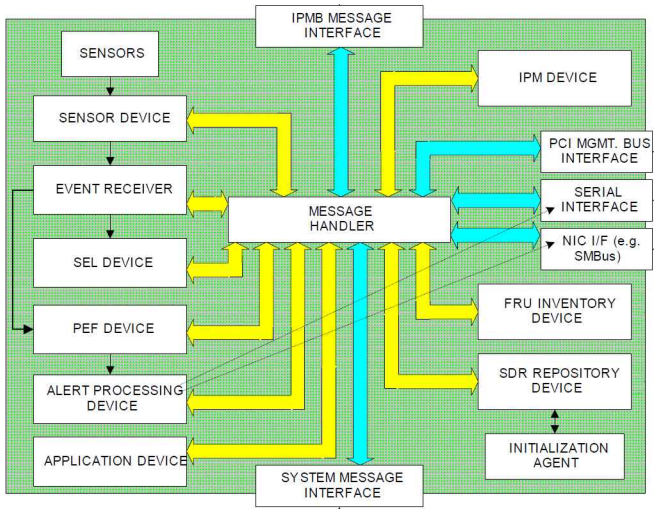
■ BMC, Event Generator, SDR Repository, ICMB, IPMB, LAN Controller, Serial Controller, Sensor & Control circuitry, Non-volatile Storage Private management Busses 등으로 구성됨




[그림 9] IPMI BMC 하드웨어 및 인터페이스 구성 요소

## 3. BMC 매니지먼트 인터페이스

■ BMC는 PCI MGMT.BUS INTERFACE, SERIAL INTERFACE, NIC INTERFACE, IPMB MESSAGE INTERFACE, SYSTEM MESSAGE INTERFACE, SMBus SYSTEM INTERFACE, SENSOR REPOSITORY INTERFACE 등 다양한 시스템 인터페이스를 제공하며 시스템 제어를 위해 각 인터페이스 간 통신 수단이 요구됨




[그림 10] 메시지 핸들러 중심의 BMC 인터페이스

## 4. BMC 구동 안정성

■ BMC 및 하드웨어 장치들을 안정적으로 구동시키기 위한 운영체제 및 드라이버가 요구됨. 운영체제를 시동하기에 앞서 하드웨어를 점검 및 초기화시켜주어 커널을 안정적으로 비휘발성 저장소로부터 메모리에 적재 가능하도록 해주는 펌웨어 요소가 요구됨

## 5. 모듈 패키지 기반 펌웨어

■ 모듈 패키지 적재 또는 제거를 통해 다양한 시스템에 유동적으로 적용 가능한 모듈 기반 펌웨어, BMC에 따라 펌웨어 소형화 및 최적화가 필요하기 때문에 펌웨어 기능의 모듈화가 요구됨

## 6. BMC 펌웨어 확장성 및 개선 용이성

■ 펌웨어 적재 시 커널 레벨에서 발생 가능한 버그 또는 호환성 문제 등 운영체제의 기능 개선이 요구되는 상황에 대비하여, 오픈소스 커뮤니티 등 다수의 개발 그룹에 의해 지속적으로 업데이트 가능한 운영체제가 요구됨

## 7. 하드웨어 제어 및 상태 모니터링 시스템 지원

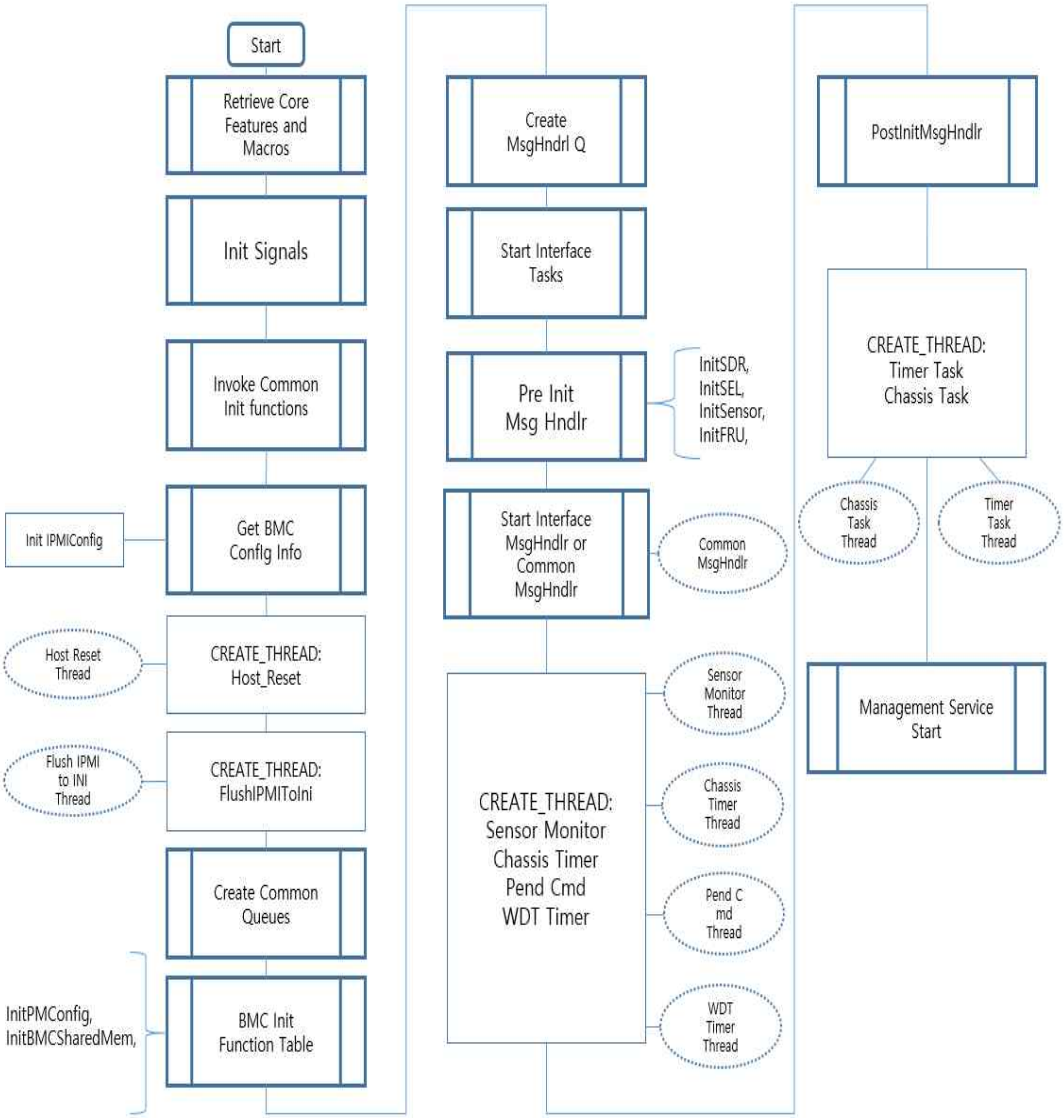
- IPMI driver 및 ipmitool 등 하드웨어 모니터링 소프트웨어 구동에 대한 지원 및 호환성이 요구됨

## 8. BMC 펌웨어 컴포넌트 요구사항

- BMC 및 디바이스 장치에 대한 구동 드라이버, BMC 매니지먼트 인터페이스들에 대한 제어 소프트웨어, 다양한 BMC 매니지먼트 인터페이스 및 디바이스 장치에 대한 Message Handler, BMC 구동을 위한 운영체제 및 부트로더, 시스템 제어 및 모니터링을 위한 유저 애플리케이션 등이 요구됨

<표 1> BMC 펌웨어 요구사항

| 구성 요소 | 요구 사항 및 역할 |
|---|---|
| Bootloader (u-boot) | BMC 구동 운영체제를 메모리에 적재시켜 시동시키기 위한 구성 요소 |
| Kernel (uImage) | BMC 구동을 위한 운영체제, BMC에 따라 Embedded Linux와 같은 오픈 소스 운영체제로 구동 가능 |
| File System packages | 장치 드라이버, 애플리케이션 등 커널에서 실행 될 소프트웨어 요소들을 위한 파일 시스템 관련 패키지 |
| Device Drivers | ADC, BT, GPIO, I2C, PECI 등 BMC 디바이스 장치들을 제어/구동하기 위한 요소 |
| Device libraries | 장치 드라이버 및 애플리케이션을 적절하게 실행시키기 위한 함수와 변수들이 저장되어 있는 소프트웨어 컴포넌트 |
| Device Apps | BMC 디바이스 장치들을 사용자 수준에서 제어하고 모니터링하기 위한 구성 요소 |
| IPMI driver | IPMI 구동에 요구되는 소프트웨어 컴포넌트, IPMI 인터페이스 및 IPMI Message Handler 제어 역할 등을 수행 |
| IPMI Main Process | IPMI Message Handler를 통한 BMC 간 통신 및 유저 인터페이스로의 시스템 정보 제공 등 시스템 모니터링 및 제어 전반에 걸친 역할 수행 |
| IPMI tools (IPMI application) | IPMI commands를 통해 SDR 및 LAN 정보 등의 시스템 정보를 사용자 수준에서 모니터링/제어 가능하게 해주는 유저 레이어 애플리케이션 |




[그림 11] 시스템 관리를 위한 BMC 펌웨어 메인 서비스 구동 흐름도

# 제 4 장 안정적인 BMC 구현을 위한 IPMI 스펙 분석

## 1. IPMI 기본 개념

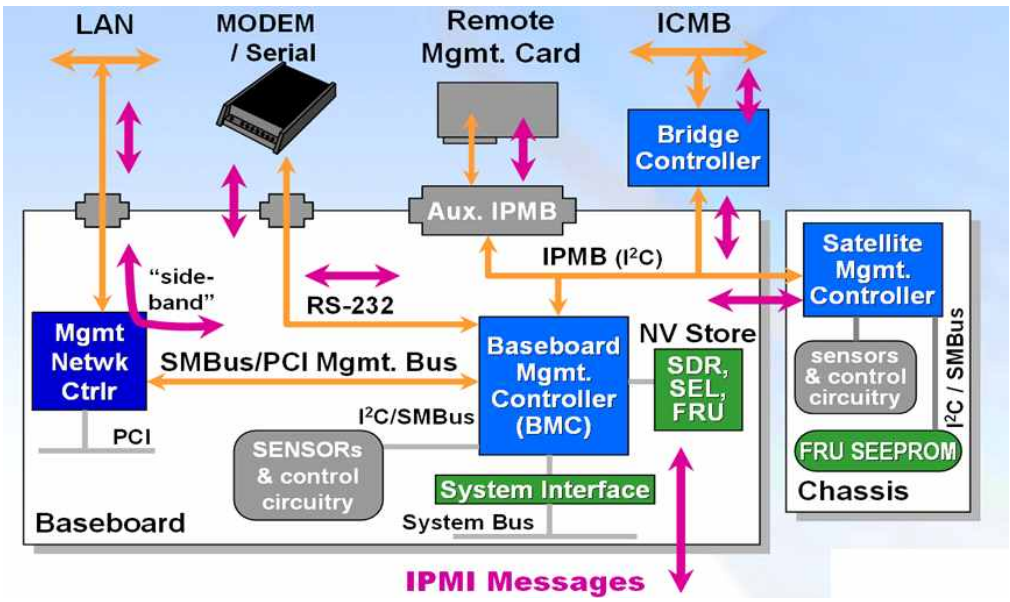
### 1) IPMI 개요

- IPMI는 지능형 플랫폼 관리 인터페이스로 소위 “플랫폼 관리” 서비스에서 표준화 된 인터페이스를 제공하는 일련의 사양
- IPMI 표준은 서버 하드웨어의 관리에 집중된 구조로서 1998년 IntelTM 과 HP 중심으로 70 여개의 회사가 모여서 만든 IPMI 포럼에 의해 IPMI1.0 표준 규격이 제정

### 2) IPMI의 기본 주요 기능

- 모니터링 (시스템정보: 팬, 온도등 감독)
- 복구 제어 (서버 복구 / 다시 시작)
- 비정상 이벤트 로깅 (하드웨어에 대한 프로토콜 "out-of-range"상태)
- 인벤토리 (하드웨어 인벤토리 목록)

### 3) IPMI 구성 요소




[그림 12] IPMI 구성요소

■ 베이스 보드 관리 컨트롤러 (BMC: Baseboard Management Controller)

- 마이크로 컨트롤러 BMC는 IPMI 아키텍처의 핵심:
  - 시스템 관리 소프트웨어와 사용 중인 하드웨어 사이의 인페이스 (BMC가 IPMB 및 ICMB를 사용하여 연결됨)
  - 독립적으로 모니터링
  - 독립적으로 이벤트 로깅
  - 복구 제어

■ 지능형 플랫폼 관리 버스 (IPMB: Intelligent Platform Management Bus)

- IPMI는 IPMB 표준을 적용하여 추가 관리 컨트롤러 (MC)를 통해 BMC를 확장 할 수 있도록 함
- IPMB는 I²C 기반의 직렬 버스로서 하나의 섀시 내부의 다양한 보드와의 연결을 가능하게 한다. IPMB는 관리 컨트롤러 (MC) 간의 통신에 사용됨

■ 지능형 섀시 관리 버스 (ICMB: Intelligent Chassis Management Bus)

- ICMB는 chasses 간의 통신 및 제어를 위한 표준화 된 인터페이스를 제공함

4) IPMI Memory Areas

IPMI는 SEL (System Event Log), SDR (Sensor Data Record) 저장소 및 FRU (Field Replaceable Units)에 정보를 저장함

■ 시스템 이벤트 로그 (SEL: System Event Log)

- BMC에는 중앙의 비휘발성 시스템 이벤트 로그 (SEL)가 있다. 이 SEL은 BMC에 의해 관리되기 때문에 예를 들어 IPMI LAN 액세스 등을 통해 서버의 CPU 오류가 발생한 후에도 액세스 가능

- IPMI 명령을 사용하여 SEL을 읽고 삭제가능
- SEL의 메모리는 제한적이므로 정기적으로 확인하고 삭제해야 추가 이벤트를 저장 가능

■ 센서 데이터 레코드 (SDR: Sensor Data Record) 저장소
- 센서 데이터 레코드는 센서의 유형 및 개수에 대한 정보가 들어있는 레코드
- 따라서 센서 데이터 레코드는 특정 센서를 정보를 저장하다. 센서 데이터 레코드는 BMC에 의해 관리되는 중앙의 비 휘발성 저장 영역에 저장
- 이 저장 영역을 센서 데이터 레코드 저장소 (SDR 저장소)라고 함

■ 현장 교체 가능 장치 (FRU: Field Replaceable Unit) 정보
- IPMI는 시스템의 다양한 모듈에 대한 현장 교체 가능 장치 (FRU) 정보의 저장을 지원
- FRU 데이터에는 일련 번호, 부품 번호, 모델 및 재고 번호("자산 태그"라고도 함)와 같은 정보가 들어 있음

5) 통신 인터페이스

IPMI는 다음 인터페이스를 통해 IPMI 메시징을 허용함

■ 시스템 인터페이스(system interfaces)
- IPMI는 시스템 인터페이스에서 BMC로의 로컬 액세스를 위한 몇 가지 시스템 인터페이스를 정의
- 가장 광범위한 마이크로 컨트롤러를 지원하기 위한 인터페이스로 IO 또는 mamory-mapped access를 통해 액세스 가능

- IPMI 시스템 인터페이스
- 키보드 컨트롤러 스타일 (KCS : keyboard controller style )
- 시스템 관리 인터페이스 칩 (SMIC : system management interface chip )

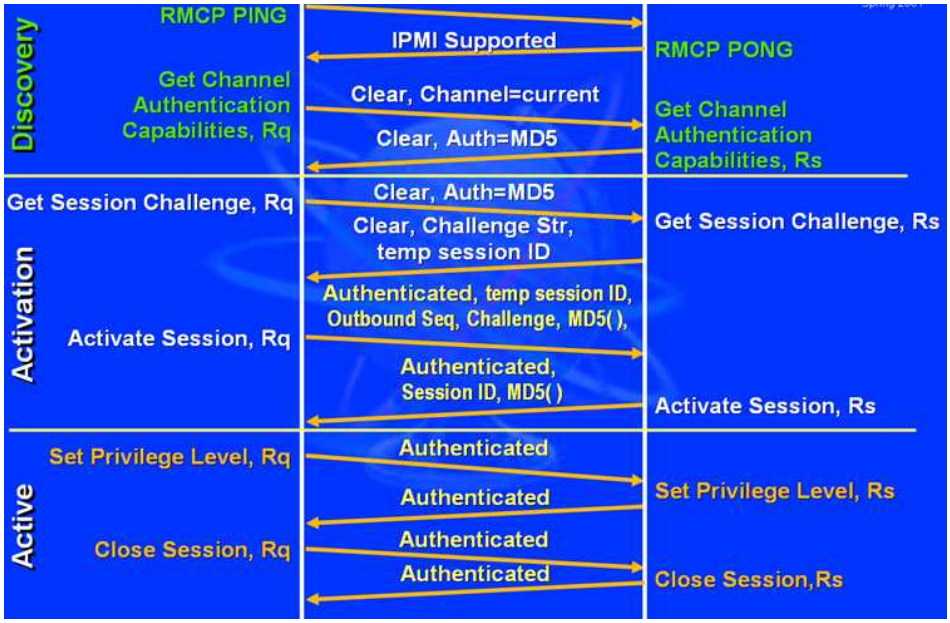
- 블록 전송 (BT : block transfer )
- SMBus 시스템 인터페이스 (SSIF : SMBus system interface )

■ 직렬 / 모뎀 인터페이스(Serial/Modem Interface)

- 직렬 또는 모뎀 인터페이스 사양은 직접 직렬 또는 외부 모뎀 연결을 통해 BMC와의 IPMI 메시지 전송 방법을 정의
- 직렬 / 모뎀 인터페이스(Serial/Modem Interface)를 위해 세 가지 연결 모드가 지원됨
  ⓐ 기본 모드
  ⓑ PPP 모드
  ⓒ 터미널 모드

■ LAN 인터페이스

- LAN 인터페이스 사양은 캡슐화 된 원격 관리 제어 프로토콜 (RMCP:Remote Management Control Protocol ) UDP datagrams ( asf-rmcp의 UDP 대상 포트 623)에서 BMC 와의 IPMI 메시지 전송 방법을 정의
- 이 기능은 "IPMI-over-LAN"이라고도 하든데 IPMI는 LAN 관련 구성 설정을 정의하며 IP 주소와 다소 유사
- RMCP(Remote Management Control Protocol)는 DMTF (Distributed Management Task Force)에서 시작되었으며, 이 패킷 형식은 IPMI 외에 DMTF Alert Standard Forum (ASF) 사양에도 사용
- 추가 패킷 형식 (RCMP +)은 IPMI 2.0에도 정의되어 있으며, RMCP +는 인증을 위한 다양한 확장 외에도 암호화 된 데이터 전송을 지원
- IPMI 메시지 외에도 RMCP + IPMI 세션을 통해 더 많은 데이터를 전송하는 기능
- 추가 유형의 데이터에 대한 한 가지 예로는 SOL (Serial-over-LAN)이 있음




[그림 13] IPMI LAN Session activation

6) 채널 모델

IPMI는 통신 인터페이스와 BMC 간의 직접 통신에 채널 모델 을 사용하며 각 채널에는 고유 한 속성과 자체 구성이 존재함

- 고유 한 채널 번호
- 통신 인터페이스 유형 (예 : LAN 인터페이스)
- 사용자 및 암호 (채널에 대해 개별적으로 작성, admin이라는 이름의 사용자에게 여러 채널에 대한 다양한 암호를 할당 가능)
- 개별적으로 지원되는 인증 모드 (예 : MD5)
- 개별적으로 구성 가능한 채널 권한 한도
- IMPI 메시징 및 경고는 각 채널에 대해 개별적으로 활성화 또는 비활성화 가능

■ 채널 번호

- 각 채널에는 개별 채널 번호가 존재함
- 기본 IPMB (채널 번호 0) 및 시스템 인터페이스 (채널 번호 0x0F 또는 15)에 대한 널 번호 만 사전 정의됨
- 나머지 채널 번호는 각 구현에 따라 상이함

| Channel Number | Type and protocol | Description |
|---|---|---|
| 0 | Primary IPMB | Channel 0 has been reserved for communication with the primary IPMB. |
| 1-11 (1-Bh) | Specific to the implementation | These channels can be used for various types of communications channels. The respectively available channels depend on the specific IPMI implementation for a specific server system. Often, Channel 1 is used as the LAN channel (some servers like the Intel SR2500 also have additional LAN channels). |
| 12-13 (Ch-Dh) | - | Reserved. |
| 14 (Eh) | Current interface | This interface is used for identifying the channel currently in use. A program can issue the IPMI *Get Channel Info* IPMI command on this channel in order to discover the channel through communication is currently being transmitted. |
| 15 (Fh) | System interface | This channel is used for the system interface. |

[그림 14] 채널 번호

■ 세션

채널은 세션 기반 또는 세션 기반, 이 세션 에서는 다음 두 가지 목적을 수행:

- 세션은 사용자 인증을 위한 프레임 워크를 제공함
- 하나의 세션에서 여러 IPMI 메시징 스트림을 단일 채널에서 처리 가능

세션 기반 채널의 예는 LAN 및 직렬 / 모뎀 채널이고 세션 없는 채널의 예는 시스템 인터페이스 및 IPMB 채널임

■ 채널 권한 수준

- 특정 최대 권한 수준에서 사용할 수 있도록 채널을 구성 가능, 다양한 권한 수준은 다음과 같음

| Privilege level | Description |
|---|---|
| Callback | This is the lowest privilege level. It permits the initiation of a callback. |
| User | Only IPMI *begin* commands will be allowed. They are primarily commands for reading and requesting about status information (sensors). Other functions (like changes to the BMC configuration, writing data to the BMC, and executing reset, power-on and power-off procedures) are not possible here. |
| Operator | All BMC commands are allowed except for those for changing the out-of-band interfaces. Deactivating channels or changing user access privileges are not possible at the operator privilege level. |
| Administrator | All BMC commands are allowed. |

[그림 15] 권한 수준

- 참고 : 채널 (예 : LAN 채널)이 관리자 권한 수준에서 변경하는 기능을 제공하는 경우, 해당 채널의 개별 사용자는 더 적은 권한 ( 사용자 권한 한도 )이 부여
- 따라서 LAN 채널 자체는 Administrator 로 Channel Privilege Limit 로 구성 될 수 있음
- 예를 들어 두 명의 사용자 ( 사용자 권한 한도 는 관리자 , 사용자 권한 한도 는 사용자 )는 그에 상응하여 설정가능

7) SOL (직렬 오버 LAN)

- SOL (Serial-over-LAN)은 IPMI 세션을 통해 마더 보드의 직렬 포트로의 데이터 트래픽 리다이렉션을 나타냄
- 어느 정도까지는 BIOS 인터페이스에 대한 액세스가 가능 (직렬 리다이렉션이 구성된 경우)
- Grub와 같은 부트 로더 또는 Linux 명령 콘솔 (직렬 콘솔이 구성되어있는 경우)에 대한 액세스도 가능
- SOL은 IPMI v2.0 (RMCP +)에서 페이로드 유형으로 정의
- IPMI Serial over LAN (SOL) 문서에서 SOL을 구성하고 사용하는 방법에 대한 구체적인 예를 검색 가능

8) IPMI 의 변경 사항

■ IPMI v1.5의 변경 사항(IPMI 2.0, 5.1 절 참조)

- 직렬 / 모뎀 메시징 및 경고
- 직렬 포트 공유
- 부팅 옵션
- LAN 메시징 및 경고
- 확장 된 BMC 메시징 '채널 모델'
- 추가 센서 및 이벤트 유형
- 플랫폼 이벤트 필터링 (PEF)
- 알림 정책

■ IPMI v2.0의 변경 사항 (IPMI 2.0, 1.6 절 참조)

- 강화된 인증 (IPMI-over-IP 추가 : RMCP +)
- VLAN 지원
- SOL (Serial Over LAN)은 RMCP +의 새로운 페이로드 기능에서 사용자 정의 페이로드 유형으로 정의
- 페이로드 (RMCP +는 IPMI-over-IP 세션을 통해 IMPI 메시지 이외의 데이터를 전송가능)
- 암호화 지원 (IPMI 메시지 및 RMCP +를 통해 전송할 수 있는 기타 페이로드 유형은 암호화 가능)
- 확장된 사용자 로그인 옵션
- 펌웨어 방화벽
- SMBus 시스템 인터페이스 (SSIF는 하드웨어 인터페이스를 위한 새로운 옵션으로, SMBus 호스트 컨트롤러를 통해 BMC에 로컬 액세스 가능하며, SSI는 저렴한 BMC 구현을 지원해야함)

## 2. IPMI & BMC Disadvantage

- IPMI는 서버를 원격 관리 할 수 있는 프로토콜임
- 보편적으로 관리뿐만 아니라 응용 프로그램 및 운영 체제의 구축 및 배포, 소프트웨어 설치 등을 위해 널리 사용됨
- BMC 라고 하는 임베디드 서버는 IPMI를 구현하고 서버 마더 보드에 있음
- 일반적으로 Linux를 실행하며 자체 CPU, 메모리 및 스토리지가 거의 없음
- 또한 BMC는 전자 메일 기능, LDAP 지원, 원격 CD 및 기타 미디어 에뮬레이션 및 기타 다양한 기능과 함께 원격 웹 액세스를 제공함
- BMC는 강력하며 매우 낮은 수준에서 서버를 작동 및 제어
- 서버의 전원이 꺼진 경우에도 비트가 팬에 부딪힐 때 작동하도록 설계
- BMC 또는 IPMI를 제어하는 사람은 서버를 완전히 제어 가능

위와 같은 이유로 IPMI와 BMC는 다음과 같은 단점을 가지고 있음:

- Insecure
- Lots of OEM commands
- Scability problems

### 1) Insecure

- 임베디드 시스템의 특성으로 많은 프로그램 기능에 비하여 느린 패치가 이루어짐
- 사용자는 BMC 보안 문제를 수정하거나 패치 불가능
- 벤더는 BMC에 백도어를 설치하여 지원 담당자가 액세스 할 수 없도록 제어 가능
- 공격자는 IPMI를 이용하여 물리적 레벨에서 손쉽게 서버에 접근 가능함

- 공격자는 운영 체제 컨트롤을 우회하여 시스템을 재부팅하거나, 새 운영 체제를 설치하거나, 데이터를 손상시킬 수 있음
- IPMI 인증을 위한 암호는 일반 텍스트로 저장됨
- 하나의 IPMI 암호를 알고 있으면 IPMI 관리 그룹의 모든 컴퓨터에 대한 암호가 제공됨
- IPMI 시스템의 루트 액세스는 시스템의 하드웨어, 소프트웨어, 펌웨어를 완벽하게 제어함
- BMC 는 종종 취약한 과도한 네트워크 서비스를 실행함
- IPMI 액세스는 시스템에 대한 원격 콘솔 액세스를 허용하여 BIOS에 액세스 가능함
- BMC 가 손상되었는지를 탐지하는 데 사용할 수 있는 모니터링 도구가 거의 없음
- BMC 로부터 오는 특정 유형의 트래픽은 암호화되지 않음
- 마더 보드의 손상 없이 IPMI 암호를 제거하는 방법이 명확하지 않음
- 공격자는 일반적으로 인터넷에 연결된 대상 시스템을 쉽게 검색하고 식별 할 수 있으며 IPMI도 예외가 아님

2) Lots of OEM commands

- OEM 고유의 IPMI 명령을 추가 하는 것은 OEM에서 사용자에게 더 나은 인터페이스를 제공하기 위한 것
- 그러나, OEM 명령이 특정 마더 보드에서 작동한다는 보장이 없음
- OEM 확장은 마더 보드의 특정 하드웨어 및 펌웨어 대한 리비전을 지원하지 않을 가능성이 존재
- 확장 기능은 동일 OEM의 다른 마더 보드 라인도 지원하지 않을 가능성이 존재함

3) Scability problems

- 최근 전통적인 데이터 센터 환경은 scale-out 형태로 변화
- 즉, 단순한 형태의 서버를 대규모의 분산된 형태로 구축하는

서비스 형태가 발전함에 따라 이기종, multi-vendor 환경에 적용 가능한 standard 기반의 관리 인터페이스를 필요함

- 그러나 기존의 관리 플랫폼은 이러한 요구 사항을 만족 시키지 못함
- 오래된 표준인 IPMI의 경우 Power On/Off/Reboot, 온도, 텍스트 기반 콘솔과 같이 최소한의 공통된 관리 인터페이스만을 제공하기 때문에 확장성에 한계가 있음

## 3. Redfish - IPMI 확장 한계점 극복을 위한 방안

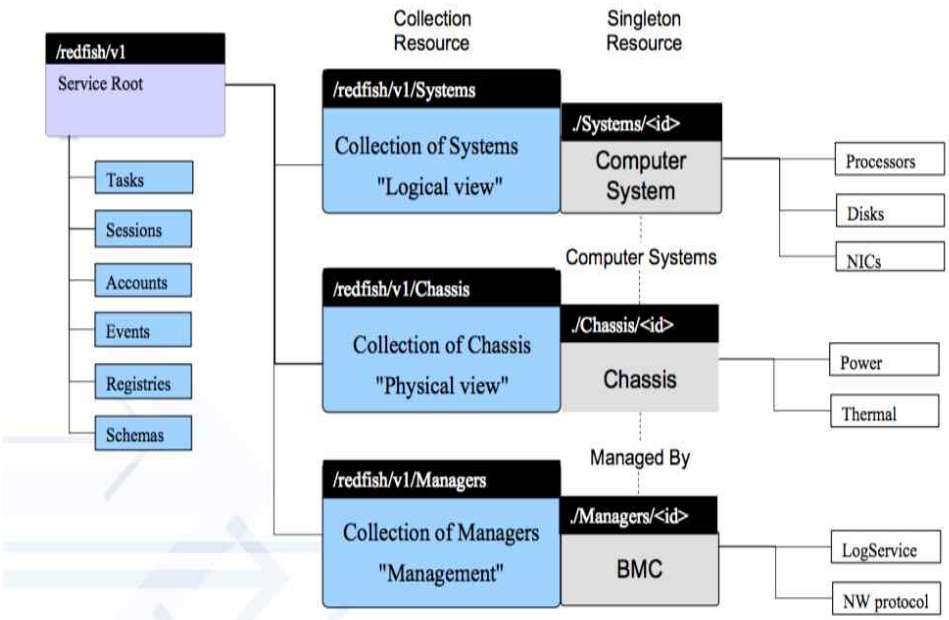
- 4절에서 제시한 것과 같이 하이퍼스케일의 서버를 사용하는 환경에서는 전통적인 엔터프라이즈 플랫폼과는 다른 새로운 관리 기법이 필요함
- 기존 IPMI는 확장 한계점을 가지고 있음
- 한계점 극복을 위한 방안 중의 하나로 DMTF(Distributed Management Task Force)의 SPMF(Scalable Platforms Management Forum)에서 제안

### 1) Redfish 기본 개념

- Redfish는 Distributed Management Task Force에서 시스템 플랫폼 관리를 위해 표준화된 형태로 제시된 REST API로 대규모의 서버 관리를 위해 Redfish는 RESTful 인터페이스와 JSON, OData를 사용함
- Redfish에서 URL은 자원, 서비스, 또는 자원의 집합을 의미
- 또한 URIs (Uniform Resource Identifiers)는 자원이나 자원을 제어하기 위한 클라이언트의 위치를 나타냄
- IPMI-over-LAN의 대체를 위한 안전한 다중 노드 기능이 있음
- 스키마 기반이지만 사람이 읽을 수 있는 출력이 가능
- OCP 원격 컴퓨터 관리 요구 사항을 충족함

Redfish의 스키마는 다음 2가지 형태로 정의됨 :

- OData Common Schema Definition Language (CSDL): 일반적인 OData tools이나 응용 프로그램에서 사용할 수 있음
- JSON Schema format: Python scripts, JavaScript와 같이 다른 환경에서 활용할 수 있음




[그림 16] Redfish Resource Map

2) Collection

Redfish에서는 Systems, Managers, Chassis와 같이 유사한 자원을 그룹화 가능함

■ System

- 컴퓨터 시스템의 논리적 view를 나타냄
- 호스트 CPU에서 접근 가능한 모든 하위 시스템을 System resource로 표기
- 모든 system instance는 CPUs, memory, 또는 많은 구성 요소를 가질 수 있다. 이러한 컴퓨터 시스템을 System collection의 하위 구성 요소로 할 수 있음

■ Manager

- BMC나 인프라를 관리할 수 있는 다양한 관리 요소를 포함
- Manager는 다양한 관리 서비스를 구성할 수 있으며 NIC 과 같은 구성 요소를 가질수 있음

■ Chassis

- 인프라의 물리적 리소스를 포함함

- 단일 Chassis 자원에는 house 센서, 팬과 같은 것이 포함가능하며 랙, 블레이드와 같은 것은 Chassis 자원의 예

3) Operation

Redfish는 GET, PUT, PATCH, POST, DELETE, HEAD와 같은 HTTP 운영 함수를 사용가능함

- GET: 데이터 검색에 사용
- POST: 자원을 생성하거나 실제 사용하는 명령어로 사용
- DELETE: 사용된 자원을 제거
- PATCH: 1개 이상의 자원 속성을 변경
- PUT: 자원을 완전히 대체하는데 사용
- HEAD: 반환되는 데이터가 없는 GET과 유사하며 URI 구조를 파악하는데 활용가능

## 4. BMC 기능별 IPMI 2.0 스펙분석의 예

■ BMC를 사용하면 데이터 센터, 컴퓨터실 또는 먼 곳에서 서버에 방문하지 않고도 대부분의 기능을 원격으로 수행 가능

■ BMC 기능사용의 몇 가지 예

- BMC 원격 콘솔과 가상 전원을 사용하면 중단된 원격 서버를 블루 스크린 상태로 볼 수 있으며 온사이트 지원이 없어도 서버를 다시 시작 가능
- BMC 가상 KVM 기술은 운영 체제와 응용 프로그램을 항상 원격으로 관리할 수 있는 고성능 원격 콘솔을 제공
- BMC 가상 CD/DVD-ROM 또는 플로피를 사용하여 워크스테이션이나 중앙 집중식 웹 서버에 있는 이미지로 네트워크 상에서 운영 체제 또는 플래시 시스템 펌웨어를 설치가능
- 내장된 상태 BMC가 서버 내의 온도를 모니터링하고 수정된 신호를 팬에 전달하여 적절한 서버냉각 상태를 유지하는 등 BMC는 서버 상태를 모니터링하고 유지 관리하는 데 활발히 관여
- BMC는 온도 모니터링과 함께 팬 상태 모니터링도 제공함

### 1) 센서

■ BMC는 IPMI 센서를 통해 환경모니터링을 제공

■ 센서는 Sensor Data Record(SDR, IPMI v2.0 33장) Repository 탐색을 통하여 시스템 구성을 찾음

■ IPMI 센서 장치 명령(IPMI v2.0 35장, 센서 장치 명령)을 통하여 엑세스 가능

| Command | Section | O/M |
|---|---|---|
| Get Device ID | 20.1 | M |
| Cold Reset | 20.2 | O |
| Warm Reset | 20.3 | O[3] |
| Get Self Test Results | 20.4 | M |
| Manufacturing Test Mode On | 20.5 | O |
| Broadcast Get Device ID | 20.6 | M |
| reserved | - | - |
| Device Specific Commands | - | - |
| Get Device SDR Info | 35.2 | O |
| Get Device SDR | 35.3 | O[5] |
| Reserve Device SDR Repository | 35.4 | O[5] |
| Get Sensor Reading Factors | 35.5 | O[2] |
| Set Sensor Hysteresis | 35.6 | O |
| Get Sensor Hysteresis | 35.7 | O |
| Set Sensor Threshold | 35.8 | O |
| Get Sensor Threshold | 35.9 | O[4] |
| Set Sensor Event Enable | 35.10 | O |
| Get Sensor Event Enable | 35.11 | O[4] |
| Re-arm Sensor Events | 35.12 | O[3] |
| Get Sensor Event Status | 35.13 | O |
| reserved | - | - |
| Get Sensor Reading | 35.14 | M |
| Set Sensor Type | 35.15 | O |
| Get Sensor Type | 35.16 | O[4] |
| Set Sensor Reading and Event Status | 35.17 | O |
| Set Event Receiver | 29.1 | M[1] |
| Get Event Receiver | 29.2 | M[1] |
| Platform Event (a.k.a. Event Message) | 29.3 | M[1] |

Notes:
1. - Mandatory for Event Message Generators only
2. - Mandatory for Non-linear Sensors
3. - Mandatory for manual re-arm Sensors
4. - Mandatory if corresponding 'Set' command is implemented.
5. - Mandatory per information returned in *Get Device SDR Info*

[그림 17] 센서 장치 명령

- IPMI는 각 SDR에서 16바이트 문자열 식별자를 제공(ASCII기반 문자열은 시스템 관리 소프트웨어로 해석)
- 이러한 토큰은 SDR 문자열 식별자 필드에서 사용되면 SDR 의 다른 필드와 함께 필요한 특성을 제공

2) 외부 이벤트 생성

- BMC는 시스템의 다른 이벤트 수신자에게 경고를 전송하도록 구성(IPMIv2.0 29장, 이벤트 명령) 참조
- 이벤트 생성자에 필수적인 모든 이벤트 명령은 BMC 펌웨어에서 구현됨

3) LAN 메세징

- BMC는 온칩 NIC를 사용하여 LAN 상에서 IPMI 메시징을 지원함
- IPMI 메시지 RMCP 패킷으로 전송되며, BMC가 메시지에 수신하고 응답할 온칩 NIC를 통해 메시지 전달(자세한 사항은 IPMIv2.0 사양 13장 IPMI 인터페이스를 참조)
- LAN 세션을 지원(단일 사용자의 경우에는 동시에 지원하는 세션 수에 제한)
- 모든 필수 IPMI LAN 메시징 명령은 BMC 핵심 펌웨어에서 구현
- RMCP+는 IPMI 2.0 이하에서 로토콜 보정은 모든 RMCP 형식을 준수
- 확장명은 여러 페이로드 유형,
- 암호화가 가능 (IPMIv2.0 13.3절 RMCP+를 참조)
- IPMI 펌웨어는 아래 표와 같은 선택적 암호화 방법을 지원(ID 0~14)

| ID | 특징 | 암호 그룹 | 인증 알고리즘 | 통합 알고리즘 | 기밀성 알고리즘 |
|---|---|---|---|---|---|
| 0 | " no password" | 00h, 00h, 00h | RAKP-none | 없음 | 없음 |
| 1 | S | 01h, 00h, 00h | RAKP-HMAC-SHA1 | 없음 | 없음 |
| 2 | S, A | 01h, 01h, 00h | | HMAC-SHA1-96 | 없음 |
| 3 | S, A, E | 01h, 01h, 01h | | | AES-CBC-128 |
| 4 | S, A, E | 01h, 01h, 02h | | | xRC4-128 |
| 5 | S, A, E | 01h, 01h, 03h | | | xRC4-40 |
| 6 | S | 02h, 00h, 00h | RAKP-HMAC-MD5 | 없음 | 없음 |
| 7 | S, A | 02h, 02h, 00h | | HMAC-MD5-128 | 없음 |
| 8 | S, A, E | 02h, 02h, 01h | | | AES-CBC-128 |
| 9 | S, A, E | 02h, 02h, 02h | | | xRC4-128 |
| 10 | S, A, E | 02h, 02h, 03h | | | xRC4-40 |
| 11 | S, A | 02h, 03h, 00h | | MD5-128 | 없음 |
| 12 | S, A, E | 02h, 03h, 01h | | | AES-CBC-128 |
| 13 | S, A, E | 02h, 03h, 02h | | | xRC4-128 |
| 14 | S, A, E | 02h, 03h, 03h | | | xRC4-40 |

Key:
S = 인증된 세션 설정(세션 설정에 필요한 올바른 역할, 사용자 이름 및 암호/키)
A = 인증된 페이로드 데이터가 지원됨
E = 인증 및 암호화된 페이로드 데이터가 지원됨

[그림 18] IPMI의 선택적 암호 지원 방법

4) PEF(플랫폼 이벤트 필터링) 및 경고 정책

- PEF(IPMIv2.0 17장 플랫폼 이벤트 필터링)가 지원되어 구성된 시스템 이벤트에 대한 응답에서 경고가 가능
- 사용자가 추가적인 필터나 경고 정책을 원하는 경우에는 Set PEF 구성 매개변수 명령을 사용하여 구성해야 함 (IPMIv2.0 30.3절 참조)

5) 펌웨어 방화벽

- IPMI v2.0 사양 21장을 참조

# [별첨] 참고문헌


[1] BMC를 이용한 컴퓨터 관리 기술
(http://www.itfind.or.kr/WZIN/jugidong/1375/file52073-137501.pdf)

[2] AMI_Aptio_5.x_UEFI_Intro_Training_NDA.pdf

[3] AMI_Aptio_5.x_SEC_and_PEI_Training_NDA.pdf

[4] AMI_Aptio_5.x_DXE_and_BDS_Training_NDA.pdf

[5] IPMI v2.0 rev. 1.0 specification markup for IPMI v2.0/v1.5 errata revision 4

[6] http://www.intel.com/design/servers/ipmi/ Information regarding IPMI from Intel

[7] Server Monitoring and Management using IPMI (from ADMIN-Magazin 03/2010)

[8] https://www.cern.ch/it-dep-fio-ds/Presentations/2004/ipmi_server_management.ppt

[9] https://www.thomas-krenn.com/en/wiki/IPMI_Basics

[10] A Penetration Tester's Guide to IPMI and BMCs
https://blog.rapid7.com/2013/07/02/a-penetration-testers-guide-to-ipmi/)

[11] Dan Farmer's IPMI++ Security Best Practices
(http://fish2.com/ipmi/remote-pw-cracking.html)

[12] http://www.gnu.org/software/freeipmi/

[13] "Introduction to Redfish," Jeff Autor, co-ChairDMTF Scalable Platforms Management Forum, May 2015.

[14] HP MicroServer BMC 펌웨어 기능 매뉴얼(https://www.hpe.com)