Agenda

• Introduction
• UEFI Building Blocks
• Creating a Solution
• Demo
• Call to Action
Introduction
Corporations need methods to configure and deploy many systems on their network. Currently, configuring the firmware, updates and OS Deployment can be:

- Annoying
- Time consuming
- Can cause costly downtime
- May require operating system based tools
- Difficulty rises with the number of systems
- Each system requires individual attention

Current industry solutions are proprietary and do not offer a unified solution for multiple hardware vendors.
Industry Needs

• But the ecosystem is missing:
  – Bare metal
  – Remote configuration
  – Remote firmware update
  – Security
  – Scalability
  – Migration
  – Cloning
  – Scripting
  – Solution for system with and without BMC
Industry Standard Initiative

• Combining several industry standards together offers a highly manageable solution:
  – **REST** Software Architectural “style” from [http://www.w3.org](http://www.w3.org)
  – **JSON** is a data-interchange format from [http://www.json.org/](http://www.json.org/)
  – **oData (open data protocol)** that defines the best practice for building and consuming RESTful APIs from [OASIS](https://www.oasis-open.org/).
  – **Redfish** from [https://www.dmtf.org/](https://www.dmtf.org/)
What is REST?

- **RE**presentational **S**tate **T**ransfer
- Scalable Software Architectural “style”
- Standardized operations (RESTful Interface)
  - HTTP GET, POST, PUT, and DELETE
  - Practical implementations add HTTP PATCH, HEAD
- Standardized operands (nouns)
  - Resources uniquely identified by URIs
- Stateless, atomic operations
  - No client/application context stored server
What is JSON?

- **JavaScript Object Notation**
- Lightweight data-interchange format
  - Easy for humans to read and edit
  - Easy for machines to parse and generate
- Much smaller grammar than XML
  - XML good for “documents”
  - JSON better for “data structures” used in programming languages
What is oData?

• Open Data Protocol (oData) is an open protocol which allows the creation and consumption of query-able and interoperable RESTFul APIs in a standard way

• Microsoft initiated oData in 2007
  – Version 1/2/3 defined by MS
  – Version 4+ moved to OASIS

• Multiple open source projects available to support oData based schemas
What is Redfish?

• Architectural successor to previous manageability interfaces
• Industry Standard
  – DMTF* Scalable Platforms Management Forum (SPMF)
  – www.dmtf.org/standards/redfish
• Technology for Management Controller (BMC) to expose RESTful interface for management clients
• Redfish allows RESTful interfaces:
  – JSON format
  – Secure (HTTPs)
  – Multi-node and aggregated rack-level servers capable
  – Schema-backed, human readable output
• UEFI firmware communicates with BMC using Redfish standard for typical BIOS-BMC data exchange
  – For systems without BMC, network communication with an oData server can be done instead
UEFI Building Blocks
UEFI Highlights

• Network Stack
  • UNDI / SNP / MNP / IPV4 / IPV6 / TCP / UDP / ARP / DHCP
  • DNS (IPv4 / IPv6)
  • HTTP (IPv4 / IPv6)
  • TLS (for HTTPs)
  • HTTP Boot Wire Protocol

• EFI REST Protocol support

• UEFI Configuration Language

Keywords
UEFI Native HTTP(S)

- HTTP Support
  - Native support for HTTP
- HTTPS Support
  - Native support for secure data transfer through HTTPS
- Both can be used to transfer data and go to specific URIs
BIOS REST Support

- New in UEFI v2.5, `EFI_REST_PROTOCOL` interface to communicate with REST servers
- Uses `EFI_HTTPUTILITY_PROTOCOL` to build/pars HTTP headers
- Standard pre-boot access to a RESTful API and Redfish like interface:

  ```c
  typedef struct _EFI_REST_PROTOCOL {
    EFI_REST_SEND_RECEIVE SendReceive; // Provides an HTTP-like interface to send and receive resources from a REST service.
    EFI_REST_GET_TIME GetServiceTime; // (Optional) Returns the current time of the REST service.
  } EFI_REST_PROTOCOL;
  
  typedef EFI_STATUS (EFIAPICALLTYPE *EFI_REST_SEND_RECEIVE)(
    IN EFI_REST_PROTOCOL *This,
    IN EFI_HTTP_MESSAGE *RequestMessage,
    OUT EFI_HTTP_MESSAGE *ResponseMessage);
  ```

- Abstracts the communication to application/driver that wants to use the REST service.
- The `EFI_REST_PROTOCOL` Instance can be installed for in-band communication with BMC as well as instance for communicating via Network with oData Server.
UEFI Configuration Language

Keywords (Mapping Language)

- UEFI 2.5 recommends representing questions by configuration language
- Unique keywords defined in a separate configuration language for each question’s prompt field
- These question keywords are unique strings and allow identification of the questions
- Generic drivers can implement support by using the “x-UEFI” Language
- Can be used to transfer question details in JSON format for Redfish based configuration
Creating a Solution
OOB Firmware Configuration:

Introduction

- BIOS Firmware Configuration are traditionally done in-band in the BIOS environment or using OS tools
- The configuration is stored in NVRAM as blob of Binary data
  - Configuration data now needs to be properly matched with individual setup questions
- OOB Firmware Configuration allows configuring the BIOS firmware remotely via a Management Controller using Redfish
- With the following considerations
  - Redfish compliant configuration representation
  - Configuration need be managed independent of BIOS versions
  - Configuration can be maintained for different system models
UEFI Configuration Language Keywords (Mapping Language)

1. Retrieve the Platform Exported Data

2. Search token with the keyword in the configuration language (e.g. EnableXYZ in x-UEFI or x-AMI)

3. After finding token #33 search the associated IFR opcodes for prompt which refers token #33.

4. Once the IFR question is found use that questions value to read or write.
OOB Firmware Configuration:
Remote firmware UI

- Remote Firmware Setup UI application hosted by BMC or oData server
- HTML5 pages are pushed from BIOS to BMC or oData server
- BMC or oData server parse setup question details to show controls and get/set the values
- HTML5 provides an easy customization capability
OOB Firmware Configuration:

VFR Data

- Uefi Configuration Language Codes (Mapping Language) and VFR

```plaintext
Unifile and SD file

#string STR_ACPI_AUTO_PROMPT            #language eng "Enable ACPI Auto Configuration"
    #language x-AMI "ACPI004"

checkbox varid = SETUP_DATA.AcpiAuto,
    prompt = STRING_TOKEN(STR_ACPI_AUTO_PROMPT),
    help = STRING_TOKEN(STR_ACPI_AUTO_HELP),
    flags = 0,
    default = DEFAULT_AUTO_ACPI,
    defaultstore = AmiMfgDefault,
endcheckbox;

#string STR_ACPI_S4_PROMPT              #language eng "Enable Hibernation"
    #language x-AMI "ACPI002"

checkbox varid = SETUP_DATA.AcpiHibernate,
    prompt = STRING_TOKEN(STR_ACPI_S4_PROMPT),
    help = STRING_TOKEN(STR_ACPI_S4_HELP),
    flags = 0,
    default = DEFAULT_SS4,
    defaultstore = AmiMfgDefault,
endcheckbox;
```
BIOS Setup Question Information as JSON

```
{  
  "AttributeName": "ACPI001",
  "DefaultValue": "S3 (Suspend to RAM)",
  "DisplayName": "ACPI Sleep State",
  "HelpText": "Select the highest ACPI sleep state the system will enter when the SUSPEND button is pressed.",
  "ReadOnly": false,
  "Type": "Enumeration",
  "Value": [
    {
      "ValueDisplayName": "Suspend Disabled",
      "ValueName": "Suspend Disabled"
    },
    {
      "ValueDisplayName": "S1 (CPU Stop Clock)",
      "ValueName": "S1 (CPU Stop Clock)"
    },
    {
      "ValueDisplayName": "S3 (Suspend to RAM)",
      "ValueName": "S3 (Suspend to RAM)"
    }
  ]
}
```
OOB Firmware Configuration:
JSON Config Data

• BIOS Configuration

```json
<Server>/redfish/v1/Systems/Self/Bios

........
"AttributeRegistry": "BiosAttributeRegistry0ACAK.0.19.0",
........
"Attributes": {
  "ACPI001": "S1 (CPU Stop Clock)",
  "ACPI002": true,
  "ACPI003": false,
  "ACPI004": false,
........
```
OOB Firmware Configuration: JSON Change Config Data

- Bios Configuration Setting Object from Remote Client

```
<Server>/redfish/v1/Systems/Self/Bios/SD

......
"Attributes": {
  "ACPI002": false,
  ......}
```
Communication Block Diagrams

- **BMC**
  - REST
  - Pass-through
  - BMC In-band Interface

- **Firmware**
  - REST
  - FW Over System Interface
  - REST
  - FW Setup

- **oData Server**
  - REST
  - oData server service

- **Firmware**
  - REST
  - UEFI Network Stack
  - REST
  - FW Setup
OOB Firmware Management with BMC

Server with Aptio

In-Band System Interface

Out of – Band Communication

Management Network
Firmware Configuration with oData

LAN / WAN

UEFI Pre-Boot UI

System 1

System 2

System N

oData Server

Management Clients (Redfish clients, tools, etc.)

Firmware Configuration Data

RESTful API Via Network

[Diagram showing network connections between systems, an OData server, and management clients.]
Extending Solutions for Additional Features

• Using oData server or BMC solutions can be created beyond firmware configuration:
  – OS deployment
  – Firmware update deployment
  – System cloning
  – Diagnostic deployment on demand
  – OS remote backup and restore

• Since all interfaces are abstract, end user organizations can create their own tools and make IT admins happy
Demo
Demo Video

• In video demo, non-BMC firmware management will be shown including:
• Changing firmware settings of a target desktop
• Changing firmware settings of a target laptop
• Pushing changes made locally on target machines to oData server
• Pushing changes made remotely on the oData server to target machines
Call to Action
Call to Action

• OEMs should get involved and read the specifications and add support accordingly
• End customers should get involved and ask their OEMs for Redfish based solutions
• The industry should work together to create an ecosystem where more advanced solutions can be created
• Everyone should get involved in the UEFI and other specifications to continue the evolution
Thanks for attending the UEFI US Fall Plugfest 2016

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org