



# The Role of Redfish in UEFI Forum Firmware Specifications

**Presented by UEFI Forum**

*July 17, 2019*

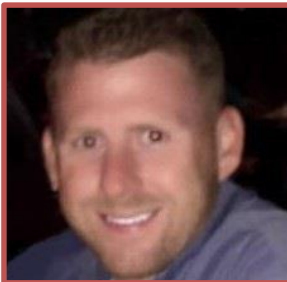
# Welcome & Introductions



Moderator: Brian Richardson  
Firmware Ecosystem Development  
Member Company: Intel Corporation  
@intel\_brian



Panelist: Samer El Haj Mahmoud  
DMTF Representative  
@samerhaj



Panelist: Jason Spottswood  
Member Company: Hewlett Packard  
Enterprise



Panelist: Zach Bobroff  
Member Company: American  
Megatrends

# Samer El Haj Mahmoud



*Lenovo, DMTF*

Principal Engineer at Lenovo Data Center Group. Author and contributor of the DMTF Redfish specifications

- Lead architect for Operating Systems and Solutions technology enablement at Lenovo DCG
- Representing DMTF Redfish Forum
- Active participant and contributor to industry standards bodies, including the DMTF Redfish Forum, and the UEFI Forum
- Leading Lenovo's DMTF Redfish industry standard participation, architecture, and Redfish firmware implementation
- 20 years of experience in server development, in the areas of firmware, operating systems, system software, hardware management, and industry standards

# Jason Spottswood



## *Hewlett Packard Enterprise*

- Senior Manager of UEFI core development for HPE servers, and direct development team responsible of UEFI/Redfish support.
- Member of DTMF Redfish and UEFI Forum
- 18 year career in BIOS development
- Chairman of the NVDIMM sub-team (NVST)

# Zach Bobroff



## *American Megatrends*

- Technical marketing manager of UEFI and remote management products
- Member of DMTF Redfish and UEFI Forum
- 11 years of BIOS development at AMI

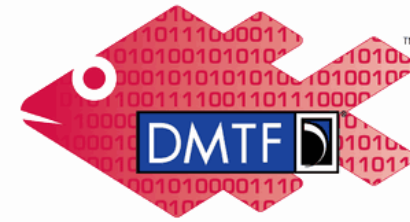


# Samer El Haj Mahmoud

DMTF

*What is Redfish? How can I use it to help manage my servers?*

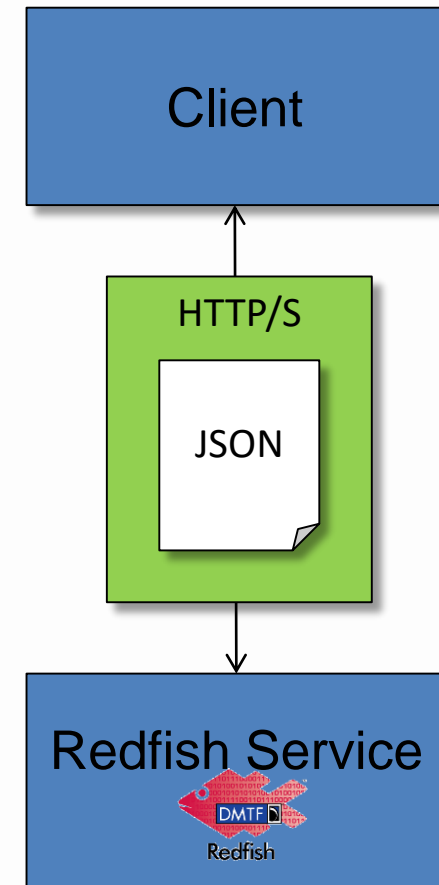
# What is Redfish™ ?



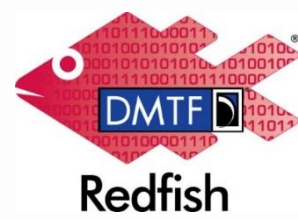
Redfish



- A **DMTF** industry standard (<http://redfish.dmtf.org>)
  - DMTF developing manageability standards for 27 years
  - Allied with 21 standards and 80+ research organizations
- **Redfish**: RESTful interface for managing IT Infrastructure
  - Built on modern tool-chain (HTTP(s)/TLS/JSON)
  - Content is human-readable JSON, backed by schema (CSDL, json-schema, OpenAPI)
- Redfish Models
  - Covering Compute, storage, networking, and power/cooling equipment, platforms, and services



# DMTF Redfish Forum



## Redfish Forum Leadership Companies



## Redfish Forum Supporting Companies

American Megatrends (AMI), ARM, Artesyn Embedded Technologies, Cray, Eaton, Fujitsu, Huawei, IBM, Insyde, Mellanox, Microchip, NetApp, Newisys, OSISOFT, Quanta, Solarflare, Toshiba, Western Digital

## Redfish Industry Alliance Partners & efforts





# Redfish Approach to HW Management



## Design Tenets

- Leverage common Internet / Web Services, and other standards
- Represent modern HW designs (standalone to scale-out, OCP)
- Separation of protocol and data model: can be revised independently

## Protocol Suite

- HTTPS / SSL: Primary data transport
- SSDP from uPnP: Service Discovery
- HTTP-based alert subscription (HTTP callbacks, SSE streaming)

## REST & JSON

- Modern, standards-based
- Widely used for web services, software defined and public APIs
- Easy for IT professionals and amateurs to utilize

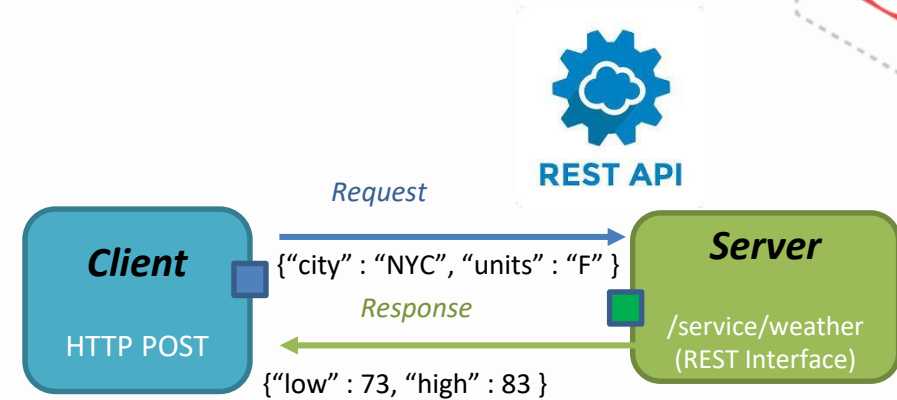
## Data Model

- Schema-based, starting with CSDL, JSON, OpenAPI Schema
- Data model easy for humans to read/edit, and easy for code to parse
- New data modeling tenants to facilitate ease of design

# Why HTTP, REST, and JSON?



- **HTTP(S):** The Web protocol
  - Well-understood by admins
  - Known security model
  - Known network configuration



- **REST - RE**presentational **S**tate **T**ransfer
  - Preferred Software Architectural “style” for web development
  - Standard verbs (HTTP GET / PUT / PATCH / POST / DELETE / HEAD) and nouns (resources, identified by URIs)

- **JSON** – <http://www.json.org>
  - Modern data format. Simpler than XML
  - Easy for humans to read and write
  - Easy for machines to parse and generate

{JSON}

```
{
  "BootMode": "Uefi",
  "EmbeddedSata": "Raid",
  "Nic1Enable": true,
  "ProcCoreCount": 8
}
```

# Redfish API usage example



## Client Python code

```
rawData = urllib.urlopen('https://10.243.1.18/redfish/v1/Systems/1')
jsonData = json.loads(rawData)
print( jsonData['SerialNumber'] )
```

## Output

```
KVX0151
```

Three lines of code:

- Point to the resource
- Get the data
- Print the serial number.

# Sample Redfish Data Model



## Service Root

**/redfish/v1**  
Root  
Links to all content

## Collection

**/redfish/v1/Systems**  
Collection of Systems  
"Logical view"

**/redfish/v1/Chassis**  
Collection of Chassis  
"Physical view"

**/redfish/v1/Managers**  
Collection of Managers  
"OOB manageability"

**/redfish/v1/Fabrics**  
Collection of Fabric Inter-connect

## Singleton

**/redfish/v1/Systems/<id>**  
Server System  
"Logical computer system"

**ComputerSystems**  
**/redfish/v1/Chassis/<id>**  
Chassis  
"Physical asset info"

**ManagedBy**  
**/redfish/v1/Managers/<id>**  
Baseboard Mgmt Ctr (BMC)

**/redfish/v1/Fabrics/PCIe**  
PCIe Fabric

- Processors
- Memory
- Storage
- NICs
- LogService
- BIOS
- PCleDevices
- Power
- Thermal
- PCle Slots
- Virtual Media
- Net protocol
- Serial Interface
- Endpoints
- Switches
- Zones

- Sessions
- Accounts
- Events
- Tasks
- Jobs
- Updates
- Registries
- Schemas

# Sample Redfish Output - Service Root



HTTP GET @ https://<ip>/redfish/v1/

```
{
  "@odata.id": "/redfish/v1/",
  "@odata.type": "#ServiceRoot.1.0.0.ServiceRoot",
  "@odata.context": "/redfish/v1/$metadata#ServiceRoot",
  "RedfishVersion": "1.0.0",
  "UUID": "00000000-0000-0000-0005-000000000001",
  "Chassis": {
    "@odata.id": "/redfish/v1/Chassis/",
  },
  "Managers": {
    "@odata.id": "/redfish/v1/Managers/",
  },
  "Systems": {
    "@odata.id": "/redfish/v1/Systems/"
  },
  "SessionService": {
    "@odata.id": "/redfish/v1/SessionService/",
  },
  "Registries": {
    "@odata.id": "/redfish/v1/Registries/"
  },
  "JsonSchemas": {
    "@odata.id": "/redfish/v1/JsonSchemas/"
  }
}
```

## Starting point

- Links to all resources
- Systems, Chassis, Managers, Fabric Collections

## Services

- Events
- Accounts
- Tasks
- Jobs
- Sessions
- FW Updates

## Metadata

- Links to Schema (JSON, CSDL XML)
- Links to Registries (BIOS Attributes, Messages)

# Sample Redfish output - Computer System



HTTP GET @ https://<ip>/redfish/v1/Systems/1

```
{
  "@odata.id": "/redfish/v1/Systems/1",
  "@odata.type": "#ComputerSystem.1.5.0.ComputerSystem",
  "Manufacturer": "Contoso",
  "Model": "3500",
  "SerialNumber": "437XR1138R2",
  "UUID": "38947555-7742-3448-3784-823347823834",
  "ProcessorSummary": {
    "Count": 2,
    "Model": "Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz"
  },
  "Actions": {
    "#ComputerSystem.Reset": {
      "target": "/redfish/v1/Systems/1/Actions/ComputerSystem.Reset"
    }
  },
  "Bios": {
    "@odata.id": "/redfish/v1/Systems/437XR1138R2/BIOS"
  },
  "Processors": {
    "@odata.id": "/redfish/v1/Systems/1/Processors"
  },
  "Memory": {
    "@odata.id": "/redfish/v1/Systems/437XR1138R2/Memory"
  }
}
```

## Boot flow

- Power Control
- Boot Order and override

## System Info

- BIOS Version
- UUID
- Serial Number
- Asset Tag
- Manufacturer
- Model
- SKU

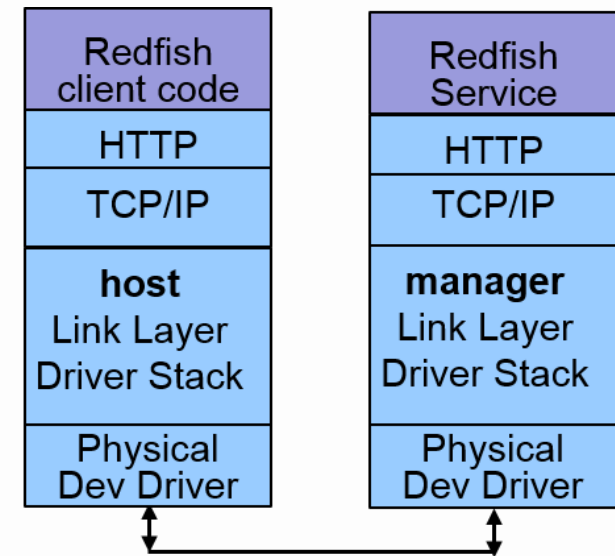
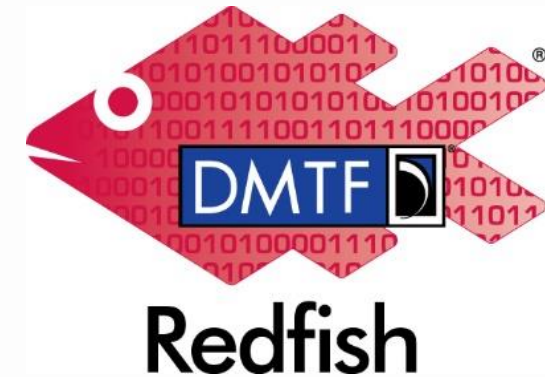
## Links to components

- Memory
- CPU
- Storage
- Networking
- TPM
- BIOS
- SecureBoot

# Redfish Host Interface




- **DMTF Host Interface Specification - [DSP0270](#)**
  - “In-band” access to the Redfish service from UEFI/Host OS
  - Replacement for IPMI-over-KCS
- **TCP/IP Based**
  - Redfish HTTPs requests & responses over a TCP/IP network connection between Host/client and Manager/service.
  - Over any physical or logical interconnect that can route TCP/IP
- **OS Discovery**
  - SMBIOS Type 42 to identify the network interface



# DMTF Redfish Resources



- **Redfish User Forum**
  - User forum for questions, suggestions and discussion
  - <http://www.redfishforum.com>
- **Redfish Developer Portal**
  - Redfish Interactive Resource Explorer
  - Educational material, Hosted Schema files, documentation
  - <http://redfish.dmtf.org>
- **Redfish Standards page**
  - Schemas, Specs, Mockups, White Papers, Educational Material
  - <http://dmtof.org/redfish>
- **DMTF Redfish Forum**
  - Companies involved, Schedules & Future work, Charter
  - Join the DMTF to get involved in future work
  - <http://www.dmtf.org/standards/spmf>

DMTF  DISTRIBUTED MANAGEMENT TASK FORCE, INC.  
**Redfish™ Developer Hub**

Home Mockups About the Redfish API

## Welcome to the Redfish Developer Hub

DMTF's Redfish™ API is an open industry standard specification and schema that helps enable simple and secure management of modern scalable platform hardware. By specifying a RESTful interface and utilizing JSON and OData, Redfish helps customers integrate solutions within their existing tool chains. An aggressive development schedule is quickly advancing Redfish toward its goal of addressing all the components in the data center with a consistent API.

**Welcome Developers**




The DMTF's Redfish Developer Hub is a one-stop, in-depth technical resource – by developers, for developers – **designed to provide all the files, tools, community support, tutorials and other advanced education you may need to help you use Redfish.**

## Redfish Specification Forum

Home Help Search Welcome Guest. Please [Login](#) or [Register](#).

Redfish Specification Forum > Home >

News Welcome to our new forum!

Specification, Protocol, Schema and Payloads				
	Board	Threads	Posts	Last Post
	<b>Protocol and Specification</b> Discussion about the Redfish Specification and the RESTful HTTP protocol. Moderator: <a href="#">Admin</a>	1	2	Retrieving individual properties by j2hilland Sep 12, 2016 at 7:42am
	<b>CSDL and json-schema</b> Discussion about the contents of the standard Redfish schemas, and the published CSDL (XML) or json-schema definition files	1	2	How to use the Location property under Resource ? by mraineri Aug 12, 2016 at 6:33am
	<b>Feature Requests</b> Requests to add features to the Redfish Specification, make additions to existing Schema, or to create a new Schema.	1	2	Creating a webinterface/KVM-over-IP session for user by jautor Aug 12, 2016 at 6:33am



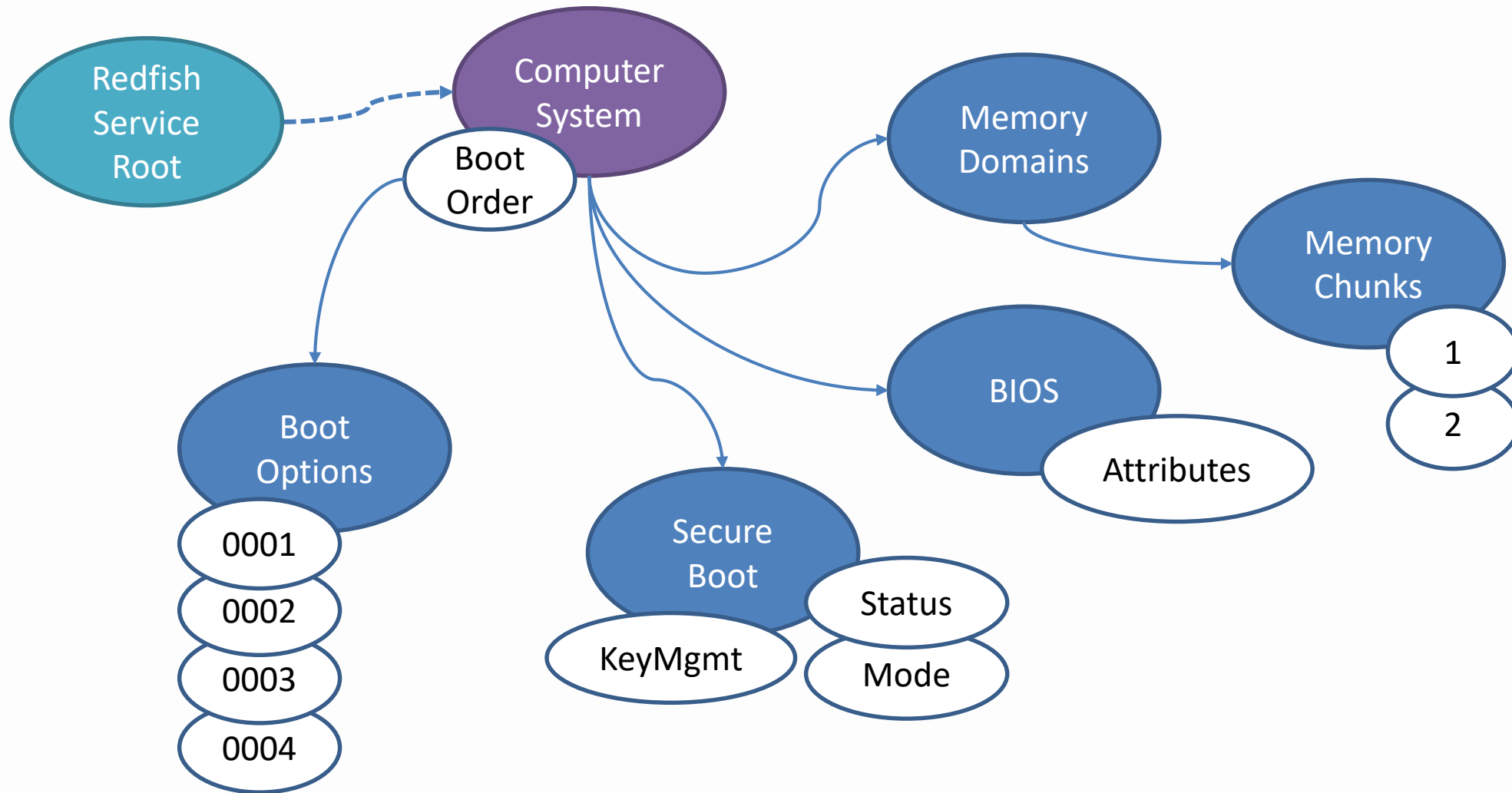


# Jason Spottswood

HPE

*As a user or administrator, I am familiar with configuring a system from the UEFI pre-boot system utility. Does Redfish provide resources for platform configuration, and how are the resources modeled?*

# BIOS Configuration Resources



# Boot Order Override



```
"Boot": {  
  "BootSourceOverrideEnabled": "Once",  
  "BootSourceOverrideMode": "UEFI",  
  "BootSourceOverrideTarget": "Pxe",  
  "BootSourceOverrideTarget@Redfish.AllowableValues": [  
    "None",  
    "Pxe",  
    "Floppy",  
    "Cd",  
    "Usb",  
    "Hdd",  
    "BiosSetup",  
    "Utilities",  
    "Diags",  
    "UefiTarget",  
    "SDCard",  
    "UefiHttp",  
    "UefiBootNext"  
  ],  
  "UefiTargetBootSourceOverride": "",
```

- Boot section taken from ComputerSystem resource
- Allows a single target override to UEFI boot order
- Can specify an alias or specific UEFI target
- UEFI target uses UEFI device path

# UEFI Boot Order



```
"Boot": {  
  "BootOptions": {  
    "@odata.id": "/redfish/v1/Systems/1/BootOptions"  
  },  
  "BootNext": "Boot0003",  
  "BootOrder": [  
    "Boot0001",  
    "Boot0000",  
    "Boot0002",  
    "Boot0004",  
    "Boot0003"  
  ]  
},
```

- On UEFI systems, these properties affect the UEFI variables for boot: *Boot####*, *BootOrder*, *BootNext*
- URI link to the collection of boot options

# Boot Options



```
{
  "@Redfish.Copyright": "Copyright 2017-2019 DMTF. All rights reserved.",
  "@odata.id": "/redfish/v1/Systems/1/BootOptions/1",
  "@odata.type": "#BootOption.v1_0_2.BootOption",
  "Id": "1",
  "Name": "Boot Option",
  "Description": "UEFI Boot Option",
  "BootOptionReference": "Boot0000",
  "DisplayName": "Windows Boot Manager",
  "UefiDevicePath": "PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x0)/Scsi(0x0,0x0)/HD(2,GPT,B0
  "Alias": "Hdd",
  "RelatedItem": [
    {
      "@odata.id": "/redfish/v1/Systems/1/
    }
  ],
  "Oem": {}
}
```

- BootOptionReference: The UEFI Boot Option variable name
- UefiDevicePath: The UEFI device path used to access the UEFI Boot Option
- Alias: class of the boot device

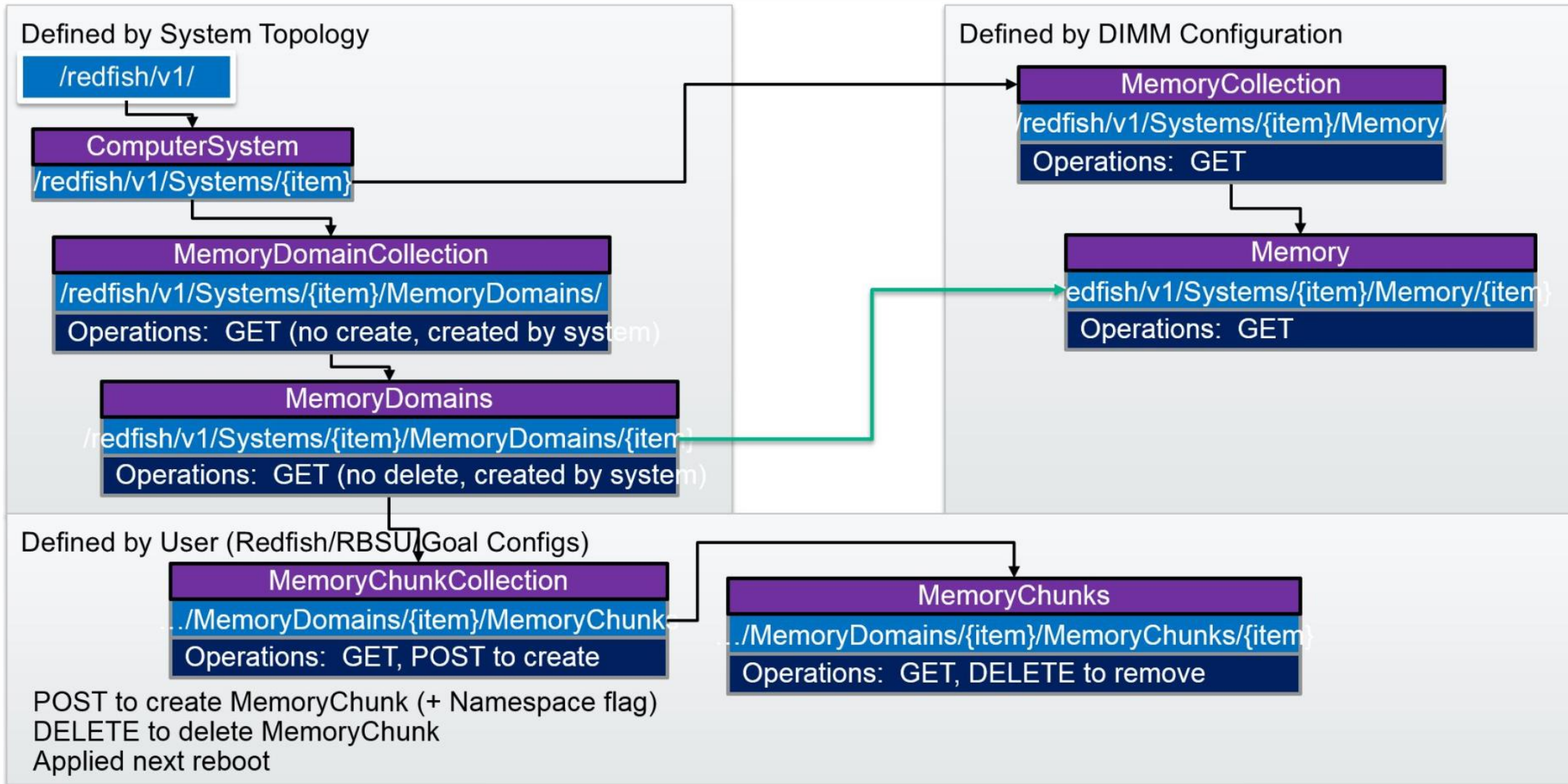
# Secure Boot



```
{
  "@Redfish.Copyright": "Copyright 2014-2019 DMTF. All rights reserved.",
  "@odata.id": "/redfish/v1/Systems/1/SecureBoot",
  "@odata.type": "#SecureBoot.v1_0_5.SecureBoot",
  "Id": "SecureBoot",
  "Name": "UEFI Secure Boot",
  "Actions": {
    "#SecureBoot.ResetKeys": {
      "target": "/redfish/v1/Systems/1/SecureBoot/Actions/SecureBoot.ResetKeys",
      "ResetKeysType@Redfish.AllowableValues": [
        "ResetAllKeysToDefault",
        "DeleteAllKeys",
        "DeletePK"
      ]
    },
    "Oem": {}
  },
  "SecureBootEnable": false,
  "SecureBootCurrentBoot": "Disabled",
  "SecureBootMode": "UserMode",
  "Oem": {}
}
```

- Resetting and deleting Secure boot keys
- SecureBootCurrentBoot: UEFI Secure Boot state
- SecureBootMode: UEFI Secure Boot mode

# Memory Configuration



# BIOS Settings & Attribute Registry



```
{
  "@Redfish.Copyright": "Copyright 2016-2019 DMTF. All rights reserved.",
  "@odata.id": "/redfish/v1/Systems/1/Bios",
  "@odata.type": "#Bios.v1_1_0.Bios",
  "Id": "Bios",
  "Name": "BIOS Configuration Current Settings",
  "Description": "BIOS Configuration Current Settings",
  "AttributeRegistry": "BiosAttributeRegistryP89.v1_0_0",
  "Attributes": {
    "AdminPhone": "",
    "BootMode": "Uefi",
    "EmbeddedSata": "Raid",
    "NicBoot1": "NetworkBoot",
    "NicBoot2": "Disabled",
    "PowerProfile": "MaxPerf",
    "ProcCoreDisable": 0,
    "ProcHyperthreading": "Enabled",
    "ProcTurboMode": "Enabled",
    "UsbControl": "UsbEnabled",
    "ConsoleBaudRate": "115200"
  }
}
```

## Attributes:

- Can be mapped to HII UEFI terms
- Can be used to configure UEFI functionality
- Some OEMs may map these to keywords in the UEFI configuration Namespace Registry

## Configuration Namespace Registry:

- “This is an industry registry of vendor-specific namespace names for purposes of platform configuration”
- <https://uefi.org/confignamespace>



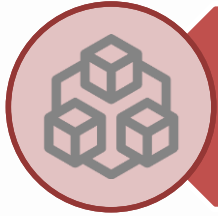
# Zach Bobroff

AMI

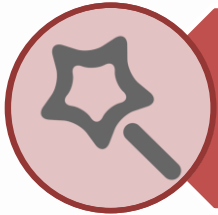
*As a firmware developer of UEFI and Redfish, what has your company learned?*



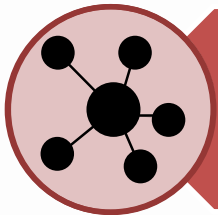
# Redfish: Modern System Management



Redfish is an excellent framework for modern platform management



IPMI has been a very specific implementation that requires OEM specific tools



Redfish is defined in an abstract manner that allows end users to manage non-homogenous HW in a standardized way

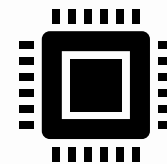


Gives the end user the freedom to manage systems the way they want!

# Redfish Hardware



- The Redfish specification does not define the hardware configuration of the system
  - System can have a BMC, share one BMC for multiple systems or, even not have a BMC!
- The only defined interface is the Redfish Host Interface, which is a network connection
- Common hardware design is to have a BMC on the platform and connect it to the host system via a USB connection that will emulate a LAN device
  - Commonly called LAN over USB



# Redfish Implementation



Redfish provides the API's, the implementation is up to the developer

- This is very similar to the way UEFI defines interfaces

Many of the Redfish interfaces must provide information that cannot normally be gathered by a BMC on its own

- Requires tight integration of information exchange between the UEFI FW and the Redfish FW
- While the host interface is defined by the specification, much of the underlying data transfer between UEFI FW and Redfish FW is proprietary and is important to optimize

# System Configuration: BIOS Setup



Aptio Setup Utility - Copyright (C) 2020 American Megatrends International

Main Advanced Chipset Security Boot Save & Exit

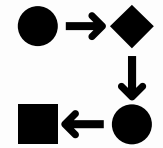
BIOS Information		Choose the system default language
BIOS Vendor	American Megatrends	
Core Version	6.01 j/k	
Compliance	UEFI 2.x; PI 1.x	
Project Version	KGBCIA 1.01 x64	
Build Date and Time	2/30/2020 16:20:01	
Access Level	Administrator	
Board Information		
Board ID	Default string	
Fab ID	Default string	
LAN PHY Revision	N/A	
Processor Information		++: Select Screen
Name	GroomLake-NV	↑↓/Click: Select Item
Type	Intel(R) Xenomorph	Enter/Db1 Click: Select
Speed	LV-426 @ Zeta II Reticuli	+/-: Change Opt.
ID	FTL	F1: General Help
Stepping	616d69	F2: Previous Values
Package	Rx/Sx/Nx	F3: Optimized Defaults
Number of Processors	Not Implemented Yet	F4: Save & Exit
Microcode Revision	A lot / Infinite	ESC/Right Click: Exit
GT Info	Hidden (MJ-12)	
	Hidden (MJ-12)	

Version 2.30.2020 Copyright (C) 2020 American Megatrends International LLC

# System Configuration Data



- The UEFI specification has a very clearly defined method for UEFI Configuration data explained in the HII Chapter
- Redfish has its own method for storing the data called the system attribute registry
- The UEFI and Redfish specifications have differences regarding system configuration that can cause problems



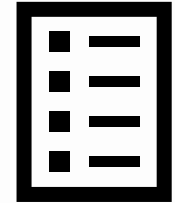
- Complicated layouts where a single question exists multiple times with suppressions makes it difficult to determine the true instance
- UEFI has the luxury that the HII browser can determine everything at UEFI boot-time, Redfish must be available at all times!

Not as easy as just exporting the HII DB!

# System Inventory



- System inventory information is the detailed list of devices in the system
  - Should go to details of connection point and serial number of devices
- A BMC can detect some information, but UEFI generally has all the information due to how it did the base initialization
  - What BMC can detect is very dependent on HW design
- UEFI provides simple abstract interfaces that can be used to easily gather system inventory information



Pcilo, SimpleNetwork, SimpleFilesystem, Smbios and other UEFI protocols can be used to determine the system inventory information that Redfish needs

# Data Exchange



As UEFI collects system configuration and inventory information, it needs to transfer it to the Redfish Host



This is done over LAN and uses the UEFI REST Protocol



Using the REST Protocol, the UEFI FW is able to easily update information in Redfish registries



The amount of data being exchanged is megabytes of information and can slow down system boot time dramatically



# Data Exchange Timing



Should this be exchanged on every boot?


- Maybe only exchange it when the data changes? Maybe use a checksum or hash?
- What about after a Redfish Host or UEFI FW update?

What if the BMC and BIOS boot in parallel? How do you know when the BMC is ready?

- Does the system boot without exchanging the data or wait until Redfish Host Interface is up?
- Legacy interfaces like KCS can be used, but what if the Redfish Host Interface never comes up? Should there be a timeout?
- System downtime is at a premium in the datacenter so a missed update will cause stale information to persist in the Redfish registry!

# UEFI Redfish Authentication



- To transfer the information UEFI needs similar inventory or configuration, it will need to authenticate with the Redfish Host 
- Should UEFI use a defined user/password? Would it be machine specific or model specific?
  - Think how easily both can be compromised!
- What about the OS, what credentials does it use to authenticate?

Redfish specification defines an auto authentication setup for FW/OS which is generated on every boot!

# Auto Authentication



How does UEFI retrieve the login credentials every boot?

- Legacy interfaces like KCS can be used again, but this requires the HW to support it

How does UEFI FW provide the OS their own login credentials in a secure manner?

- Is NVRAM secure enough?
- Remember that Redfish Host Interface is network, so OS must be sure to secure the network interface connected to the Redfish Host from other applications

# UEFI and Redfish: Working Together



- UEFI and Redfish are complementary specifications
- Each specification does not normally define HW requirements – much of the final implementation is up to the firmware provider
- Lots of data is transferred between UEFI and Redfish – be careful how and when data is transferred
  - Poorly designed implementations can effect product adoption
- The UEFI and Redfish specifications have many common members – expect many issues to be worked out in coming versions



**Questions?**