# LinuxCon Europe

## UEFI Mini-Summit

## 7 October 2015

Session 4 – Goodbye PXE, Hello HTTP Boot

**Dong Wei, HP**

# Agenda

- Challenges of Firmware in the Data Center
- PXE and HTTP Boot
- UEFI Shell Scripting
- Data Center Manageability: Redfish and REST APIs
- Putting it all together: HP* ProLiant* Servers
- Summary and Q&A

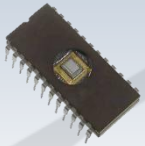
# Agenda

- Challenges of Firmware in the Data Center
- PXE and HTTP Boot
- UEFI Shell Scripting
- Data Center Manageability: Redfish and REST APIs
- Putting it all together: HP* ProLiant* Servers
- Summary and Q&A

# Firmware Challenges In The Data Center

Bare Metal Provisioning

Deployment



Firmware Updates

Firmware Configuration


Automation

Security

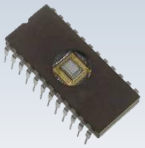
Scalability

Eco-system

# The UEFI Solution

**Bare Metal Provisioning**

- Pre-Boot networking
- IPv4, IPv6 TCP/UDP
- PXE, iSCSI, HTTP, FTP



**Firmware Updates**

- Firmware Management Protocol
- Capsule updates

**Deployment**

- Boot device selection
- Boot order control
- OS install & recovery

**Firmware Configuration**

- Human Interface Infrastructure (HII)
- Platform-To-Driver Configuration (CLP)
- REST Protocol


**Automation**

- UEFI Shell
- Scripting language

**Scalability**

- New hardware abstraction with UEFI protocols
- UEFI driver model
- UEFI device path

# The UEFI Solution

**Security**

- Secure Boot and driver signing
- Security technologies (OpenSSL®, RNG, etc…)
- Encrypted disks and key management
- Interoperability with TCG standards

**Eco-system**

- Standards (UEFI Forum)
- Compliance: Self Certification Test (SCT), Linux* UEFI Validation (LUV)
- Open source code (EDK2 - http://tianocore.org)
- Ubiquitous vendor support (OEMs, ISVs, IHVs, OSVs)

***UEFI offers solutions to today's data center firmware challenges***

# Data Center Manageability Interface Requirements

- **Use security best practices**
- **Support modern architectures**
  - Describe modern architectures (multi-node servers)
  - UEFI-aware (boot order selection, Secure Boot)
- **Scaling**
  - Scale-out servers usage model drastically different from traditional/enterprise servers
  - Management complexities grow exponentially
- **Interoperability for “OEM extensions”**

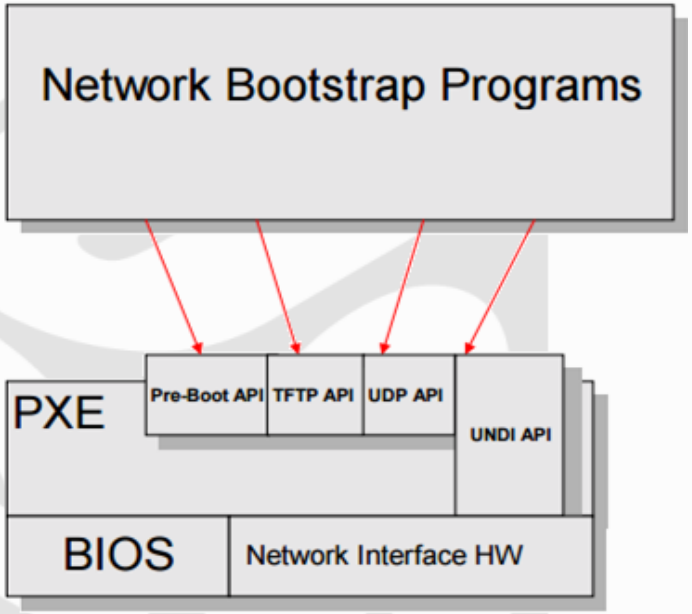
***Today’s Data Center Manageability Interfaces do not meet all of these needs***

# Agenda

- Challenges of Firmware in the Data Center
- PXE and HTTP Boot
- UEFI Shell Scripting
- Data Center Manageability: Redfish and REST APIs
- Putting it all together: HP* ProLiant* Servers
- Summary and Q&A

# PXE Boot Challenges

- **P**reboot e**X**ecution **E**nvironment
- Security Issues
  - Only physical. No encryption or authentication.
  - Rouge DHCP servers, man-in-the-middle attacks
- Scaling issues
  - Circa 1998
  - TFTP timeouts / UDP packet loss
  - Download time = deployment time = $$$
  - Aggravated in density-optimized data centers
- OEMs and users workarounds
  - Chain-load 3rd party boot loaders (iPXE, mini-OS)



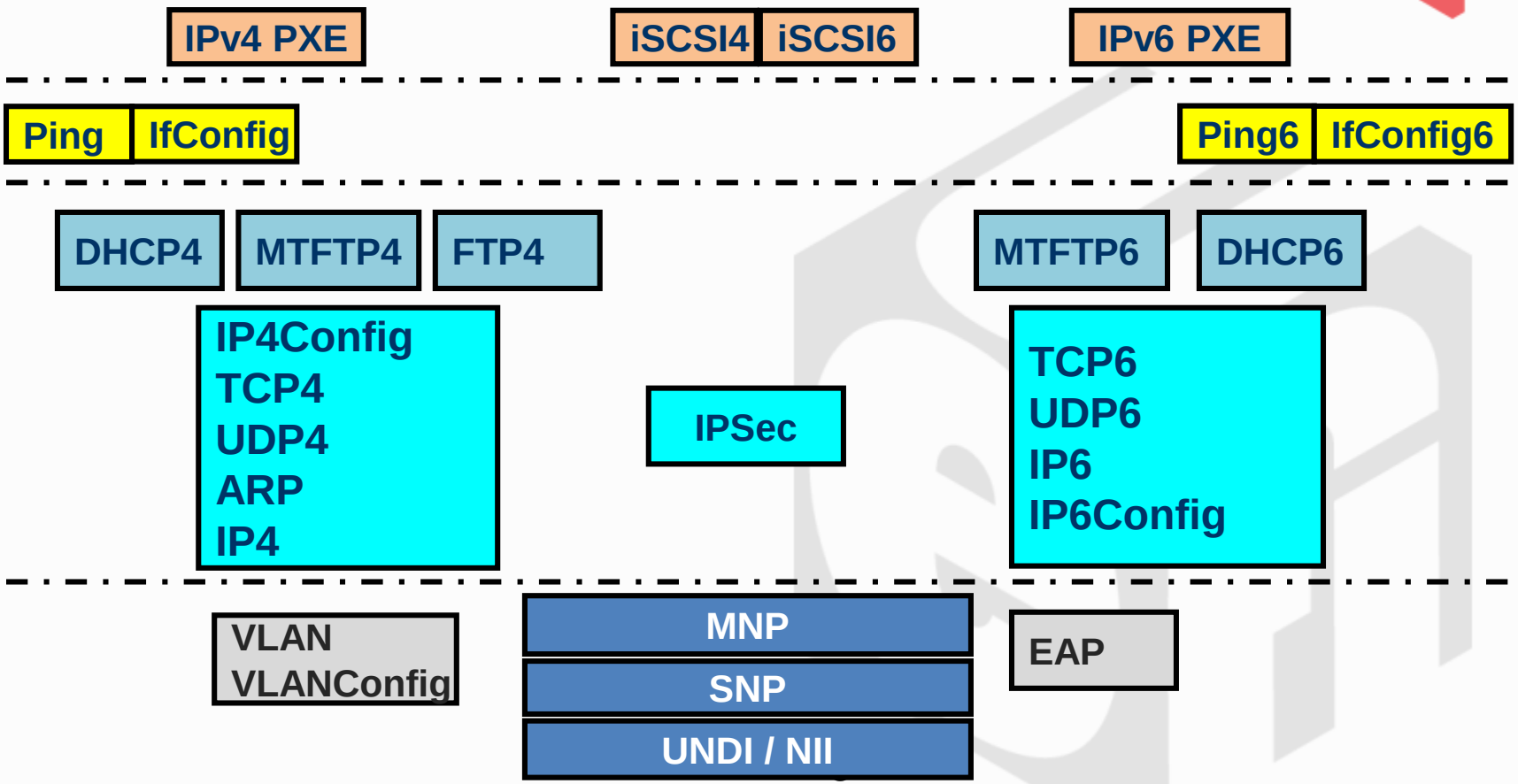

***PXE is not keeping up with modern data center requirements***

# iPXE (http://ipxe.org) Before UEFI 2.5

- Open-source PXE client and bootloader
  - Required chain loading (PXE boot to iPXE then run iPXE to HTTP download)
- Adds support of HTTP Boot:
  - Used to only work with traditional BIOS, users have to choose between **HTTP Boot** and **UEFI Secure Boot**
  - Used to only provides low-level SNP interface (no HTTP Boot) in UEFI
  - Recently “the iPXE UEFI vision has mostly been implemented”
  - Not part of the UEFI standard
- iPXE UEFI vision
  - *“Provide the same advanced features within the UEFI environment as are currently provided within the Traditional BIOS environment” - http://ipxe.org/efi/vision*

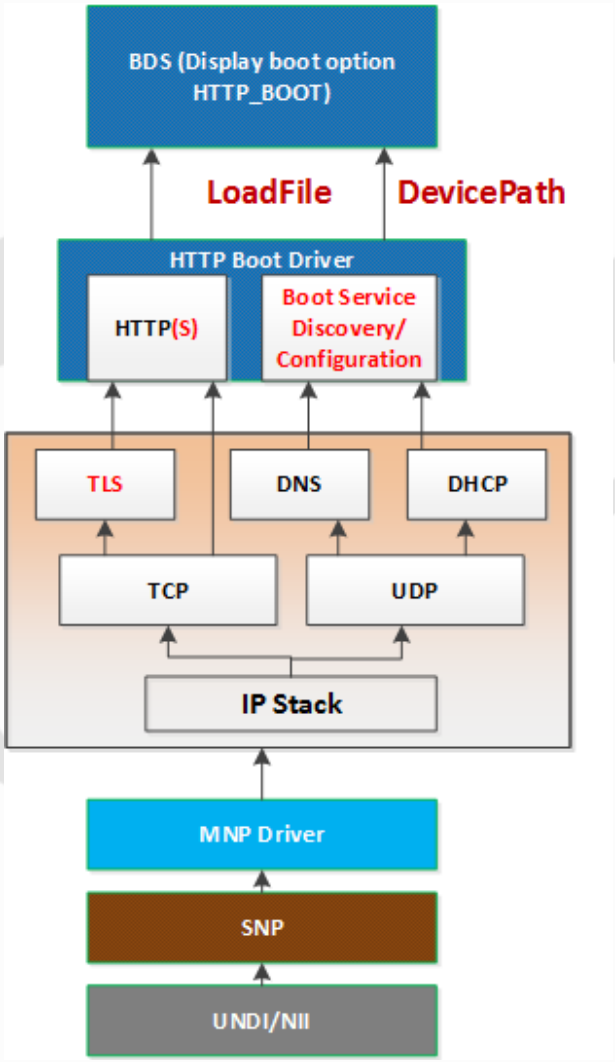
***Why not solve the PXE boot challenges natively in a standard way in UEFI?***

# Network Stack In UEFI v2.4



# Network Stack In UEFI v2.5

- Builds on top of UEFI 2.4
- DNS (IPv4 / IPv6)
- HTTP (IPv4 / IPv6)
- TLS (for HTTPs)
- HTTP Boot Wire Protocol
- Bluetooth® technology
- Wi-Fi*



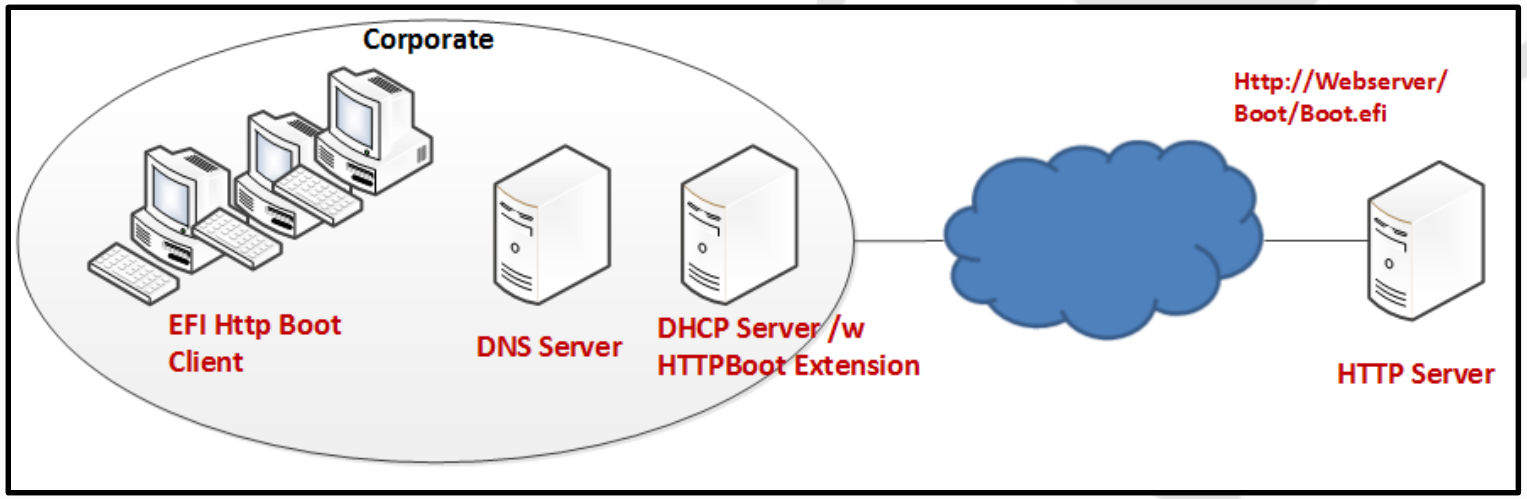
# UEFI Native HTTP Boot

## HTTP Boot Wire Protocol

- Boot from a URL
- Target can be:
  1. EFI Network Boot Program (NBP)
  2. Shrink-wrapped ISO image
- URL pre-configured or auto-discovered (DHCP)

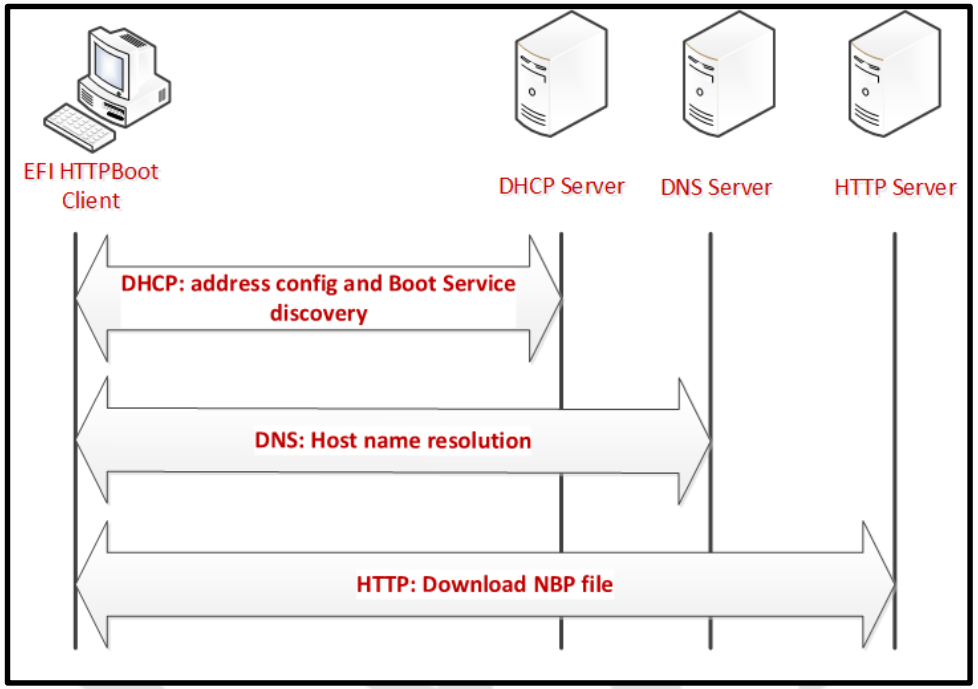
## Addresses PXE issues

- HTTPs addresses security
- TCP reliability
- HTTP load balancing



# HTTP Boot DHCP Discovery

- **HTTP Boot DHCP Discovery**
  - New HTTP Boot “Architectural Types” to distinguish from PXE
  - Client sends DHCP Discover request
  - DHCP Server responds with offer that includes the boot file URL
  - Clients resolves URL server name from DNS
  - Client downloads boot image from HTTP server using HTTP(s)



# RAM Disk Standard

- UEFI 2.5 defined RAM Disk device path nodes
  - Standard access to a RAM Disk in UEFI
  - Supports Virtual Disk and Virtual CD (ISO image) in persistent or volatile memory
- ACPI 6.0 NVDIMM Firmware Interface Table (NFIT)
  - Describe the RAM Disks to the OS
  - Runtime access of the ISO boot image in memory

***HTTP Boot is the emerging solution for modern data centers.***

# iPXE After UEFI 2.5

- Open-source HTTP client and bootloader
  - Still requires chain loading (HTTP boot to iPXE then run iPXE to HTTP download)
- Application note on using UEFI HTTP Boot to chain load into iPXE (courtesy of Michael Brown)
  - http://ipxe.org/appnote/uefihttp

***Options to address the PXE challenges:***
***Native UEFI HTTP Boot , iPXE using UEFI HTTP***

# Agenda

- Challenges of Firmware in the Data Center
- PXE and HTTP Boot
- UEFI Shell Scripting
- Data Center Manageability: Redfish and REST APIs
- Putting it all together: HP* ProLiant* Servers
- Summary and Q&A

# UEFI Shell


- UEFI Pre-boot command line interface (CLI)
  - Much like DOS* or Linux*/Unix* Shell environment
- Interactive prompt and scriptable
- Built-in commands
  - **Standard Commands:** File manipulations, driver management, device access, scripting control, system information, basic network operations
  - **Extensible:** OEMs can provide value-add commands
- Can be embedded as a boot option or bootable from storage
- Fully documented
  - Latest UEFI Shell Specification v2.1

# UEFI Shell Standard Commands


**Scripting**

- echo, stall, set, shift, pause, parse, if / else / endif, for/endfor, reset, exit, cls
- **startup.nsh** auto-start script
- Parsable comma-separated output (-sfo)

**File Operations**

- dir cd, md, rd, mv, copy, del, type, edit, touch, attrib, setsize, comp, compress
- Read/Write files (FAT/FAT32)
- Console/file redirection and piping

**Debug and Test**

- **UEFI Drivers Debug:** load, unload, connect, disconnect, drivers, devices, devtree, dh, openinfo
- **System debug:** memmap, dmem, smbiosview, pci, dblk

# Agenda

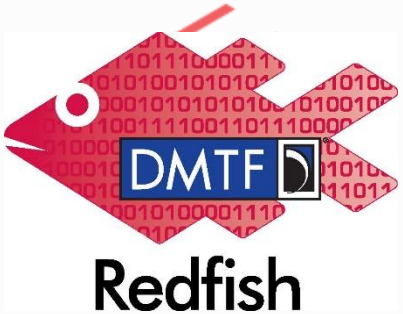
- Challenges of Firmware in the Data Center
- PXE and HTTP Boot
- UEFI Shell Scripting
- Data Center Manageability: Redfish and REST APIs
- Putting it all together: HP* ProLiant* Servers
- Summary and Q&A

# Data Center Manageability Interface Requirements

- Use security best practices
- Support modern architectures
- Scaling
- Interoperability for “OEM extensions”

***Today’s Data Center Manageability Interfaces do not meet all of these needs***

# Redfish

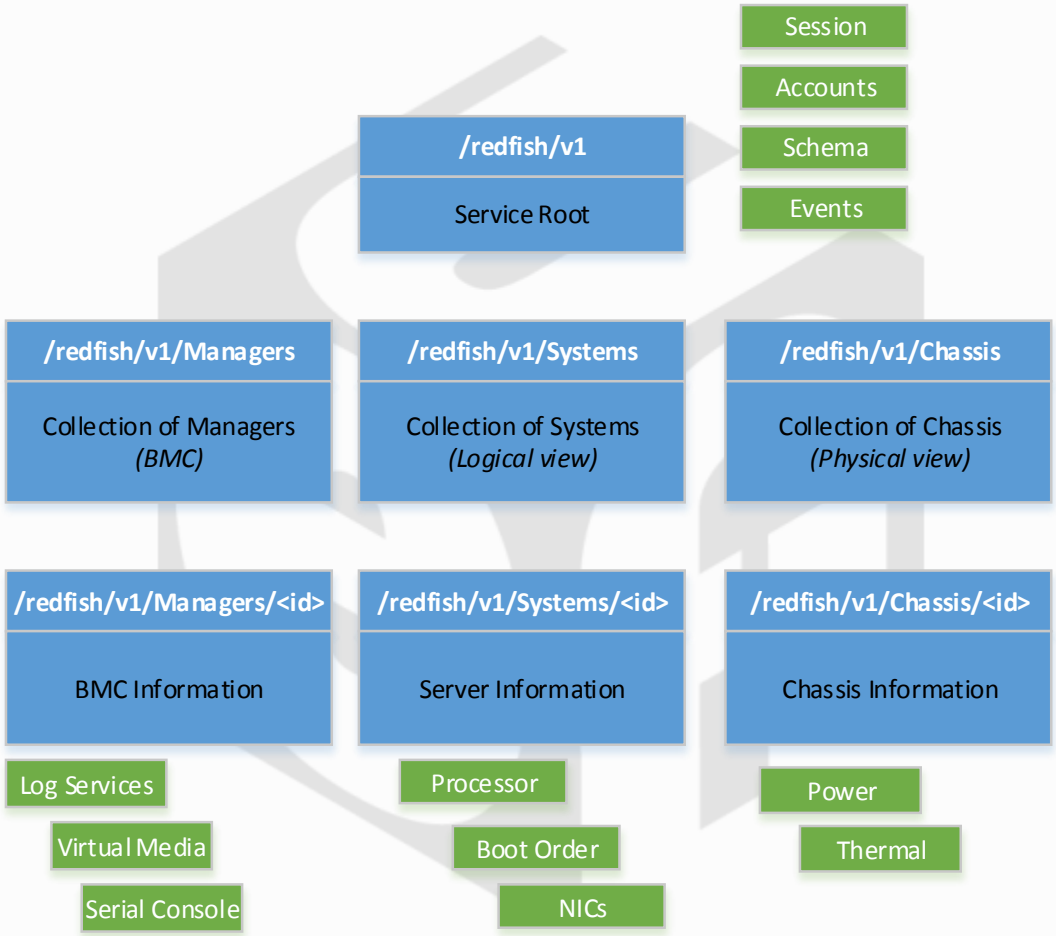



## Architectural successor to previous manageability interfaces (e.g., IPMI)

- **Industry Standard**
  - DMTF* Scalable Platforms Management Forum (SPMF)
    - www.dmtf.org/standards/redfish
    - Specification, schema, mockup, whitepaper, FAQ, resource browser
- **RESTful interface over HTTPs**
  - JSON format
  - Secure (HTTPs)
  - Multi-node and aggregated rack-level servers capable
  - Schema-backed, human readable output

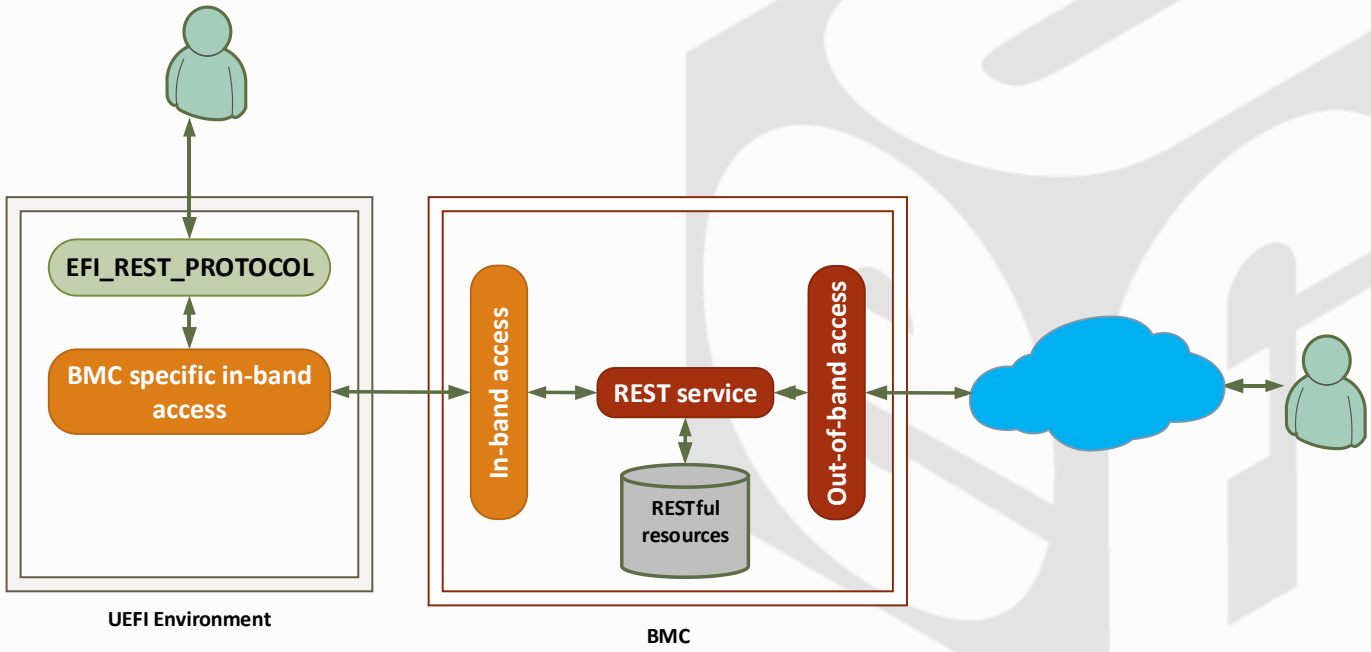
# Redfish Data Model

- Root of service "/redfish/v1"
- Each resource has a type
  - Versioned schema
  - Meta-data
  - OEM extensions
- Collections to describe versatile server hardware architectures
  - Stand-alone
  - Multi-node
  - Rack-level aggregated



# UEFI REST Protocol

- New in UEFI v2.5
- Standard pre-boot in-band access to a RESTful API, like Redfish
- Abstracts BMC-specific access methods (proprietary)



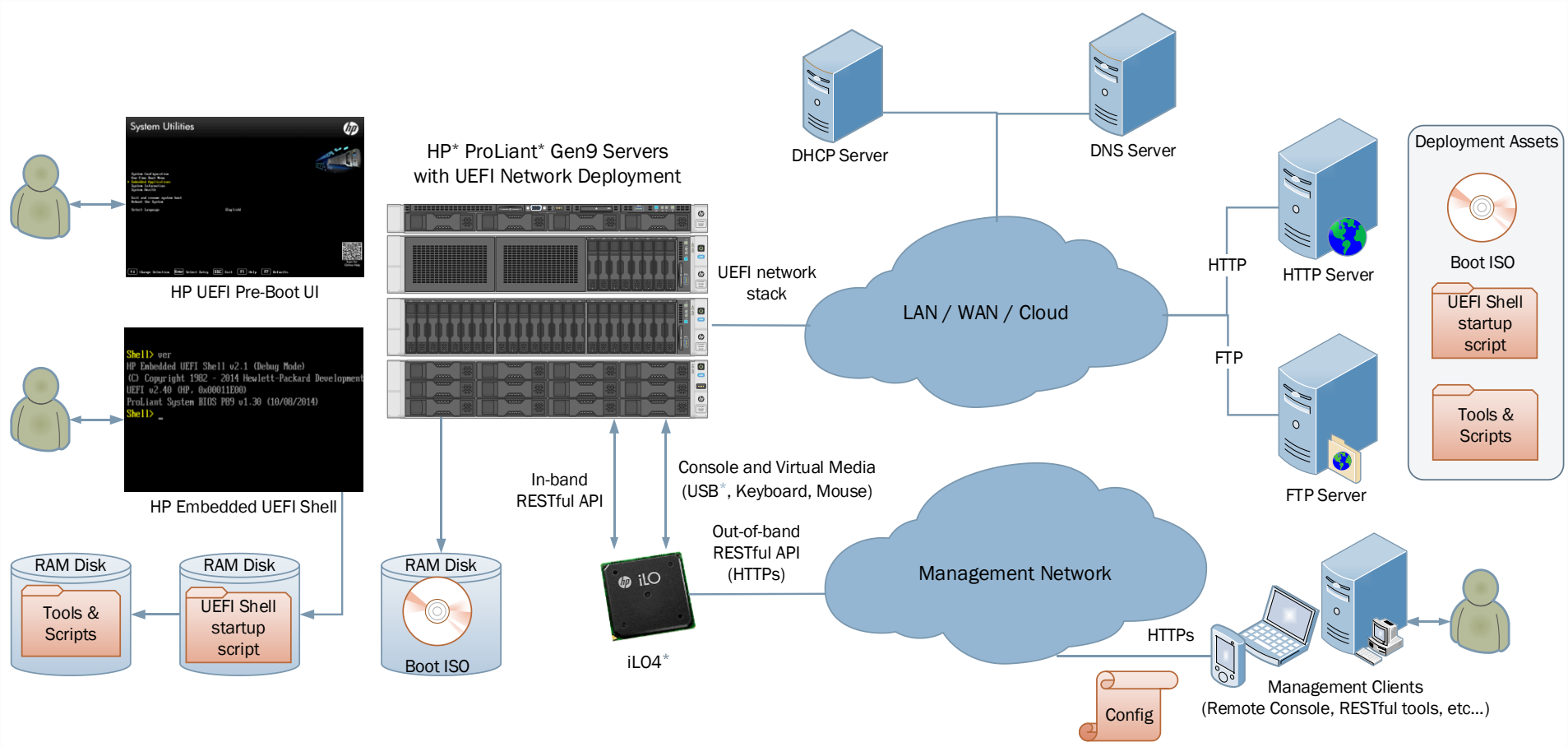
# Agenda

- Challenges of Firmware in the Data Center
- PXE and HTTP Boot
- UEFI Shell Scripting
- Data Center Manageability: Redfish and REST APIs
- Putting it all together: HP* ProLiant* Servers
- Summary and Q&A

# UEFI Deployment Solution On HP* ProLiant* Servers

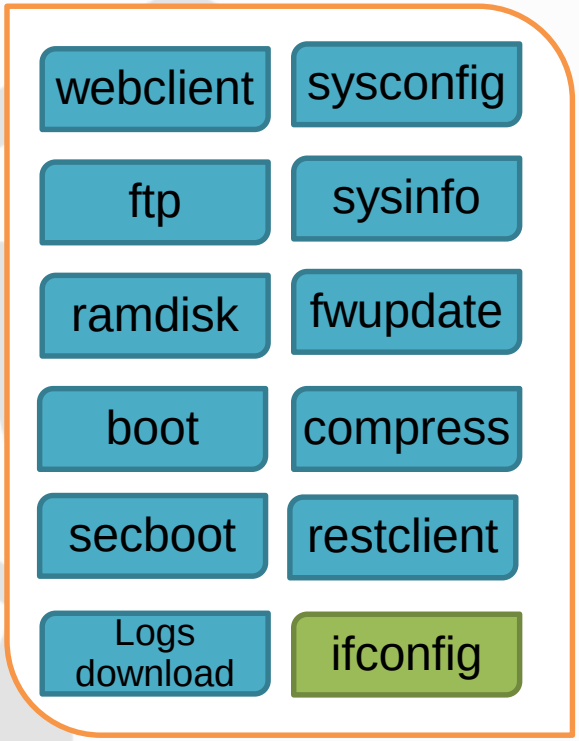
- **UEFI Network Stack Extensions**
  - HTTP, FTP, DNS
  - “Boot from URL” to EFI file or ISO image
  - UEFI iSCSI Software Initiator
- **HP RESTful API**
  - Accessible in-band (from OS) or out-of-band (iLO4* HTTPs)
  - HP* OEM extensions including support for UEFI BIOS configuration
- **Embedded UEFI Shell**
  - Built into the system firmware
  - HP value-add commands for bare-metal deployment
  - Startup script loading from media or network location

# UEFI Deployment Solution On HP* ProLiant* Servers



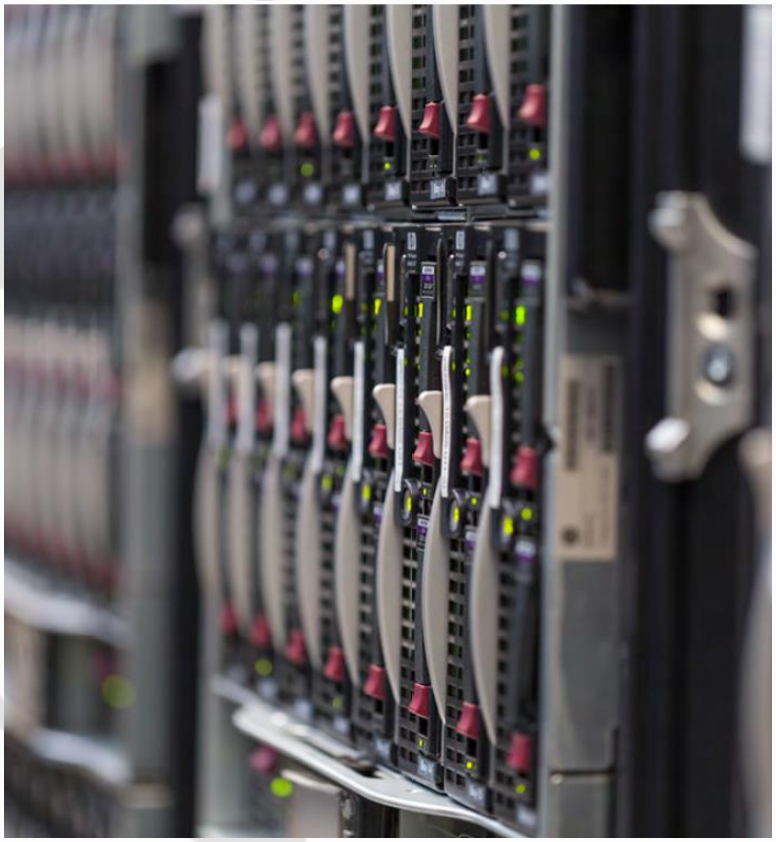
# Embedded UEFI Shell HP* Commands


- **HP* value-add commands for bare-metal deployment**
- **ramdisk** : Provision memory disks and mount ISO files
- **webclient** and **ftp :** Scriptable network download/upload
- **restclient**: In-band client for the HP RESTful API
- **sysconfig** : Configuration CLI (integrates with HP* RESTful API)
- **secboot** : Secure Boot management (physical presence)
- **boot** : Transition to OS/boot targets without rebooting
- **sysinfo** : System hardware/firmware inventory
- **fwupdate** : Firmware updates
- **compress** : ZIP/UNZIP archives
- **ifconfig** : UEFI network stack configuration
- Commands to collect server service/troubleshooting logs



# HP* RESTful API

- HP* RESTful API in iLO4*
  - Modern management API for HP ProLiant* and Moonshot servers
  - Comprehensive inventory and server configuration
  - Conformance with Redfish 1.0
- Integrated with UEFI
  - UEFI BIOS settings configuration
  - UEFI Boot Order and Secure Boot configuration
  - UEFI iSCSI Software Initiator configuration



# HP* RESTful API Example: UEFI BIOS Settings

**GET @ /rest/v1/systems/1/bios**

- Get a list of all UEFI BIOS settings (name/values)


```
"AdminName": "",
"AdminOtherInfo": "",
"AdminPassword": null,
"AdminPhone": "5555555",
"AdvancedMemProtection": "AdvancedEcc",
"AsrStatus": "Enabled",
"AsrTimeoutMinutes": "10",
"AssetTagProtection": "Unlocked",
"AttributeRegistry": "HpBiosAttributeRegistryP89.1.0.40",
"AutoPowerOn": "RestoreLastState",
"BootMode": "Uefi",
```

# HP* RESTful API Example: Secure Boot

**GET @ /rest/v1/systems/1/secureboot**

- Enable/Disable Secure Boot
- Reset all Secure Boot variables to defaults
- Clear all keys (Setup Mode)


```
{
    "Name": "SecureBoot",
    "ResetAllKeys": false,
    "ResetToDefaultKeys": false,
    "SecureBootCurrentState": false,
    "SecureBootEnable": false,
    "Type": "HpSecureBoot.0.9.5"
}
```

# Sample Configuration Script Using HPREST Tool

```
# Login to iLO
hprest login https://clientilo.domain.com -u username -p password

# Configure UEFI network settings (Use Auto and DHCP defaults)
hprest set PreBootNetwork=Auto --selector HpBios.
hprest set Dhcpv4=Enabled

# Configure UEFI Shell startup script from URL
hprest set UefiShellStartup=Enabled
hprest set UefiShellStartupLocation=NetworkLocation
hprest set UefiShellStartupUrl=http://192.168.1.1/deploy/startup.nsh

# Set one-time-boot to Embedded UEFI Shell
hprest set Boot/BootSourceOverrideEnabled=Once --selector ComputerSystem.
hprest set Boot/BootSourceOverrideTarget=UefiShell

# Save and reboot server
hprest commit --reboot=ON
```

# Sample UEFI Shell Deployment Script (startup)

```
# Create FAT32 RAM Disk
ramdisk -c -s 512 -v MYRAMDISK -t F32
FS0:


# Download provisioning OS files from HTTP to RAM Disk
webclient -g http://repo.hp.com/deploy/efilinux.efi
webclient -g http://repo.hp.com/deploy/deploy.kernel
webclient -g http://repo.hp.com/deploy/deploy.ramdisk


# Start provisioning OS
efilinux.efi -f deploy.kernel initrd=deploy.ramdisk
```

# Agenda

- Challenges of Firmware in the Data Center
- PXE and HTTP Boot
- UEFI Shell Scripting
- Data Center Manageability: Redfish and REST APIs
- Putting it all together: HP* ProLiant* Servers
- Summary and Q&A

# Summary

- UEFI 2.5 HTTP Boot bridges the gaps of network boot in the data center
- Redfish is emerging RESTful management API to address modern data center requirements
- HP* ProLiant* Servers showcase of a bare-metal UEFI deployment solution using HTTP Boot, Embedded UEFI Shell, and RESTful APIs

# Next Steps/Call to Action

- Adopt UEFI 2.5 implementations with HTTP Boot
- Adopt Redfish implementations in servers and management software
- Transition data centers to use HTTP Boot and Redfish REST APIs

# Interested In Joining?

**www.uefi.org/membership**

**UEFI FW/OS Forum:**
**uefi.org/FWOSForum**
A free public forum focused on firmware and O/S integration

**USRT Security Issue Reporting:**
**uefi.org/security**
A safe reporting site to inform the UEFI of any security issue or vulnerability based on firmware