EFI – 확장 펌웨어 인터페이스

n0fate

feedbeef.blogspot.com

개요

1. 확장 펌웨어 인터페이스
2. 기존 시스템과 비교
3. EFI 기반 부팅 과정
4. EFI 개발 환경 구축하기
5. EFI 애플리케이션 개발 예제
6. 참조 문서

확장 펌웨어 인터페이스

- 확장 펌웨어 인터페이스의 정의
- EFI의 중요성

확장 펌웨어 인터페이스

- **운영체제와 플랫폼 펌웨어간의 소프트웨어 인터페이스를 정의하는 규격**
  - 역사적으로 오래된 바이오스 인터페이스를 대체하기 위해 나온 표준 규격
  - 인텔 아이테니엄 개발 과정에서 문제점 제기
    - ✓ 서버 플랫폼 채용의 한계
    - ✓ 즉, 최초엔 인텔의 주도하에 규격 제정

- **UEFI?**
  - EFI를 다양한 기업에서 참여하여 규격을 정의하는 United EFI Forum으로 이전
  - 참여 업체
    - ✓ AMD, AMI, Apple, Dell, HP, IBM, Insyde, Intel, Lenovo, MS, Phoenix가 참여
  - 현재 UEFI 규격은 EFI 1.1.0 규격을 기반으로 작성되고 있음.
  - EFI Services - Boot Services, Runtime Services로 나눌 수 있음

확장 펌웨어 인터페이스의 장점

- 재사용이 가능함
- 확장성과 모듈성
  - 코어 EFI 기능은 펌웨어에 정의
  - 서드파티 드라이버는 디스크나 펌웨어에 존재
- 고수준 언어 개발 환경에 따른 생산성 증가
  - 부트로더/드라이버는 C로 작성
  - 간단하게 정의된 인터페이스를 통한 통신
  - EFI Byte Code (EBC)를 인터프리터를 활용

확장 펌웨어 인터페이스의 정의

- **EFI 부팅은 크게 3 과정으로 나뉨**
  - 플랫폼 초기화 과정
    - ✓ 하드웨어 EFI 칩 사용, 메인보드 제조사가 정의
  - 부트 장치 선택 과정
    - ✓ FAT32 File System으로 된 EFI 파티션 접근
    - ✓ 플랫폼 개발사/소프트웨어 개발사가 정의
  - 운영체제 부팅 과정
    - ✓ 운영체제 개발사가 정의
    - ✓ EFI Image 로드

- **EFI Image**
  - LoadImage() 부트 서비스에 의해 메모리에 로드
  - PE32+ 포맷

확장 펌웨어 인터페이스 구동 환경

- **지원하는 CPU**
  - **Intel x86, ARM, Itanium**

- **지원하는 운영체제**
  - Microsoft Windows
    - ✓ Intel x86 - Windows Vista SP1, Windows 7, Windows Server 2008
    - ✓ Itanium – Windows XP, Windows Server 2003
  - Linux
    - ✓ ELILO based Linux
  - Mac OS X
    - ✓ Mac OS X 10.4 (Panther)

EFI의 중요성

- **최신 운영체제에서 지원하며, 필수 요소가 될 수 있음.**
  - Apple Mac 하드웨어에 기본 탑재
  - Windows 8의 부팅 속도 상승 효과의 이유가 EFI임이 알려짐에 따라 수요 증가 예상
  - 2테라바이트 이상의 단일 하드 디스크 지원
- **기존의 보안 이슈가 그대로 이어질 수 있음.**
  - BIOS 와 MBR을 대체하는 개념으로 동일한 이슈 발생 가능
  - Hacking the EFI
    - ✓ Hacking the Extensible Firmware Interface (Black Hat 2007 USA)
      - https://www.blackhat.com/presentations/bh-usa-07/Heasman/Presentation/bh-usa-07-heasman.pdf
    - ✓ De Mysteriis Dom Jobsivs – Loukas (Syscan Singapore 2012 예정)
      - http://beistlab.com/2012/02/27/syscan-singapore-2012/
    - ✓ Rootkit Arsenal에서도 간단히 언급하였음

기존 시스템과 비교

- BIOS vs EFI
- MBR vs GPT

BIOS vs EFI

- **바이오스**
  - IA-32 Real Mode로 구동
    - ✓ 16비트 주소체계 사용
    - ✓ 물리 메모리 주소는 최대 20비트의 크기만 가질 수 있음
      - 물리 메모리 상의 정적인 1메가바이트 크기의 주소 공간을 항상 예약해서 사용해야 함
    - ✓ 하나의 세그먼트 당 16비트의 크기를 가질 수 있음
      - 코드 세그먼트는 최대 64KB까지 밖에 지원할 수 없음
    - ✓ 지원하는 부트 장치가 미리 정의되어 있으며, 새로운 장치에 대한 부팅 지원에 시간이 오래 걸림

BIOS vs EFI

- **EFI**
  - IA-32 Protected Mode로 구동
    - ✓ 프로세서 모드에 따라 다른 주소체계를 사용하여 Pre-Boot Execution 환경에서 동작하는 애플리케이션이 해당 크기만큼의 메모리 주소에 접근할 수 있음.
      - 32Bit Addressing – x86-32, ARM
      - 64bit Addressing – x86-64, Itanium
    - ✓ UEFI는 펌웨어와 운영체제의 비트가 일치해야 함
      - OS가 64비트이면 UEFI 바이너리도 64비트여야 함
      - Mac OS X는 이 문제를 해결하기 위해 Universal Binaries에 32/64비트 EFI 바이너리를 포함하고 있음

MBR vs GPT

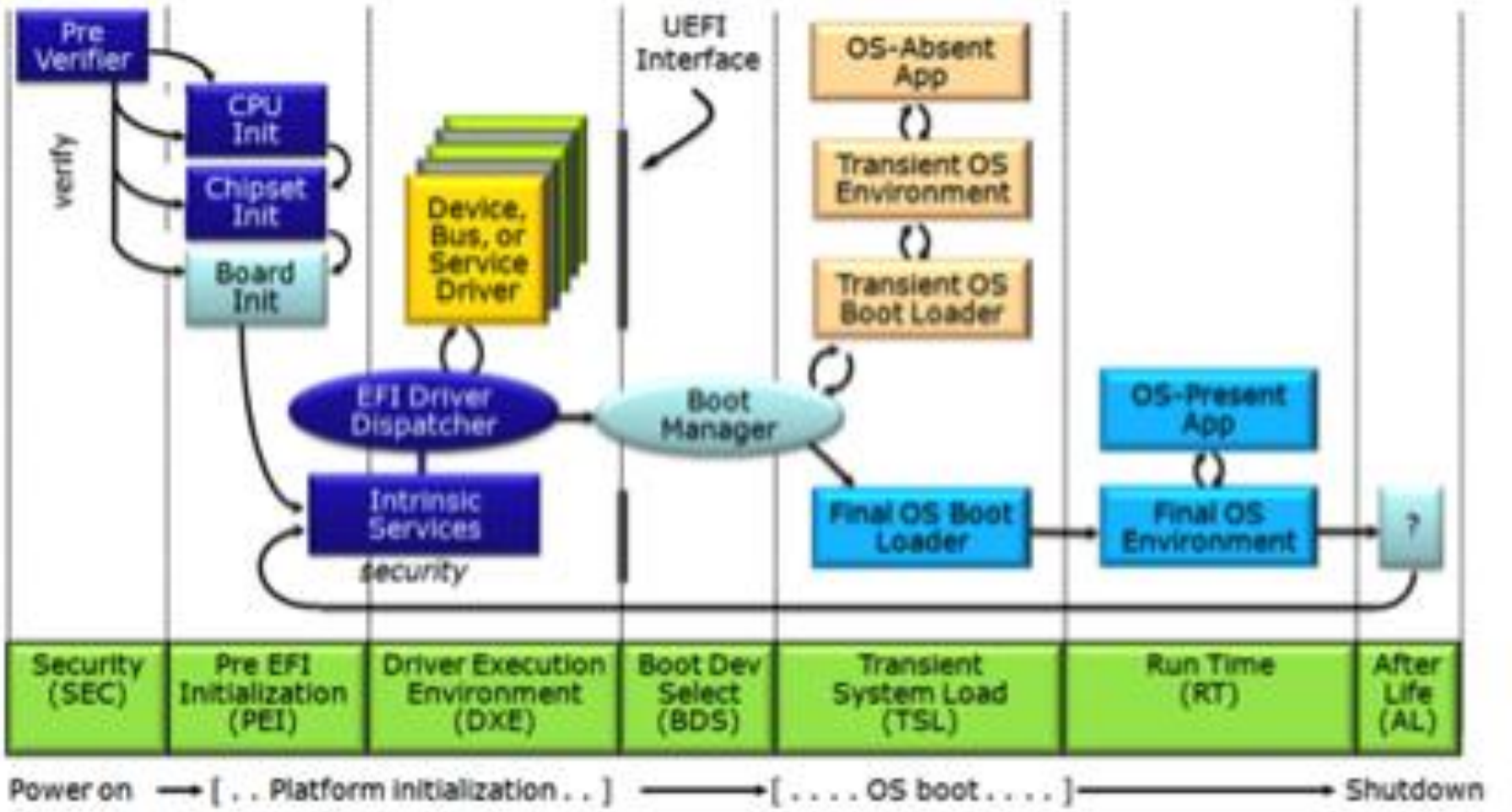
- **MBR**
  - 하드디스크 첫 번째 섹터(512바이트)에 위치함.
    - ✓ 448바이트의 코드 영역
      - 부팅 가능한 파티션 검색
      - 부트 파티션의 부트 섹터 호출
      - 에러 메시지 출력
    - ✓ 16바이트 파티션 테이블 구조체 4개 → 64바이트
      - 부팅 가능한 주 파티션은 최대 4개가 위치할 수 있음.
      - CHS → LBA 섹터 어드레싱 기법 사용
    - ✓ 2바이트의 시그너처
      - 0x55AA

MBR vs GPT

- **GPT**
  - 128개의 주 파티션 생성 가능
  - 64비트 파티션 크기 지원 – 8.04 ZB
  - CRC 체크

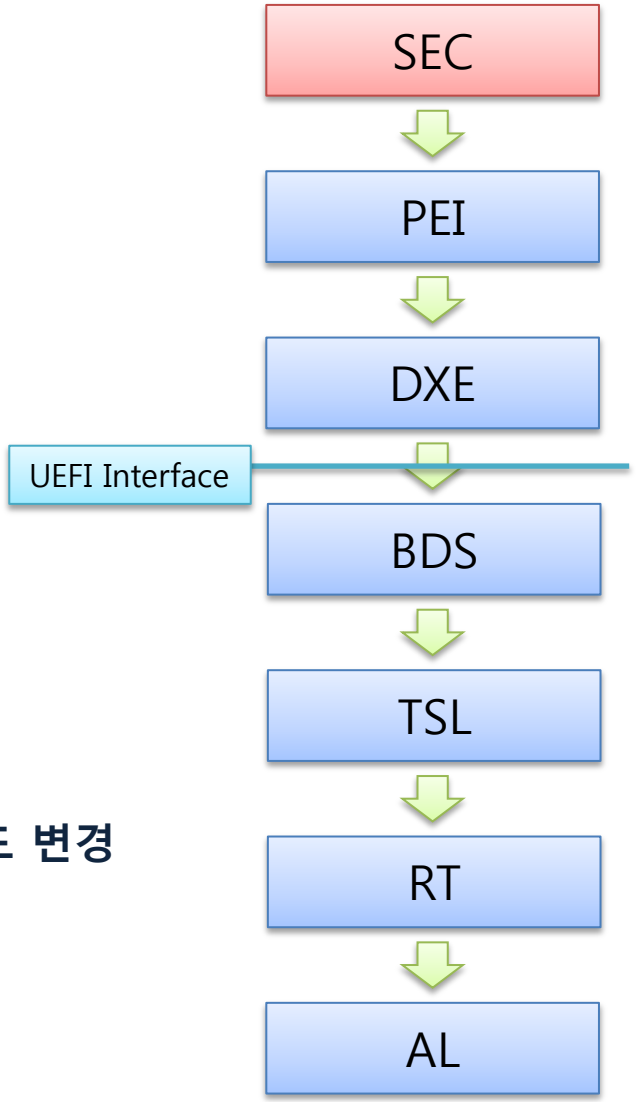
EFI 기반 부팅 과정

EFI 기반 부팅 과정


Pre Verifier
verify
CPU Init
Chipset Init
Board Init
Device, Bus, or Service Driver
UEFI Interface
OS-Absent App
Transient OS Environment
Transient OS Boot Loader
EFI Driver Dispatcher
Boot Manager
OS-Present App
Intrinsic Services
security
Final OS Boot Loader
Final OS Environment
?
Security (SEC)
Pre EFI Initialization (PEI)
Driver Execution Environment (DXE)
Boot Dev Select (BDS)
Transient System Load (TSL)
Run Time (RT)
After Life (AL)
Power on
[. . Platform initialization . .]
[. . . . OS boot . . . .]
Shutdown

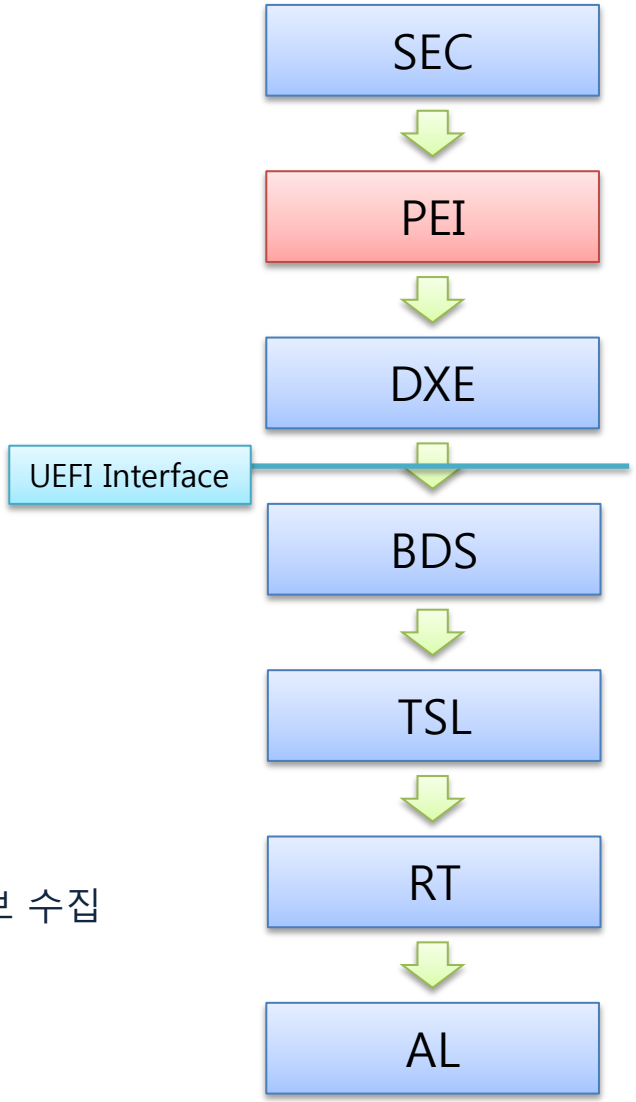
Security (SEC) Phrase

- **전원을 키는 시점에 진입**
- **플랫폼 초기화 단계를 위한 환경을 제공 함**
  - PEI 과정을 위해 캐시된 데이터와 스택 영역을 설정
  - 모든 플랫폼 재시작 이벤트를 핸들링 함.
  - PEI 과정의 하드웨어 모듈을 검증 함(root of trust)
  - PEI 단계에 필요한 정보를 전달
  - PEI로 제어를 전달
- **진행 과정에서 부트 스트랩 프로세서(BSP)가 메모리 모드 변경**
  - IA-32 Real Mode → IA3-2 Protected Mode



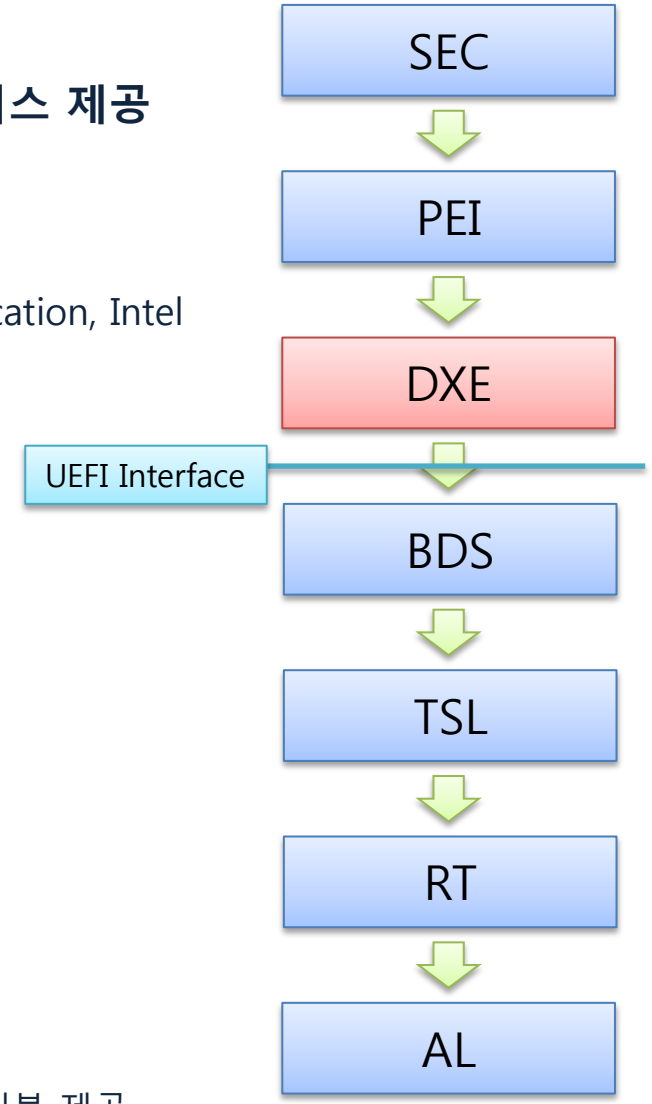
Pre-EFI Initialization(PEI) Phrase

- **하드웨어 자원 초기화 역할을 수행**
  - 메모리와 플랫폼 자원 초기화
  - 부트 모드 식별(recovery, S2 resume, normal boot)
  - 저수준의 하드웨어 모듈 로드
  - DXE Core 실행
- **PEI 과정의 주 컴포넌트가 기본 서비스를 제공**
  - 주 컴포넌트의 서비스 상에 PEI 모듈을 디스패칭함
  - 플랫폼 정보를 수집
  - 메모리에 있는 Hand Off Block 구조체 리스트에서 설정 정보 수집
  - PEI에서 DXE로 수집한 정보를 전달



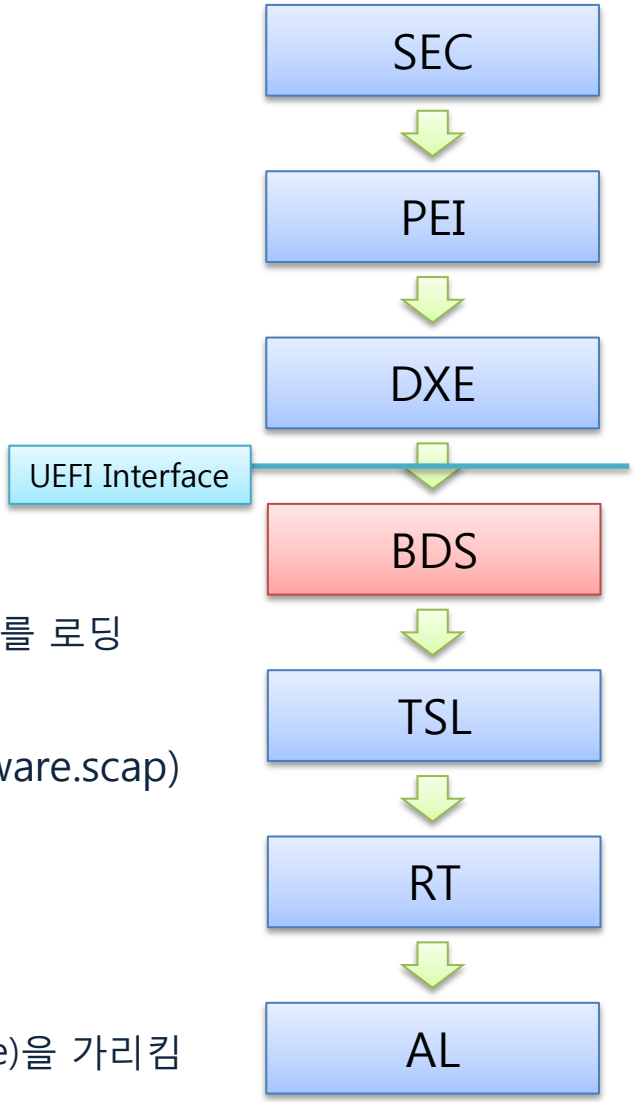
Driver Execution Environment Phrase

- **플랫폼을 초기화하고 운영체제 부팅을 위해 필요한 서비스 제공**
- **세부 설명은 다음 문서를 참조하기 바람**
  - UEFI Driver Execution Environment Core Interface Specification, Intel
- **주요 컴포넌트**
  - DXE Core – 부트, 런타임, DXE 서비스를 제공
  - DXE Dispatcher – DXE 드라이버를 식별하고 실행함
  - DXE Drivers
    - ✓ 프로세서, 칩셋, 플랫폼 컴포넌트 초기화
    - ✓ DXE 코어 추상화를 위해 아키텍처 프로토콜(AP)를 제공함
  - EFI System Table
    - ✓ EFI 서비스, 설정, 핸들, 콘솔 장치 접근을 위한 포인터 테이블 제공



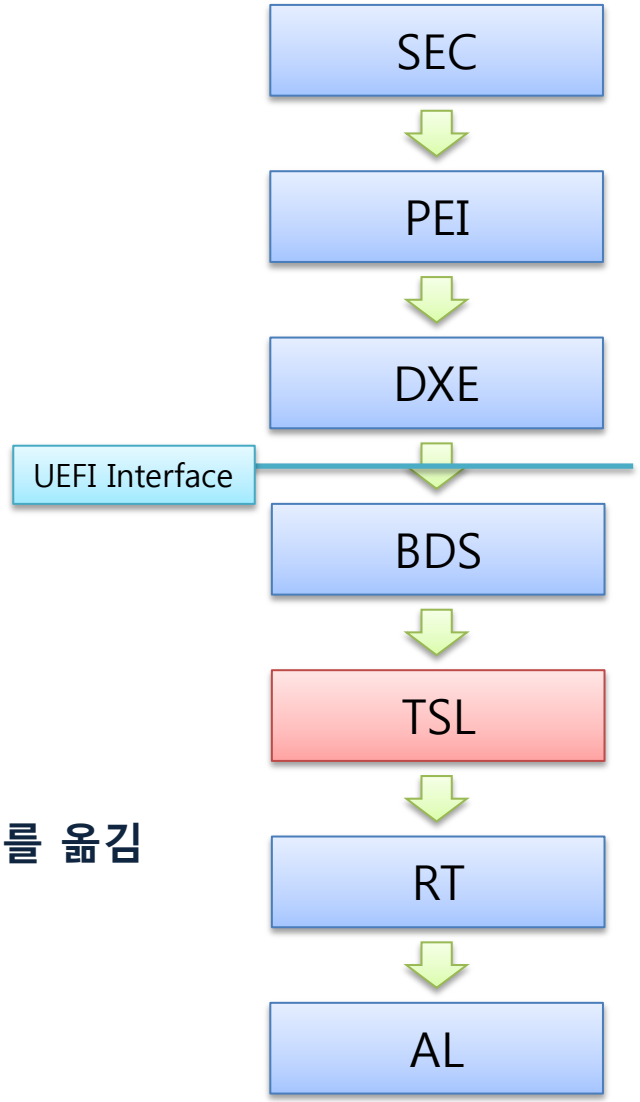
Boot Device Select(BDS) Phrase

- **부트 옵션을 실행하는데 필요한 의사 결정을 수행**
- **콘솔 장치를 초기화하고 디바이스 드라이버를 로드**
- **Boot Manager**
  - EFI 부트로더, EFI 드라이버, EFI 애플리케이션 로드
  - NVRAM 변수를 확인하여 부팅 시점에 할 일을 결정
  - 파일 시스템의 부트로더에 접근하거나, EFI에 정의된 이미지를 로딩
- 애플은 이 시점에 커스터마이징된 펌웨어를 사용함(firmware.scap)
  - Option 키를 누르고 부팅 시 사용할 부트 볼륨을 선택
  - 기본 부트 볼륨은 NVRAM에 저장되어 있음
  - HFS+ 볼륨의 헤더에는 로드할 EFI 애플리케이션(blessed file)을 가리킴



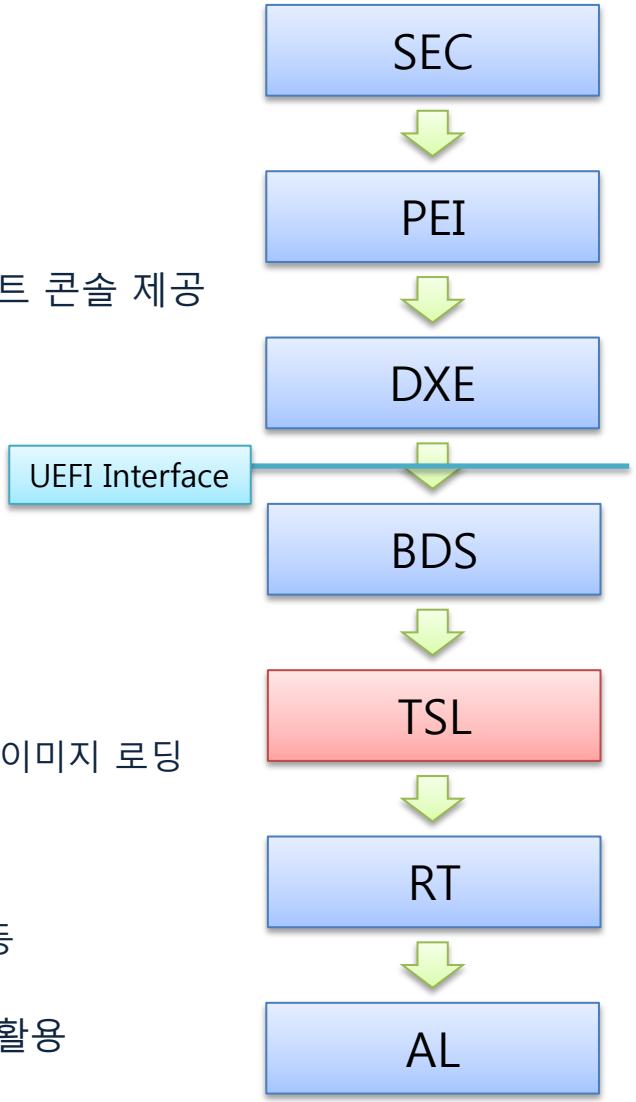
Traiencient System Load(TSL) Phrase

- **부트로더 검색**
  - Mac OS X – "/System/Library/CoreServices/boot.efi"
  - Bless 명령으로 확인할 수 있음.
- **펌웨어가 로드되면서 회색 애플로고가 화면에 나타남.**
  - 펌웨어는 Boot Services를 제공함.
  - 이 로더는 Darwin Kernel을 디스크에서 로드
  - 로드 과정에 필요한 KEXT를 구동
- **펌웨어의 ExitBootServices()가 실행되면서, 커널로 제어를 옮김**
  - NVRAM의 매개변수 정보를 수집하여 커널에게 전달
  - 펌웨어를 메모리에서 언로드



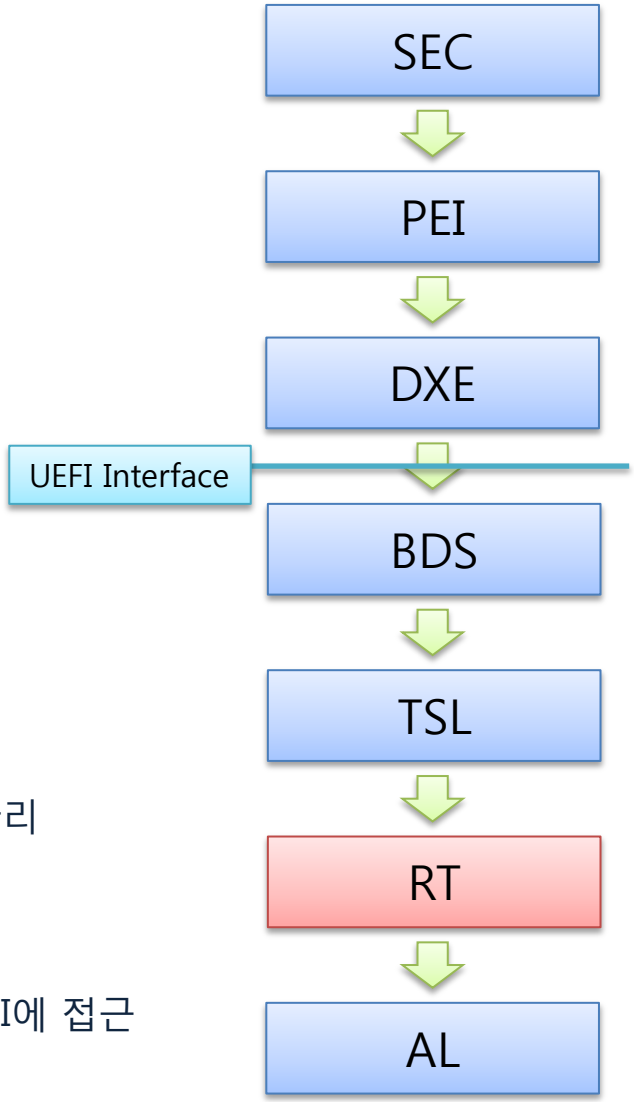
Traiencient System Load(TSL) Phrase

- **Boot Services**
  - ExitBootServices() 가 호출되기 전까지 유지되는 서비스
  - 장치와 버스(bus), 블록, 파일 서비스에 대한 그래픽 및 텍스트 콘솔 제공
  - 제공하는 서비스
    - ✓ 이벤트, 타이머, 작업 우선 순위
    - ✓ 메모리 할당
    - ✓ EFI 프로토콜 핸들링
    - ✓ EFI 애플리케이션, 부트 서비스 드라이버, 런타임 드라이버 이미지 로딩
    - ✓ 하드웨어 왓치독 타이머
    - ✓ EFI 시스템 테이블 엔트리 수정, 데이터 버퍼 체크섬 계산 등
  - 운영체제 로더는 커널을 위한 메모리를 할당하는데 서비스 활용
  - 운영체제 로더가 ExitBootServices()를 호출하여 서비스를 종료함.



Runtime (RT) Phrase

- **운영체제가 구동 중인 과정**
- **EFI Runtime Services를 통해 EFI에 접근 가능**
  - ExitBootServices() 호출에 상관없이 존재함
  - Mac OS X
    - ✓ EFI Runtime Kernel Extension(AppleEFIRuntime.kext)
    - ✓ EFI Runtime KEXT Plugin (AppleEFINVRAM.kext)
- **런타임 서비스 관리 내용**
  - EFI 애플리케이션과 EFI 환경 간의 정보 공유를 위한 변수 관리
  - 하드웨어 시간 장치 관리
  - 가상 메모리 – 운영체제가 가상 메모리 주소 체계를 통해 EFI에 접근
  - 플랫폼 시퀀스 카운터를 얻고, 리셋 기능을 제공함.



EFI 개발 환경 구축하기

- EDK Installation
- Build the NT32 Framework Environment
- Building the EFI Toolkit for NT32
- Run the EDK Framework NT32

EDK Installation

- **Windows OS에서 개발을 위해선 Visual Studio가 설치되어 있어야 함**
  - 보통 가이드라인에서 설명하는 개발 환경은 .Net 2003, 2005 환경
  - 최신 운영체제인 MS Windows 7 환경의 Visual Studio 2010에서 진행하였음.

- **최신 EDK 소스를 다운로드**
  - Tianocore.org에서 다운로드 가능
  - 현재 최신 소스는 Edk 1.06.zip
    - ✓ UEFI 2.1, Framework 0.9, PI 1.0, EFI shell, Core and sample driver source code for EDK build source code environment
  - 압축을 해제하여 원하는 디렉터리에 위치한 후 환경 변수 설정
    - ✓ C:₩EDK 에 압축 해제
    - ✓ 시스템 환경 변수 – EDK_SOURCE = C:₩EDK

Build the NT32 Framework Environment

- Visual Studio Command Prompt를 띄우고 다음 디렉터리로 이동
  - C:₩EDK₩Sample₩Platform₩Nt32₩Build

- nmake를 입력
  - 빌드 중 'fatal error: 0x02를 반환하였습니다' 메시지 발생 시, config.env를 에디터로 열어 다음을 수정 함
    - ✓ 'USE_VC8 = NO' → YES
  - 다시 nmake를 입력

Build the EFI Toolkit for NT32

- Tianocore.org에서 EFI Toolkit 2.0.0.1을 다운로드
- 다음의 과정을 진행
  - 압축을 해제하여 원하는 곳에 위치 함
    - ✓ C:₩EFI_Toolkit_2.0 – EFI Toolkit SDK Root
  - Visual Studio Command Prompt를 띄우고 EFI Toolkit SDK Root 로 이동
  - 하위의 Build/NT32 디렉터리의 sdk.env의 다음 요소를 추가
    - ✓ C_BUILD_FLAGS=/nologo /W3 /GS- /Gm/ Zi /Od /GF /Gy /Qifist /Gs8192
  - 다시 EFI Toolkit SDK Root로 이동해서 다음과 같이 입력함
    - ✓ build NT32
    - ✓ nmake

Run the EDK Framework NT32

- NT32 프레임워크에 로컬 디스크와 폴더를 마운트할 수 있음.
  - 프레임워크 빌드한 위치로 이동함
    - ✓ C:₩EDK₩Sample₩Platform₩Nt32₩Build
  - System.cmd 파일을 다음과 같이 수정
    - ✓ 창 타이틀 변경
      - Set EFI_WIN_NT_UGA="Example 1"
    - ✓ 파일시스템의 특정 디렉터리를 맵핑
      - Set EFI_WIN_NT_FILE_SYSTEM=C:₩VirtualEDK
      - 현재 디렉터리는 . 으로 표현
  - System.cmd 파일을 실행

Run the EDK Framework NT32

- EDK 프레임워크 환경을 구동
  - 프레임워크 빌드한 위치로 이동함
    - ✓ C:₩EDK₩Sample₩Platform₩Nt32₩Build
  - nmake run
    - ✓ EFI 애플리케이션이나 드라이버 구동 테스트를 위한 EFI Shell이 실행
- 샘플 프로그램 구동
  - C:₩EDK₩EFI_Toolkit_2.0₩build₩nt32₩bin 에 있는 EDIT.EFI를 복사
    - ✓ C:₩EDK₩Sample₩Platform₩nt32₩build₩IA32
  - EFI Shell을 띄우고 다음을 입력
    - ✓ fsnto: - EFI_NT_FILE_SYSTEM 환경변수에 해당하는 디바이스
    - ✓ edit.efi

Run the EDK Framework NT32

EFI 애플리케이션 개발 예제

EFI 애플리케이션 개발 예제

- **EFI 애플리케이션은 EFI Toolkit 에서 개발**
  - Makefile과 Source Code가 필요 함

- **EFI Toolkit SDK Root 아래 apps에 두 파일을 생성함. – Source, Makefile**
  - Makefile – hello.mak (다른 샘플 애플리케이션을 참조하여 작성)

```
!include $(SDK_INSTALL_DIR)\build\$(SDK_BUILD_ENV)\sdk.env

BASE_NAME=hello
IMAGE_ENTRY_POINT = InitializeHelloApplication

#Globals needed by master.mak
TARGET_APP = $(BASE_NAME)
SOURCE_DIR = $(SDK_INSTALL_DIR)\apps\$(BASE_NAME)
BUILD_DIR = $(SDK_BUILD_DIR)\apps\$(BASE_NAME)

#include paths
!include $(SDK_INSTALL_DIR)\include\$(EFI_INC_DIR)\makefile.hdr
INC= -I $(SDK_INSTALL_DIR)\include\$(EFI_INC_DIR) \
     -I $(SDK_INSTALL_DIR)\include\$(EFI_INC_DIR)\$(PROCESSOR) $(INC)

all : dirs $(LIBS) $(OBJECTS)
OBJECTS = $(OBJECTS) $(BUILD_DIR)\$(BASE_NAME).obj
$(BUILD_DIR)\$(BASE_NAME).obj : $(*B).c $(INC_DEPS)
!include $(SDK_INSTALL_DIR)\build\master.mak
```

EFI 애플리케이션 개발 예제

- **EFI 소프트웨어 개발 방법**
  - 라이브러리를 활용하는 방법
    - ✓ C언어 방식, 어셈블리 방식(FASM)을 지원함,
  - EFI 시스템 테이블을 직접 접근하여 작성하는 방법

- **본 예제에선 라이브러리를 활용하여 개발하였음.**

```
#include "efi.h“
#include "efilib.h“
EFI_STATUS
InitializeHelloLibApplication (
    IN EFI_HANDLE           ImageHandle,
    IN EFI_SYSTEM_TABLE     *SystemTable
    )
{
    InitializeLib (ImageHandle, SystemTable);
    Print(L"\n\n\nHelloLib application started\n\n\n");
    Print(L"\nHit any key to exit this image\n");
    WaitForSingleEvent(ST->ConIn->WaitForKey,0);
    ST->ConOut->OutputString (ST->ConOut, L"\n\r\n\r");
    return EFI_SUCCESS;
}
```

EFI 애플리케이션 개발 예제

- 프로그램 빌드 및 실행
  - EFI Toolkit을 빌드할 때와 동일한 방법으로 빌드 진행
  - EFI 쉘에서 hello.efi를 실행

```
fsnt0:\> hello.efi
Hello Application Started


Hit any key to exit
_
```

참조 문서

- **UEFI Forum – United Extensible Firmware Interface Forum**
  - *http://www.uefi.org*
- **rEFIt – An EFI Boot Menu and Toolkit**
  - *http://refit.sourceforge.net*
- **EDK II User Manual, Revision 0.7, Intel**
- **Intel Boot Loader Development Kit Version 2.0, Intel**
- **EDK II Module Writer, Intel**
- **EDK II C Coding Standards Specification (Draft for Review), Intel**
- **File System/MBR/GPT**
  - *http://forensic-proof.com/wp-content/uploads/2010/10/FP_File_System_MBR_GPT.pdf*

Q & A