Redfish & RDE For Storage

Jeff Hilland

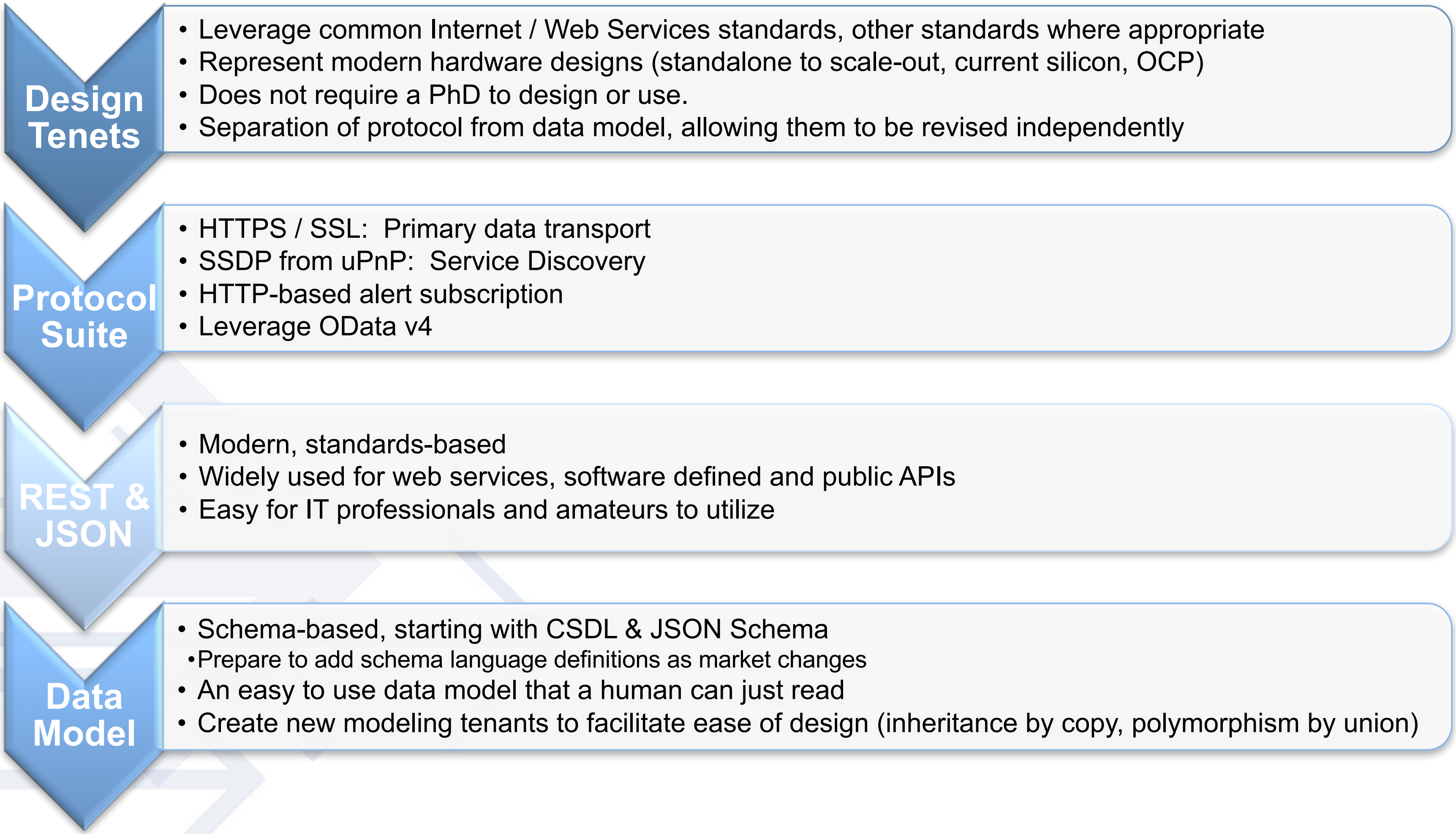
President, DMTF

Distinguished Technologist Manageability,
Hewlett Packard Enterprise

Disclaimer

- The information in this presentation represents a snapshot of work in progress within the DMTF.
- This information is subject to change without notice. The standard specifications remain the normative reference for all information.
- For additional information, see the DMTF website.

A Hybrid IT Management Solution



Design Tenets

- Leverage common Internet / Web Services standards, other standards where appropriate
- Represent modern hardware designs (standalone to scale-out, current silicon, OCP)
- Does not require a PhD to design or use.
- Separation of protocol from data model, allowing them to be revised independently

Protocol Suite

- HTTPS / SSL: Primary data transport
- SSDP from uPnP: Service Discovery
- HTTP-based alert subscription
- Leverage OData v4

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 Schema
  - Prepare to add schema language definitions as market changes
- An easy to use data model that a human can just read
- Create new modeling tenants to facilitate ease of design (inheritance by copy, polymorphism by union)

What is Redfish?

- **Industry Standard Software Defined Management for Converged, Hybrid IT**
  - HTTPS in JSON format based on OData v4
  - Schema-backed but human-readable
  - Equally usable by Apps, GUIs and Scripts
  - Extensible, Secure, Interoperable
- **Version 1 focused on Servers**
  - A secure, multi-node capable replacement for IPMI-over-LAN
  - Represent full server category: Rackmount, Blades, HPC, Racks, Future
  - Intended to meet OCP Remote Machine Management requirement
- **Expand scope over time to rest of IT infrastructure**
  - Additional features coming out approximately every 4 months
  - Working with SNIA to cover more advanced Storage (Swordfish)
  - Working with The Green Grid & ASHRAE to cover Facilities (Power/Cooling)
  - Work with the IETF to cover some level of Ethernet Switching

DMTF Redfish Forum

Co-Chairs: Jeff Autor (HPE), Mike Raineri (Dell)

Redfish Forum Leadership Companies

BROADCOM CISCO DELL ERICSSON Hewlett Packard Enterprise

intel Lenovo SUPERMICRO VERTIV vmware

Redfish Supporting Companies

American Megatrends Inc, ARM Inc, Artesyn Embedded Technologies, Cray Inc., Flex, Fujitsu, Huawei, IBM, Insyde Software Corp, Mellanox Technologies, Microsemi/Microchip, NetApp, OSIsoft LLC, Quanta Computer, Solarflare Communications, Toshiba, Western Digital Corporation

Redfish Industry Alliance Partners & efforts

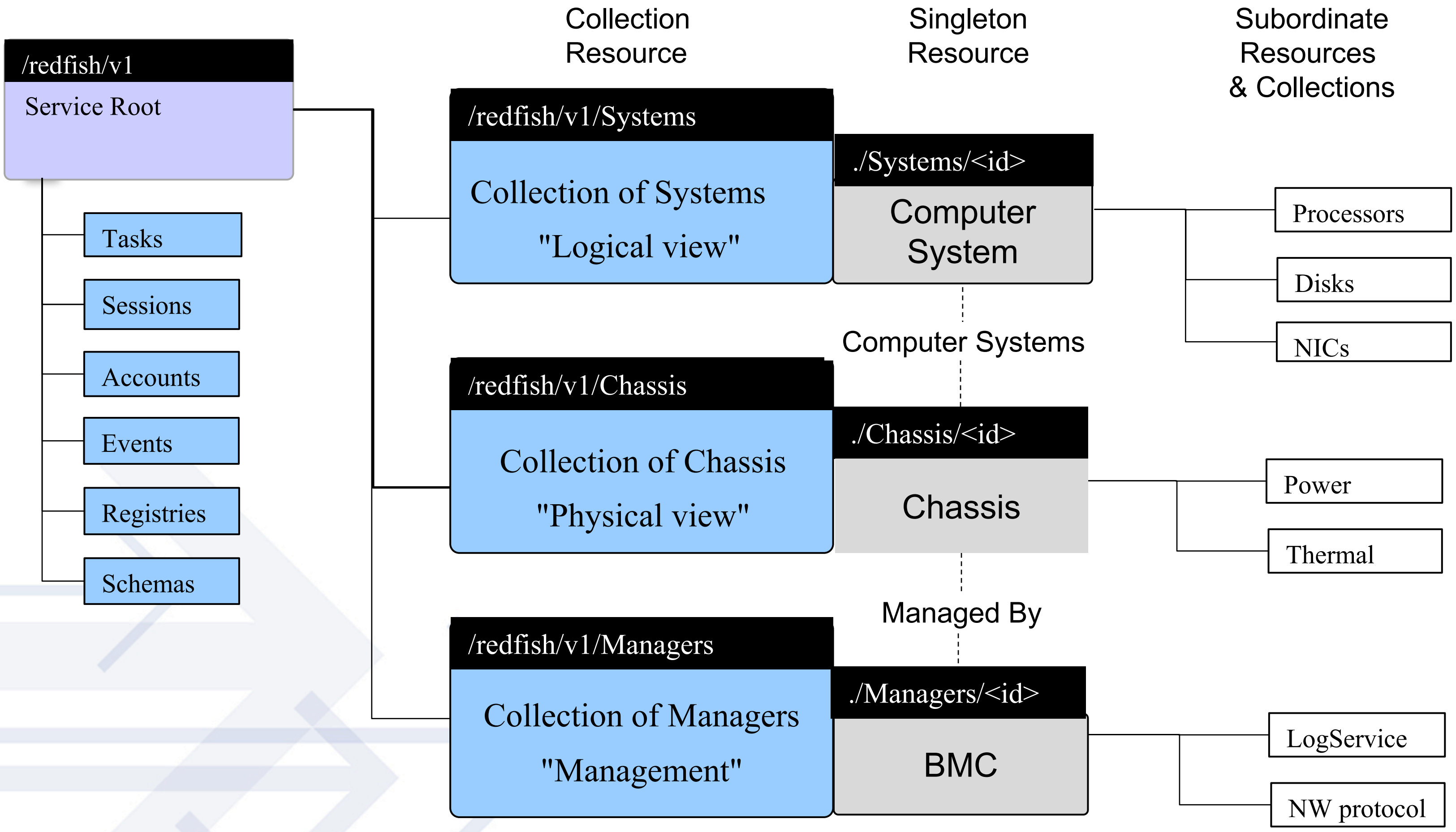
OCP (Open Compute Project) – Collaborating on profile definition
UEFI – Collaborating on Firmware Update and Host Interface work
SNIA – Collaborating on Storage modeling / alignment between SNIA SSM and Redfish
TGG – Pursuing relationship to work on Power/Cooling (existing DMTF Alliance Partner)
IETF – working on Switch modeling (no official alliance)

ASHRAE – American Society of Heating, Refrigerating and Air Conditioning Engineers
BBF – Broadband Forum
Gen-Z – Gen-Z Consortium
PICMG – Open Modular Computing for IIoT
NVMe – NVMe-MI

Timeline of Redfish® Specification

- The DMTF Redfish technology
  - Sep 2014: SPMF Formed in DMTF.
  - Aug 2015: Redfish Specification with base models (v1.0)
  - May 2016: Models for BIOS, disk drives, memory, storage, volume (2016.1)
  - Aug 2016: Models for endpoint, fabric, switch, PCIe device, zone, software/firmware inventory & update (2016.2)
  - Dec 2016: Adv. communications devices (multi-function NICs), host interface (KCS replacement), privilege mapping (2016.3)
  - May 2017: Composability (2017.1)
    - WIP for Telemetry
  - Aug 2017: Location, errata (2017.2)
    - WIPs for Ethernet Switching, DCIM, OCP & Profiles
  - Dec 2017: Profiles, Query parameters, errata (2017.3)
    - Feb 2018 – WIP for Redfish Device Enablement
  - Mar 2018: LDAP/AD, SSE, Assembly, minor enhancements & errata
  - Sept 2018: **OpenAPI**, Telemetry, Jobs, Schedule, Compose II, Message II
- Alignment with other standard organizations
  - Aug 2016: SNIA releases first model for network storage services (Swordfish)
  - Working open YANG Redfish mapping algorithm for Ethernet Switch
  - DMTF created work registers with UEFI, TGG, OCP, ASHRAE, Broadband Forum, ETSI-NFV, NVMe, PICMG, GenZ, ODCC for work on applying Redfish

Redfish Resource Map (simplified)

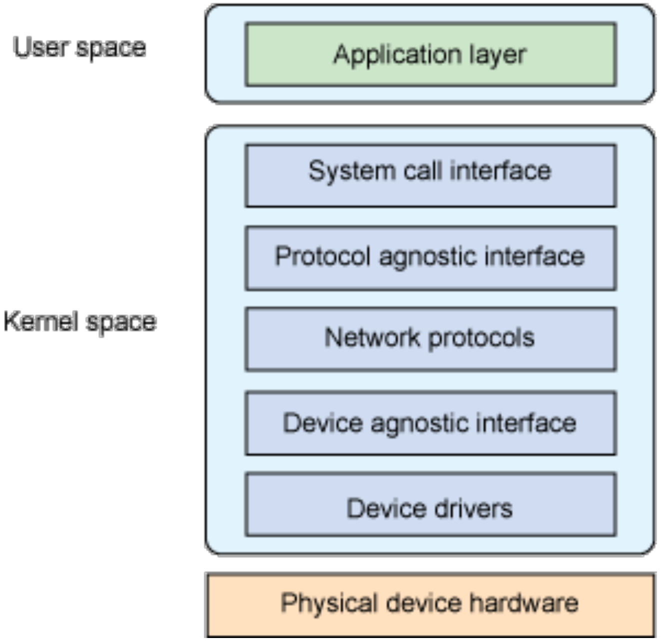


GET http://<ip-addr>/redfish/v1/Systems/{id}/Processors/{id}

Use the Redfish Resource Explorer (redfish.dmtf.org) to explore the resource map

Host Interface

- Replacement for IPMI KCS, etc.
- Exposes a NIC from Management Controller to OS
    - SMBIOS records provide information to allow kernel access
- Same access in-band as out-of-band
    - Kernel mode or user mode accessible
    - Encouraging OS vendors to begin consuming Redfish data.
    - This means you can get to the iLO homepage from the OS.
    - This means you can write your tools for the iLO homepage or Redfish and run them in the host OS.
    - Anything that accesses the out of band can be run on the host OS to access to local management subsystem.



Interoperability Profiles: Goals

- An “Interoperability Profile” provides a common ground for Service implementers, client software developers, and users
  - A profile would apply to a particular category or class of product (e.g. “Front-end web server”, “NAS”, “Enterprise-class database server”)
  - It specifies Redfish implementation requirements, but **is not** intended to mandate underlying hardware/software features of a product
  - Provides a target for implementers to meet customer requirements
  - Provide baseline expectations for client software developers utilizing Redfish
  - Enable customers to easily specify Redfish functionality / conformance in RFQs
- Create a machine-readable Profile definition
  - Document must be human-readable
  - Can be created by dev/ops personnel and non-CS professionals
- Enable authoring of Profiles by DMTF, partner organizations, and others
- Create open source tools to document and test conformance

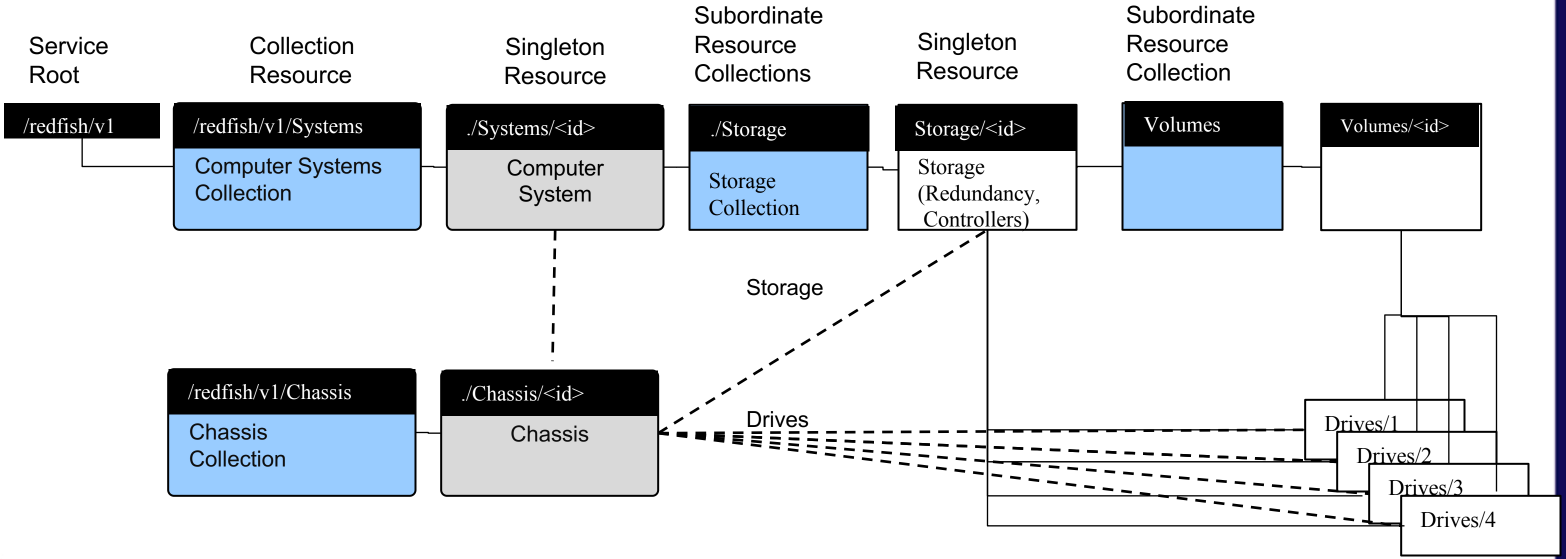
Redfish Storage Model

a.k.a. Local Storage, Storage “Lite”

Storage Resource Overview

- Storage: A representation of a storage sub-system
  - Contains sets of Volumes, Drives, and Storage Controllers
  - Storage Controller information is an array of objects in the Storage resource
    - Describes the protocols supported by the controller, the speed of the controller interface, and manufacturer information about the controller
- Drive: The physical media for the data
  - Manufacturer information about the drive (part number, serial number, etc)
  - Capability information about the drive (size, protocol, encryption, etc)
  - Contains control aspects (secure erase and LED setting)
- Volume: The logical construct used by the OS/hypervisor
  - Contains status about a volume (what drives contribute to the volume, size information, identifier information, etc)
  - Allows a client to control the volume (initialization, encryption settings, etc)

Server Storage in Redfish



Note that the Volumes are in Collections off of the Storage resource, drives are in arrays off of the storage resource and optionally the Chassis.

RDE

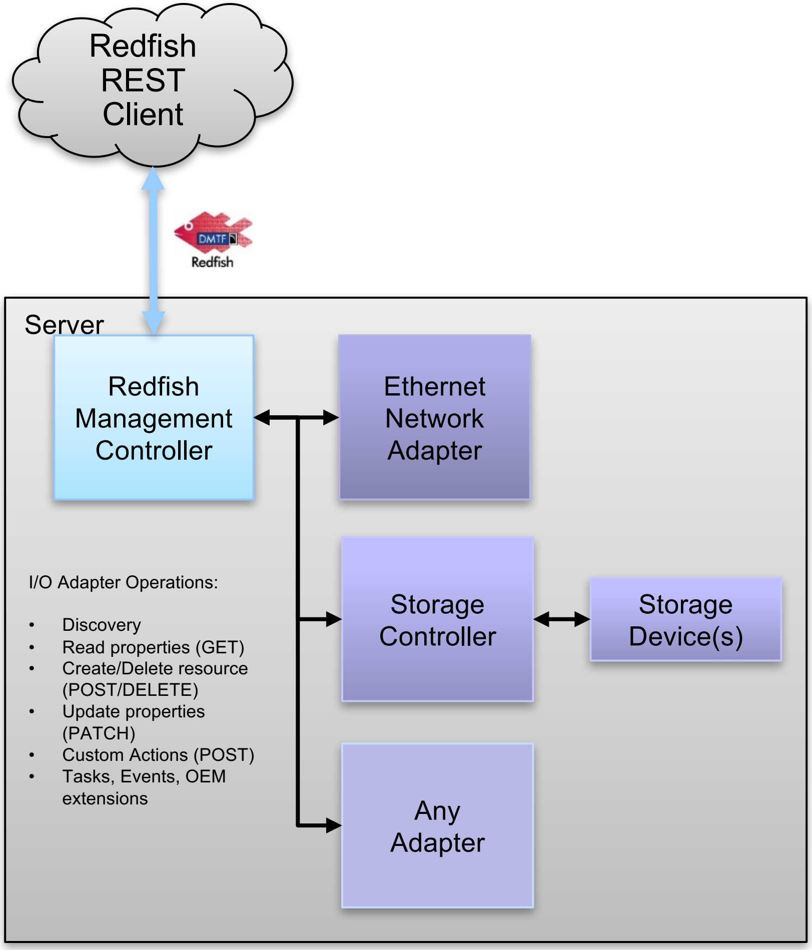
Or

"How you can fill all that storage stuff out without creating a lock step firmware dependency between the management controller firmware and the storage firmware"

Redfish Device Enablement: PLDM Redfish Providers

PMCI WG developing a standard to enable a server Management Controller to present a Redfish-conformant management of I/O Adapters without building in code specific to each adapter family/vendor/model.

- Support adapter "self-contained, self-describing" including value-add (OEM) properties
- New managed devices (and device classes) do not require Management Controller firmware updates
- Support a range of capabilities from primitive to advanced devices (lightweight/low bandwidth options)
- Leveraging PLDM, a provider architecture is being specified that can binary encode the data in a small enough format for devices to understand and support.
- MC acts as a broker to encode/decode the data to/from the provider
- PLDM works over I2C & PCIe VDM. Additional mappings under consideration.



I/O Adapter Operations:

- Discovery
- Read properties (GET)
- Create/Delete resource (POST/DELETE)
- Update properties (PATCH)
- Custom Actions (POST)
- Tasks, Events, OEM extensions

RDE Discovery & Registration

Discovery is PLDM based

- Devices are discovered using PLDM (which uses MCTP) and determines that it supports RDE (PLDM Type 6)
- MC uses RDE to negotiate parameters with the device.
  - Concurrency, Operations Supported, Provider Name
- MC uses RDE to negotiate channel parameters on each channel
  - Asynchrony, Max Chunk Size

Registration leverages Platform Data Record

- MC queries device for PDRs and any Associations and Actions
  - PDR = Platform Data Record. Equates to one or more Redfish Resources
- And gets Schema Identities and versions for the PDR
  - Can get a Resource ETag
- And Retrieves any Dictionary for the PDR
  - Dictionaries may be truncated to only what device supports

RDE BEJ – Translating JSON to binary using dictionaries

- RDE's BEJ (Binary Encoded JSON) is a binary representation of the JSON payload using the algorithm specified
- Dictionaries are the key
- So the dictionary is used to turn each of these into a compact form.
  - RDE uses the dictionary to take the JSON Body and turn it into binary
  - Includes how to nest objects, handle annotations and other Redfish nuances
  - Roughly a 10:1 compression in early analysis
- DMTF will publish Dictionaries for all DMTF Schema on the website
  - Program to generate dictionaries will be made open source

RDE Operations

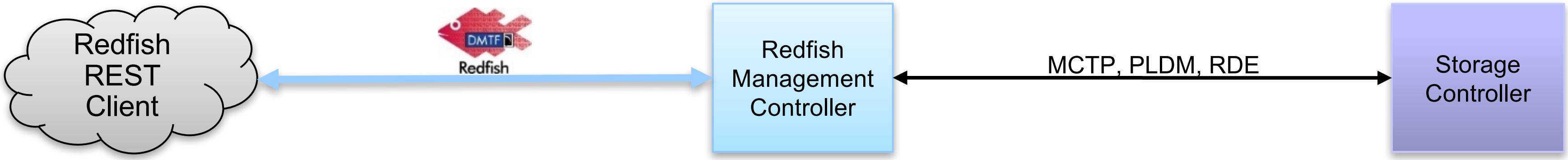
- RDE has slightly different terms than HTTP in the spec

HTTP Operation	RDE Operation
GET	Read
PUT	Replace
PATCH	Update
POST	Action or Create
DELETE	Delete
HEAD	Read Headers

- RDE supports multiple outstanding operations, tasks and events

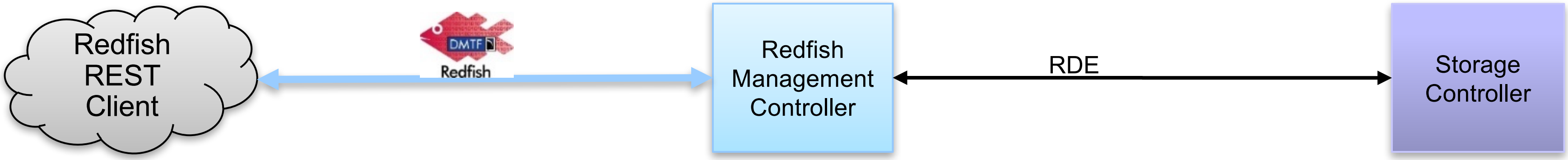
Command	Usage
RDEOperationInit	Begin an Operation
SupplyCustomRequestParameters	Provide additional parameters for Operations
RetrieveCustomResponseParameters	Get additional response data
RDEOperationStatus	Check up on an active Operation
RDEOperationComplete	Finalize an Operation
RDEOperationKill	Cancel an Operation
RDEOperationEnumerate	See what Operations are active
MultipartSend, MultipartReceive	Bulk data transfer

So how does all this fit together?



- Before a Client ever Contacts the Redfish Service:
  - The Management Controller uses MCTP to enumerate the devices
    - This is dependent on the Medium (I2C, PCIe, etc.)
  - MC then uses PLDM for next phase of discovery
  - PLDM support for Type 6 (RDE) is discovered as being supported!
  - RDE Discovery takes place over PLDM
  - RDE Device Registration
    - Get all the Resources, Actions, and Entity Associations
    - MC Retrieves the Dictionaries
- The MC is now ready to handle operations to and from the device

So how does all this fit together?



- Request comes in for client
- MC determines that the URI equates to an RDE provider.
- MC translates HTTP headers that need to be passed down
- The MC translates the JSON body into BEJ using the dictionaries.
- MC initiates the operation to the device.
- Device processes the request
- MC takes the response, reassembles it and translates from BEJ to JSON using the dictionaries.
- MC formulates HTTP Response and sends to Client

Additional information about RDE

- RDE also handles Tasks,
  specifies how Events are handled,
  and has information on handling OEM sections.

- There are state machine examples and tables.
- Binary Format for dictionaries is specified

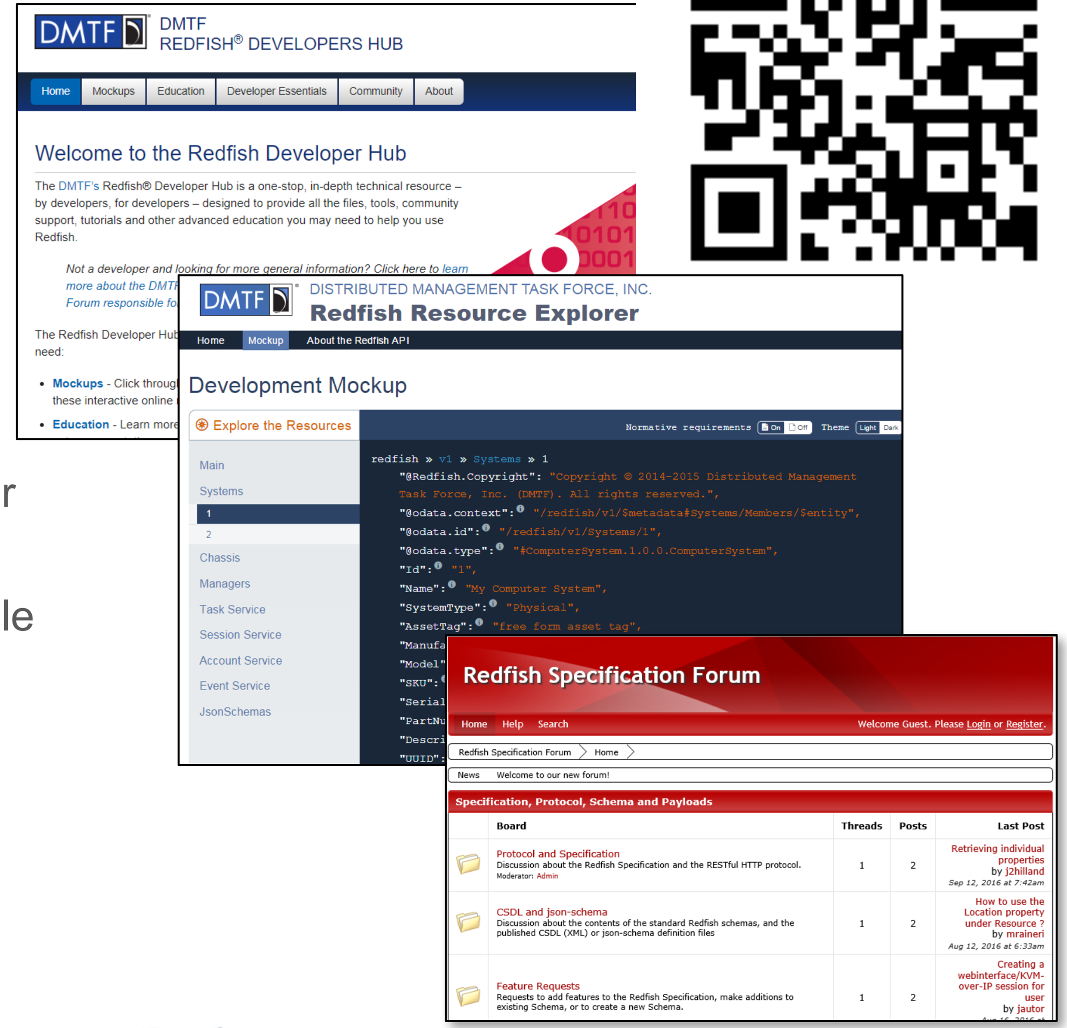
- There are examples of
  - what a Redfish tree would look like,
  - how the PDR would look like for that tree,
  - Examples of what a dictionary would look like
  - Example of BEJ encode/decode.

In Summary

- Redfish, along with the DMTF alliance partners, is working to define interoperable software defined hybrid IT management for servers, storage, networking, power/cooling, fabrics and more
- And is solving problems from composition to resource managers, aggregation engines
- As well as plumbing the mechanisms inside the box to be self contained and self describing

Redfish Developer Hub: redfish.dmtf.org

- Resources
  - Schema Index
  - Specifications
  - GitHub for Redfish Tools
  - Registries
  - Other Documentation
- Mockups
  - Simple Rack-mounted Server
  - Bladed System
  - Proposed OCP Redfish Profile
  - More being added
- Education/Community
  - Redfish User Forum
  - Whitepapers, Presentations
  - YouTube shorts & Webinars



DMTF
REDFISH® DEVELOPERS HUB

Welcome to the Redfish Developer Hub

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.

Not a developer and looking for more general information? Click here to learn more about the DMTF
Forum responsible fo

The Redfish Developer Hub
need:

- **Mockups** - Click throug
these interactive online
- **Education** - Learn more

DISTRIBUTED MANAGEMENT TASK FORCE, INC.
Redfish Resource Explorer

Development Mockup

```
redfish » v1 » Systems » 1
    "@Redfish.Copyright": "Copyright © 2014-2015 Distributed Management
    Task Force, Inc. (DMTF). All rights reserved.",
    "@odata.context": "/redfish/v1/$metadata#Systems/Members/$entity",
    "@odata.id": "/redfish/v1/Systems/1",
    "@odata.type": "#ComputerSystem.1.0.0.ComputerSystem",
    "Id": "1",
    "Name": "My Computer System",
    "SystemType": "Physical",
    "AssetTag": "free form asset tag",
    "Manufa
    "Model"
    "SKU":
    "Serial
    "PartNu
    "Descri
    "UUID":
```

Redfish Specification Forum

Welcome Guest. Please Login or Register.

Redfish Specification Forum › Home

News Welcome to our new forum!

Specification, Protocol, Schema and Payloads

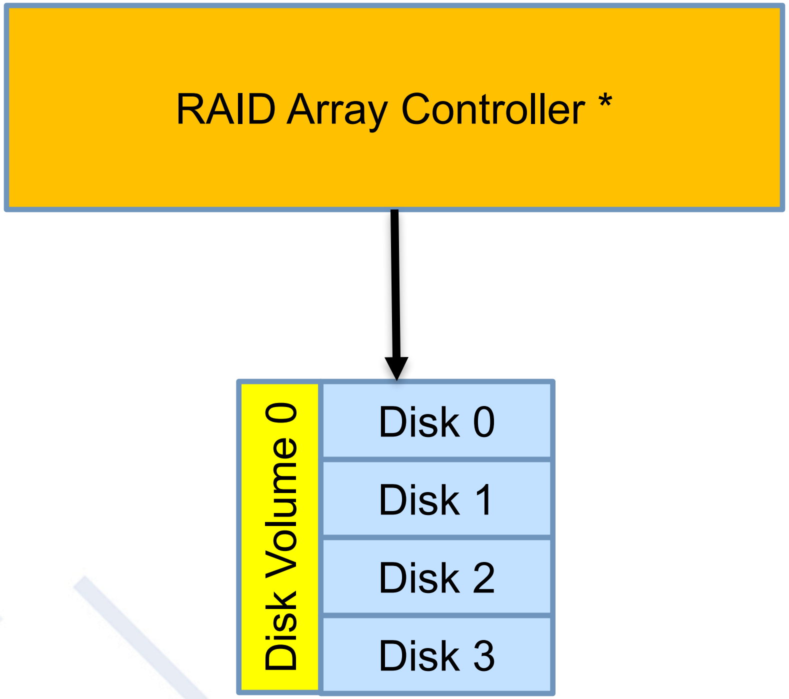
Board	Threads	Posts	Last Post
Protocol and Specification Discussion about the Redfish Specification and the RESTful HTTP protocol. Moderator: Admin	1	2	Retrieving individual properties by j2hilland *Sep 12, 2016 at 7:42am*
CSDL and json-schema Discussion about the contents of the standard Redfish schemas, 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*
Feature Requests 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

Thank you!

Redfish

Backup Material

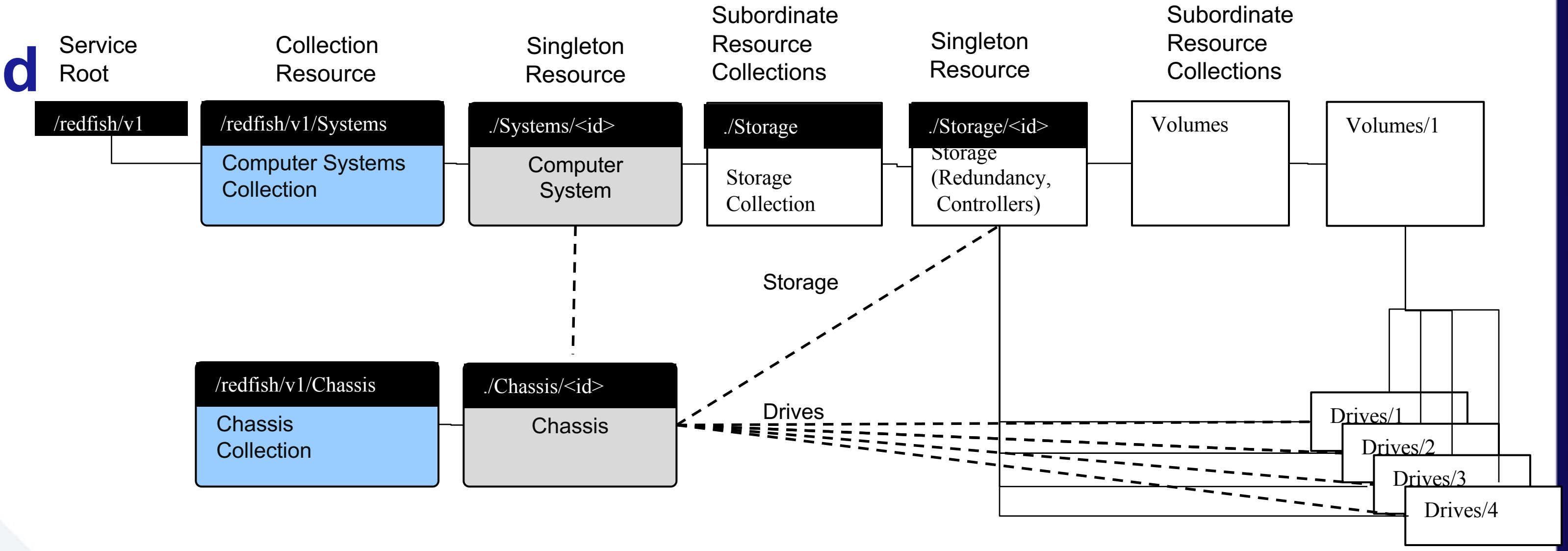
Disk Subsystem (Physical Layout)



* For this example, we're assuming that the raid array controller firmware exposes management for everything seen here

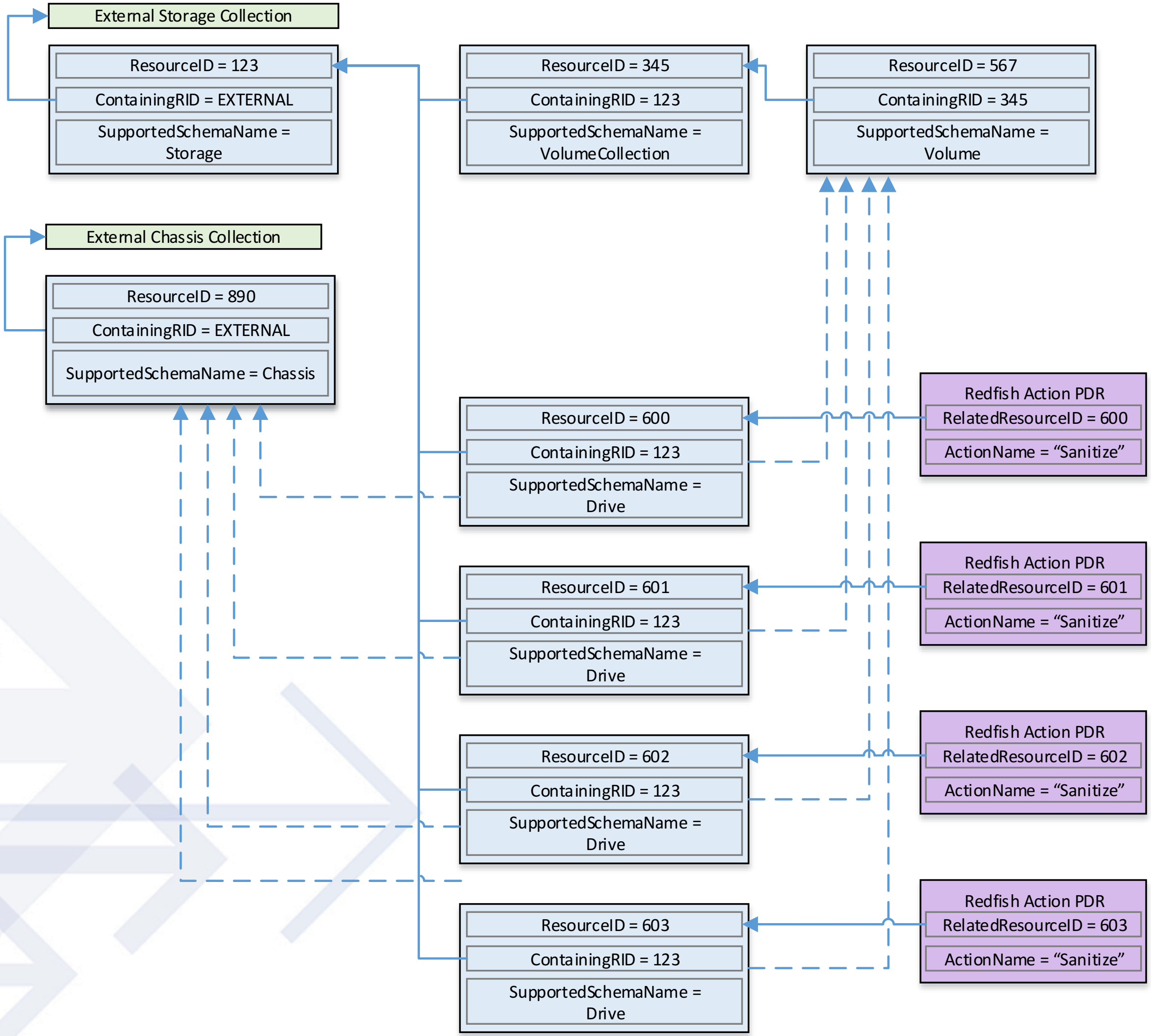
Server Storage in Redfish

Gets Modeled Like This:



Note that the Volumes are in Collections off of the Storage resource, drives are in arrays off of the storage resource and optionally the Chassis.

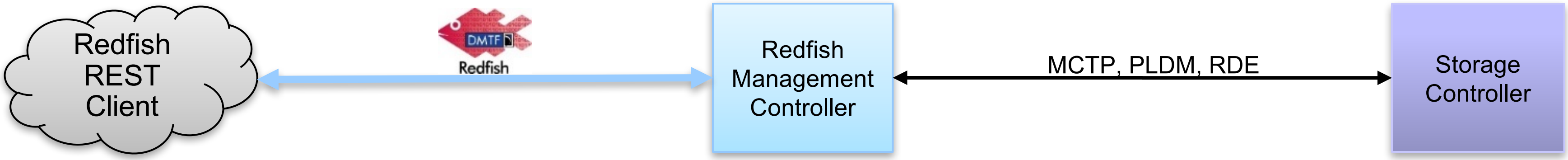
PDRs would look like this:



RDE BEJ – Translating JSON to binary using dictionaries

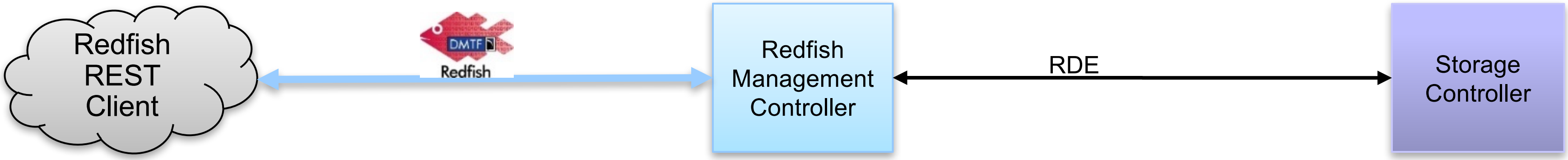
- RDE’s BEJ (Binary Encoded JSON) is a binary representation of the JSON payload using the algorithm specified.
- Dictionaries are the key
    - Truncated representation of the Schema
        - One dictionary per schema
        - 100% backward compatible! Can use a newer dictionary on older resources.
    - Used to turn property names and objects into numbers
    - Used to turn enums into numbers
    - Numbers are already easy to represent in binary.
        - Real can be tricky but there are few of them.
    - String values stay strings
- JSON objects are made up of
    - Properties, objects, numbers, enums, strings, null
- So the dictionary is used to turn each of these into a compact form.
    - RDE uses an algorithm that uses the dictionary to take the JSON Body and turn it into binary.
    - SFLV Tuples – Sequence Number, Format, Length, Value.
        - Sequence number equates to property and is relative to level.
    - Includes how to nest objects, handle annotations and other Redfish nuances
    - Roughly a 10:1 compression
- DMTF will publish Dictionaries for all DMTF Schema on the website
    - Program to generate dictionaries will be made open source

So how does all this fit together?



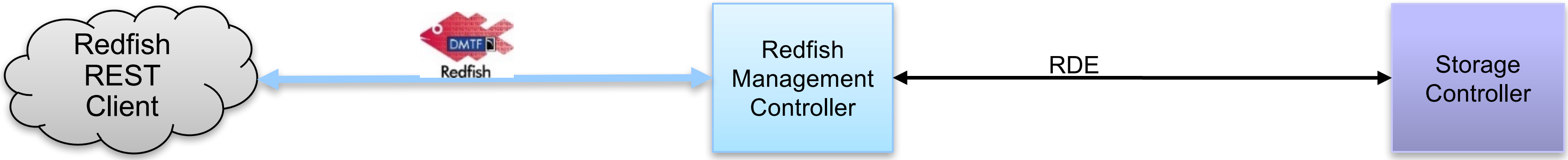
- Before a Client ever Contacts the Redfish Service:
  - The Management Controller uses MCTP to enumerate the devices
    - DSP0236 and mapping specs cover this in detail.
    - Phases: Bus enumeration, Bus address assignment, MCTP capabilities, EID assignment, routing info
  - This is dependent on the Medium (I2C, PCIe, etc.)
    - See the MCTP Mapping Specs for details
  - MC then uses PLDM for next phase of discovery
    - DSP0240 covers this in detail
    - Uses SetTID, GetTID, GetPLDMVersion, GetPLDMTypes, GetPLDMCommands
  - PLDM support for Type 6 (RDE) is discovered as being supported!
    - Other PLDM Types like FW Update and Monitoring & Control will be discovered and can be used by the MC and exported through Redfish *BUT* these are all specialized providers in the MC

So how does all this fit together?



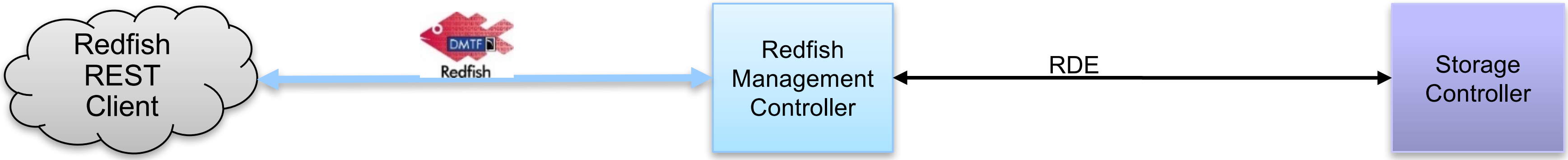
- Before a Client ever Contacts the Redfish Service:
  - RDE Discovery takes place over PLDM
    - NegotiateRedfishParameters to get the Provider name, concurrency support and features supported
    - NegotiateMediumParameters on each medium it plans to communicate on
  - RDE Device Registration
    - GetPDRRepositoryInfo & GetPDR to get PDRs for Resources, Actions, and Entity Associations
      - This is used to build the topology of URIs and links.
      - The ContainingRID is used to build the hierarchy.
    - MC Retrieves the Dictionaries
      - Dictionaries for the data (a.k.a. the major schema), Event, Annotation, Extended Info
- The MC is now ready to handle operations to and from the device

So how does all this fit together?



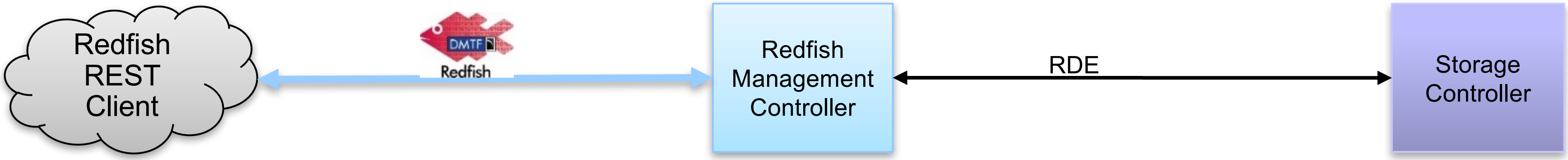
- Request comes in for client (we will use a GET)
  - MC determines that the URI equates to an RDE provider.
  - MC translates the URI into the EID and Resource ID needed to address it.
- MC translates HTTP headers that need to be passed down
  - Not all of them are sent but things like If-Match have to be
  - Query parameters are handled too.
  - Headers not specified by Redfish or RDE are also accommodated.
- The MC translates the JSON body into BEJ using the dictionaries.
  - Each element is turned into an SFLV tuple.
  - Algorithm is in the spec. Surprisingly small amount of code.
  - Accommodate deferred binding (substitution values).
    - Things like Links, Systems, $metadata, etc.

So how does all this fit together?



- MC initiates the operation to the Device.
    - Operation then follows a state diagram.
    - RDEOperationInit
        - This message has the ResourceID, OperationID, OperationType (Read), and any OperationFlags (such as customer request parameters).
        - Device responds
    - SupplyCustomRequestParameters would be used next (if needed)
        - Device responds if all went well. Example, Etag matched so proceed.
    - Since this is a GET, MultipartReceive would be used if bigger than one message.
        - Success response includes the data until done.
    - RetrieveCustomResponseParameters is executed if indicated.
        - Success response includes the data until done.
    - Operation is complete when RDEOperationComplete command.
        - Success indicates OperationID can be used for other operations

So how does all this fit together?



- MC then takes the response, reassembles it and translates from BEJ to JSON using the dictionaries.
- MC formulates HTTP Response and sends to Client