Using the UEFI Shell
October 2010 – UEFI Taipei Plugfest
San Francisco Cable Car
Agenda

- Insyde UEFI Support
- UEFI Shell 2.0 – What is it?
- UEFI Shell 2.0 – Unique Features
- Network Browsing Example Application
- ACPI Testing Example Application
- Summary
Insyde UEFI Support
Insyde UEFI Support - TODAY

- Many Insyde customers are shipping their 4th generation of EDK1117 UEFI based BIOS
  - 5th generation preparing for Mass Production NOW

- InsydeH2O® UEFI BIOS
  - The most shipped UEFI based BIOS
  - Over 100 million clients and servers on EDK1117 codebase

- Next two mainstream platform generations will continue to use EDK1117

At the same time – Insyde is developing for the Future
The industry will evolve from EDK1117 to UDK2010 (EDK II)

Some Insyde customers are shipping InsydeH2O EDK II BIOS now!

Insyde is fully engaged and aligned with Intel Client, Server, and SSG on UDK2010
  - Active Intel development partner on UDK2010

Insyde will provide a smooth transition from EDK1117 to UDK2010
  - Adding support for the new specifications
  - Improving the industry
UEFI Shell 2.0 – What is it?
What is the UEFI 2.0 shell

- An interactive BIOS extension
- Provides environment for running programs
- Scripting interpreter to execute script files
- Bootable from external storage devices
- Optionally included as boot device in BIOS
- Similar to MS-DOS or Linux command line
- Has some built-in commands
  - File manipulation, driver management, device access, informational, memory access, BIOS status, scripting control
Shell Apps vs. UEFI Drivers

- UEFI core provides services and protocols
- Drivers and Applications use UEFI services

- Drivers
  - Have higher priority
  - Usually stay resident

- Applications
  - Written to perform a task
  - Expected to exit after completing the task
The UEFI Shell 2.0 Architecture

Shell Library
Shell Applications
Shell Scripts

Command-line Parser
Shell Protocol
Shell Console Parser
Command Launcher

Script Processor
Level 3 Command Set
Profile Command Set

Shell Core

Insyde UEFI Driver
UEFI Driver
OEM UEFI Driver

Console Driver(s)

UEFI / PI Interfaces

CPU Modules
Chipset Modules

Hardware
Using the Shell

- **Shell applications**
  - Compiled C programs use Shell or UEFI protocols

- **Shell scripts**
  - Automated shell commands, shell apps, UEFI apps, or other shell scripts
  - Complex FOR, IF, and GOTO control logic

- **Start Shell apps or scripts from the console**
  - The shell can be compiled to start an app automatically
UEFI Shell 2.0
Unique Features
Differences between EFI & UEFI 2.0 Shell

- EFI and UEFI 2.0 Shell scripts are compatible
- Additional features in UEFI 2.0 Scripts
  - Query if commands are available
  - Command features are consistent with EFI Shell

- Old Shell Protocols deprecated
- UEFI Shell Protocols added
  - EFI Shell extensions require porting
  - UEFI applications will work

- Use the UDK2010 Shell Lib to support both Protocols
Manage firmware image size

- **Shell Levels** manage main features
  - Level 0 – Launching a single application
  - Level 1 – Adds scripting
  - Level 2 – Adds file manipulation
  - Level 3 – Adds UI and information retrieval

- **Shell Profiles** manage additional commands
  - Install – Adds OS loader configuration
  - Debug – Adds debug
  - Driver – Adds driver manipulation
  - Network – Adds network configuration & test
Internet Browsing Example
Internet Browsing

- Extends pre-boot space onto Internet
- Network Browsing Examples:
  - OEM or IT department support page
    - Help pages
    - Http download client
    - Access to OS recovery images
  - Remote assist system
    - System drivers download from OEM service site
    - Remote system diagnostic
    - Hardware support page
Network Browser Example

```
load Image fs0:\net\Dpc.efi loaded at 30032000 - Success
net> load mmp.efi
load Image fs0:\net\Mmp.efi loaded at 30025000 - Success
net> load arp.efi
load Image fs0:\net\Arp.efi loaded at 30A4F000 - Success
net> load ip4.efi
load Image fs0:\net\Ip4.efi loaded at 30A36000 - Success
net> load ip4config.efi
load Image fs0:\net\Ip4Config.efi loaded at 30A2F000 - Success
net> load udp4.efi
load Image fs0:\net\Udp4.efi loaded at 30A1F000 - Success
net> load tcp4.efi
load Image fs0:\net\Tcp4.efi loaded at 30A03000 - Success
net> load dhcp4.efi
load Image fs0:\net\Dhcp4.efi loaded at 309F9000 - Success
net> load simplesocket.efi
load Image fs0:\net\SimpleSocket.efi loaded at 309F0000 - Success
net> ifconfig -s eth0 dhcp
Create an IP and start to get the default address
Please wait, you console may stop responding for a while ...
The default address is: 192.168.88.17
net> load HTTPDownload.efi
load Image fs0:\net\HTTPDownload.efi loaded at 309E9000 - Success
net> load HTMLRenderer.efi
```
ACPI Testing Example Application
Complex Testing in a shell application

- Test hardware features without complex OS
  - Hardware feature development
  - Simpler debug environment than OS
  - More control for probing error conditions
  - Enable efficient testing of features
- Rapid test cycles booting just to UEFI Shell
- Easy to port Linux or MS-DOS style apps

- ACPICA
  Open Source
  www.acpica.org
ACPICA running on 4 socket platform
Summary

- UDK2010 Shell 2.0 implementation
  - Available now
  - Fully compliant to UEFI Shell Specification

- You control Shell size and features

- Network profile can connection to networks

- UEFI Shell is a great test environment
Kevin Davis
InsydeH2O Client Engineering

The most shipped UEFI BIOS