



Redfish[®]
Reference Guide

Revision 2.0

The information in this USER'S MANUAL has been carefully reviewed and is believed to be accurate. The vendor assumes no responsibility for any inaccuracies that may be contained in this document, makes no commitment to update or to keep current the information in this manual, or to notify any person organization of the updates. Please Note: For the most up-to-date version of this manual, please see our web site at www.supermicro.com.

Super Micro Computer, Inc. ("Supermicro") reserves the right to make changes to the product described in this manual at any time and without notice. This product, including software, if any, and documentation may not, in whole or in part, be copied, photocopied, reproduced, translated or reduced to any medium or machine without prior written consent.

IN NO EVENT WILL SUPERMICRO BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, SPECULATIVE OR CONSEQUENTIAL DAMAGES ARISING FROM THE USE OR INABILITY TO USE THIS PRODUCT OR DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN PARTICULAR, SUPERMICRO SHALL NOT HAVE LIABILITY FOR ANY HARDWARE, SOFTWARE, OR DATA STORED OR USED WITH THE PRODUCT, INCLUDING THE COSTS OF REPAIRING, REPLACING, INTEGRATING, INSTALLING OR RECOVERING SUCH HARDWARE, SOFTWARE, OR DATA.

Any disputes arising between manufacturer and customer shall be governed by the laws of Santa Clara County in the State of California, USA. The State of California, County of Santa Clara shall be the exclusive venue for the resolution of any such disputes. Super Micro's total liability for all claims will not exceed the price paid for the hardware product.

Information in this document is subject to change without notice. Other products and companies referred to herein are trademarks or registered trademarks of their respective companies or mark holders.

Copyright © 2017 by Super Micro Computer, Inc.

All rights reserved.

Printed in the United States of America

Manual Revision 2.0

Release Date: March 20, 2018

Unless you request and receive written permission from Super Micro Computer, Inc., you may not copy any part of this document.

Revision History

Date	Rev.	Description
August 10, 2015	1.0	Created document.
October 5, 2015	1.0a	Minor formatting modifications.
June 5, 2017	1.0b	Added more APIs to section 2.3 Added and modified list of OEM APIs (Section 3.6). Added content to Section 3.7 Modifies screenshots in Chapter 4 Modified reference links in Chapter 5
March 20, 2018	2.0	Added Section 4 (Update service) Modified Section 3.6 (OEM apis) Added new APIs in Section 2.3 Added new examples/screenshots in Section 5

Table of Contents

1 Introduction.....	6
2 HTTP Request Methods.....	6
2.1 Responses.....	7
2.2 HTTP Status Code Description.....	7
2.3 List of Available APIs.....	8
3 Using RESTful APIs.....	12
3.1 Authentication.....	12
3.1.1 Basic Authentication.....	12
3.1.2 Session Management.....	13
3.2 Account Service.....	14
3.3 Event Service.....	14
3.4 Registries.....	15
3.5 Jschema.....	15
3.6 OEM APIs.....	15
3.6.1 SMTP.....	15
3.6.2 FanMode.....	16
3.6.3 Active Directory.....	16
3.6.4 Get/Set iKVM Mouse Mode.....	16
3.6.5 Reset to Factory Default.....	17
3.6.6 NTP.....	17
3.6.7 RADIUS.....	17
3.6.8 LDAP.....	17
3.6.9 Snooping.....	17
3.6.10 IP Access Control.....	17
3.6.11 SMCRAKP.....	17
3.6.12 SNMP.....	18
3.6.13 Syslog.....	18
3.6.14 Chassis Intrusion.....	18
3.6.15 RAID Management Reference Examples.....	19
3.6.16: iKVM.....	20

3.7 BIOS Configurations: Configure BIOS over Redfish	20
4 UpdateService	23
4.1 Update SSL certificate and key	23
4.2 BIOS Update	23
4.3 BMC Firmware Update	25
5 Examples.....	26
5.1 Posting an action:	26
5.2 Getting mac address from AOC	27
5.3 Memory info through Redfish API:	28
5.4 Redfish API Response for drive connected to 3108 controller	29
5.5 Python Code for Redfish API Response	30
6 Reference Links.....	30

1 Introduction

The Redfish Scalable Platforms Management API ("Redfish") is a new interface that uses RESTful interface semantics to access data defined in a model format to perform out-of-band systems management. It is suitable for a wide range of servers, from stand-alone to rack mount and blade environments, but scales equally well for large scale cloud environments.

Redfish is a management standard which uses data model representation inside of a hypermedia RESTful interface. It is based on REST, that's how Redfish is easier to use and implement than many other solutions. Since its model oriented, it is capable of expressing the relationships between components in modern systems as well as the semantics of the services and components within them. It is also easily extensible. By using a hypermedia approach to REST, Redfish can express a large variety of systems from multiple vendors. Utilizing JSON (JavaScript Object Notation) data format which is in plain text, allows many types of parameters to be available such that it enables scalability, human readability, and flexibility for most programming environments by easily interpreting payload.

The model is exposed in terms of an interoperable OData Schema with the payload of the messages being expressed in JSON following OData JSON conventions. The schema (available in both XML and JSON formats) includes annotations to facilitate the automatic translation of the schema to JSON Schema. The ability to externally host the schema definition of the resources in a machine-readable format allows the meta data to be associated with the data without encumbering Redfish services with the meta data, thus enabling more advanced client scenarios as found in many data center and cloud environments.

Supermicro enables Redfish feature sets on their X10/X11 platforms with 3.xx and 1.xx BMC firmware respectively. These features are covered under SFT-OOB-LIC and SFT-DCMS-SINGLE license. This document will provide you with an overview of Restful API services and describe how to receive Redfish API responses directly from a Supermicro BMC (Baseboard Management Controller).

2 HTTP Request Methods

The following HTTP methods are used to implement different actions, as described below.

- Read Requests (GET):

The GET method is used to request a representation of a specified resource. The representation can be either a single resource or a collection.

- Update (PATCH):

The PATCH method is used to apply partial modifications to a resource.

- Replace (PUT):

The PUT method is used to completely replace a resource. Any properties omitted from the body of the request are reset to their default value.

- Create (POST):

The POST method is used to create a new resource. This request is submitted to the resource collection in which the new resource is meant to belong.

- Actions (POST):

The POST method may also be used to initiate operations on the object (Actions). The POST operation may not be idempotent.

- Delete (DELETE):

The DELETE method is used to remove a resource.

2.1 Responses

Four types of responses are supported, as defined below.

- Metadata Responses:

These describe the resources and types exposed by the service to generic clients.

- Resource Responses:

JSON representation of an individual resource.

- Resource Collection Responses:

JSON representation of a collections of resources.

- Error Responses:

Top-level JSON response providing additional information in the case of an HTTP error.

2.2 HTTP Status Code Description

Status Code	Description
200	OK
201	Created
202	Accepted
204	No Content
301	Moved permanently
302	Found
304	Not Modified
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
415	Unsupported Media Type

500	Internal Server Error
501	Not Implemented
503	Service Unavailable

2.3 List of Available APIs

API List	Notes:
/redfish/v1	Service root
/redfish/v1/SessionService	
/redfish/v1/Chassis	
/redfish/v1/AccountService	
/redfish/v1/Managers	
/redfish/v1/Systems	
/redfish/v1/EventService	
/redfish/v1/UpdateService	
/redfish/v1/Registries	
/redfish/v1/JsonSchemas	
/redfish/v1/SessionService/Sessions	
/redfish/v1/SessionService/Sessions/[session_num]	
/redfish/v1/Chassis/1	
/redfish/v1/Chassis/1/Thermal	
/redfish/v1/Chassis/1/Power	
/redfish/v1/Chassis/HA-RAID.[controller_num].StorageEnclosure.[enclosure_num]	For LSI 3108
/redfish/v1/Chassis/HA-RAID.[contoller_num].StorageEnclosure.[enclosure_num]/Drives/Disk.Bay.[disk_num]	For LSI 3108
/redfish/v1/Chassis/HA-RAID.[controller_num].StorageEnclosure.[enclosure_num]/Drives/Disk.Bay.[disk_num]/Actions/Oem/Drive.Indicate	Light on physical drive indication LED {"Active": "true"}
/redfish/v1/Chassis/HBA.[contoller_num].StorageEnclosure.[enclosure_num]	For LSI 3008
/redfish/v1/Chassis/HBA.[contoller_num].StorageEnclosure.[enclosure_num]/Drives/Disk.Bay.[disk_num]	For LSI 3008
/redfish/v1/Chassis/HBA.[contoller_num].StorageEnclosure.[enclosure_num]/Drives/Disk.Bay.[disk_num]/Actions/Oem/Drive.Indicate	Light on physical drive indication LED {"Active": "true"}
/redfish/v1/Chassis/StorageBackplane	For PCH SATA or RSTe, TAS must be running
/redfish/v1/Chassis/StorageBackplane/Drives/Disk.Bay.[disk_num]	For PCH SATA or RSTe, TAS must be running
/redfish/v1/Chassis/NVMeSSD.[pcie_controller_num].Group.[group_num].StorageBackplane	For NVMe

/redfish/v1/Chassis/NVMeSSD.[pcie_controller_num].Group.[group_num].StorageBackplane/Drives/Disk.Bay.[disk_num]	For NVMe
/redfish/v1/AccountService/Roles	
/redfish/v1/AccountService/Roles/Admin	
/redfish/v1/AccountService/Roles/Operator	
/redfish/v1/AccountService/Roles/ReadOnlyUser	
/redfish/v1/AccountService/Roles/Custom1	
/redfish/v1/AccountService/Accounts	
/redfish/v1/AccountService/Accounts/[account_num]	
/redfish/v1/Managers/1	
/redfish/v1/Managers/1/Actions/Manager.Reset	BMC cold reset
/redfish/v1/Managers/1/Actions/Oem/ManagerConfig.Reset	BMC factory default
/redfish/v1/Managers/1/SerialInterfaces	
/redfish/v1/Managers/1/NetworkProtocol	
/redfish/v1/Managers/1/LogServices	
/redfish/v1/Managers/1/LogServices/Log1	
/redfish/v1/Managers/1/LogServices/Log1/Actions/LogService.Reset	Clear event logs
/redfish/v1/Managers/1/LogServices/Log1/Entries	
/redfish/v1/Managers/1/LogServices/Log1/Entries/[log_num]	
/redfish/v1/Managers/1/VM1	
/redfish/v1/Managers/1/VM1/CfgCD	Configure ISO image settings: host, path, username/pass
/redfish/v1/Managers/1/VM1/CfgCD/Actions/IsoConfig.Mount	Mount ISO image
/redfish/v1/Managers/1/VM1/CfgCD/Actions/IsoConfig.UnMount	Unmount ISO image
/redfish/v1/Managers/1/VM1/CD[mounted_dev_num]	User must first mount image
/redfish/v1/Managers/1/VM1/Floppy[mounted_dev_num]	User must first mount image
/redfish/v1/Managers/1/VM1/USB[mounted_dev_num]	User must first mount image
/redfish/v1/Managers/1/EthernetInterfaces	
/redfish/v1/Managers/1/EthernetInterfaces/[eth_num]	
<Managers OEM APIs>	
/redfish/v1/Managers/1/SNMP	
/redfish/v1/Managers/1/SNMP/SNMPv2	
/redfish/v1/Managers/1/SNMP/SNMPv3	

/redfish/v1/Managers/1/FanMode	
/redfish/v1/Managers/1/MouseMode	
/redfish/v1/Managers/1/Snooping	
/redfish/v1/Managers/1/ActiveDirectory	
/redfish/v1/Managers/1/ActiveDirectory/RoleGroups	
/redfish/v1/Managers/1/ActiveDirectory/RoleGroups/[role_group]	
/redfish/v1/Managers/1/SMTP	
/redfish/v1/Managers/1/Syslog	
/redfish/v1/Managers/1/RADIUS	
/redfish/v1/Managers/1/MouseMode	
/redfish/v1/Managers/1/LDAP	
/redfish/v1/Managers/1/SMCRACKP	
/redfish/v1/Managers/1/IPAccessControl	
/redfish/v1/Managers/1/IPAccessControl/FilterRule	
/redfish/v1/Managers/1/IPAccessControl/FilterRule/[rule_num]	
/redfish/v1/Managers/1/NTP	
/redfish/v1/Managers/1/iKVM	Get a URL link to launch iKVM/HTML5
/redfish/v1/Systems/1	
/redfish/v1/Systems/1/Actions/ComputerSystem.Reset	System reset
/redfish/v1/Systems/1/Processors	
/redfish/v1/Systems/1/Processors/[processor_num]	
/redfish/v1/Systems/1/Memory	
/redfish/v1/Systems/1/Memory/[memory_num]	
/redfish/v1/Systems/1/EthernetInterfaces	
/redfish/v1/Systems/1/EthernetInterfaces/[eth_num]	Data from BIOS and TAS
/redfish/v1/Systems/1/SimpleStorage	
/redfish/v1/Systems/1/SimpleStorage/[controller_num]	
/redfish/v1/Systems/1/Storage	
/redfish/v1/Systems/1/Storage/HA-RAID	For LSI 3108
/redfish/v1/Systems/1/Storage/HA-RAID/Volumes	For LSI 3108
/redfish/v1/Systems/1/Storage/HA-RAID/Volumes/Controller.[controller_num].Volume.[volume_num]	For LSI 3108

/redfish/v1/Systems/1/Storage/HA-RAID/Volumes/Controller.[controller_num].Volume.[volume_num]/Actions/Oem/Volume.Indicate	For LSI 3108; light on virtual drive indication LED
/redfish/v1/Systems/1/Storage/HA-RAID/Volumes/Controller.[controller_num].Volume.[volume_num]/Actions/Oem/Volume.Delete	For LSI 3108; in logical view to delete specific virtual drive
/redfish/v1/Systems/1/Storage/HA-RAID/Actions/Oem/Storage.CreateVolume	For LSI 3108; create virtual drives
/redfish/v1/Systems/1/Storage/HA-RAID/Actions/Oem/Storage.ClearVolumes	For LSI 3108; in logical view to clear all configurations
/redfish/v1/Systems/1/Storage/HA-RAID/Actions/Oem/HARAIDController.Save	For LSI 3108; save controller's "BIOS Boot Mode"
/redfish/v1/Systems/1/Storage/HBA	For LSI 3008
/redfish/v1/Systems/1/Storage/RAIDIntegrated	For RSTe, TAS must be running
/redfish/v1/Systems/1/Storage/RAIDIntegrated/Volumes	For RSTe, TAS must be running
/redfish/v1/Systems/1/Storage/RAIDIntegrated/Volumes/[volume_num]	For RSTe, TAS must be running
/redfish/v1/Systems/1/Storage/SATAEmbedded	For PCH SATA, TAS must be running
/redfish/v1/Systems/1/Storage/SATAEmbedded/Volumes	For PCH SATA, TAS must be running
/redfish/v1/Systems/1/Storage/SATAEmbedded/Volumes/[volume_num]	For PCH SATA, TAS must be running
/redfish/v1/Systems/1/Bios	BIOS current settings (Only X11DP supports)
Note: Redfish BIOS configuration supported on following platforms X11DPU, X11DPU_PLUS, X11DDW, X11DPT-B, X11DPT-B PLUS, X11DPI, X11DDW	
/redfish/v1/Systems/1/Bios/SD	BIOS pending settings (only X11DP supports)
/redfish/v1/Systems/1/Bios/Actions/Bios.ResetBios	Reset BIOS settings to default (only X11DP supports)
/redfish/v1/Systems/1/Bios/Actions/Bios.ChangePassword	Change BIOS booting password (only X11DP supports)
/redfish/v1/EventService/Subscriptions	
/redfish/v1/EventService/Subscriptions/[destination_num]	
/redfish/v1/UpdateService/SSLCert	View current SSL certification info
/redfish/v1/UpdateService/SSLCert/Actions/SSLCert.Upload	Used to upload new SSL certification file

/redfish/v1/UpdateService/FirmwareInventory	Supported on X11 platforms
/redfish/v1/UpdateService/FirmwareInventory/BMC	
/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.EnterBMCUpdateMode	
/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.UploadBMC	
/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.UpdateBMC	
/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.CancelBMC	
/redfish/v1/UpdateService/FirmwareInventory/BIOS	
/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.EnterBIOSUpdateMode	
/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.UploadBIOS	
/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.UpdateBIOS	
/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.CancelBIOS	(Only X11DP supports)
/redfish/v1/Registries/Base.1.0.0	
/redfish/v1/Registries/BiosAttributeRegistry.1.0.0.json	
/redfish/v1/JsonSchemas/[variety_of_services]	

3 Using RESTful APIs

User can receive API responses through programming, by installing Postman or any other Rest API client application(s)

3.1 Authentication

Redfish supports both "Basic Authentication" and "Redfish Session Login Authentication" (as described below under Session Management). Service does not require a client to create a session when Basic Authentication is used.

3.1.1 Basic Authentication

HTTP BASIC authentication uses compliant TLS connections to transport the data between any third party authentication service and clients.

Note: Always check the status code once you get the response from the Redfish URL. You can refer to the status code table mentioned above. (All URLs/commands are case sensitive.)

3.1.2 Session Management

Redfish Service uses session management to implement authentication. This includes orphaned session timeouts and a number of simultaneous open sessions.

Step1. User can Post following username/password information in the payload field, which will create a new session.

```
{
  "UserName": "<username>",
  "Password": "<password>"
}
```

Example of applying for Authentication using a Chrome-based app (Advanced Rest Client): User will receive 201 message code with X-AUTH token created.

The screenshot displays the Advanced Rest Client interface. At the top, a POST request is configured to `https://BMC IP/redfish/v1/SessionService/Sessions/`. The request body is set to 'raw' and contains the following JSON payload:

```
1 {
2   "UserName": "<username>",
3   "Password": "<password>"
4 }
5
```

The response status is `201 Created`, with a response time of `792 ms` and a size of `470 B`. The response headers are visible, including:

- `Content-Length` → 239
- `Content-Type` → application/json
- `Date` → Fri, 14 Apr 2017 14:45:38 GMT
- `Location` → /redfish/v1/SessionService/Sessions/1
- `OData-Version` → 4.0
- `X-Auth-Token` → 9f0euw97fmimkved4lp2snxh042n7mqy

- Users can create maximum of 16 sessions.
- **Session Lifetime:** For Redfish sessions, as long as a client sends requests for the session within the session timeout period, the session will remain open and the session authentication token will remain valid. If the sessions times-out, the session will be automatically terminated.
- **According to Redfish spec, user can define session time from 30s to 86400s.**
If a user is not active in defined time frame then token will rendered invalid. Users can always patch "SessionTimeout" value if needed.
Example: [PATCH] <https://BMC IP/redfish/v1/SessionService> Payload: {"SessionTimeout": 50}
- **Session Termination or Logout:** A Redfish session is terminated when the client logs-out. This is accomplished by performing a DELETE to the session resource identified by the link returned in the location

header either when the session was created or if the Session ID is returned in the response data. The ability to DELETE a session by specifying the session resource ID allows an administrator with sufficient privilege to terminate other users sessions from a different session.

Example: [DELETE] [https://IP/redfish/v1/SessionService/Sessions/2\(num\)](https://IP/redfish/v1/SessionService/Sessions/2(num)) ->Send->Status Code: 200 OK

Log in	Log Out
Operation : POST	Operation: DELETE
URI: redfish/v1/SessionService/Sessions/	URI: redfish/v1/SessionService/Sessions/(num)
Request headers: Content-Type: application/json	Request headers: Content-Type: application/json
Request body: {"UserName":"UserName","Password":"Password"}	Requestbody: NONE
Response: 201 created	Response: 200 OK
X-Auth Token header displays Location and session ID ex: Location: /redfish/v1/SessionService/Sessions/5	

Step2. The response will include an X-Auth-token header with a session token and a location header. Parse X-Auth token value to get API response:

Note: User can apply basic authentication as well

3.2 Account Service

User can perform following operations under /redfish/v1/AccountService:

Supported operations: Get/Post/Patch/Delete

User can create new account using following API and payload. User can also delete respective accounts.

[POST] redfish/v1/AccountService/Accounts/

Payload:

```
{
  "UserName":"User_Name",
  "Password":"User_Password",
  "RoleId":"role_id",    *// Admin, Operator, ReadOnlyUser
  "Enabled":true
}
```

- User can also verify assigned privileges for different roles (ADMIN/Operator/Readonlyuser) under redfish/v1/AccountService/Roles

3.3 Event Service

The event service is a new alert mechanism for Redfish. This alert will be sent out through http to the web server that is subscribed to by the users.

First, user needs to add a subscription to inform Redfish who will receive this event.

After user adds subscriptions, he can execute "SendTestEvent" to send a testing event.

Alternatively, user can generate an event in the BMC and Redfish will automatically send an event alert to the destination(s) in the subscriptions. For this reason, you need to implement the event listener, which is like a web server that can receive https POST data that describes the Redfish event format.

For the current stage, user can launch Wireshark on the destination to sniff the packet to learn user receive the Redfish event.

Supported operations: Get/Post/Delete

To add a subscription:

[Post]: <https://IP/redfish/v1/EventService/Subscriptions/>
{ "Destination": "<http://www.dnsname.com/Destination1>", "Context": "user1_test", "EventTypes": ["Alert", "Status Change"], "Protocol": "Redfish" }

- User can subscribe to a max. Of events.

To see all subscriptions:

[GET]: <https://IP/redfish/v1/EventService/Subscriptions/>

To send a testing event:

[Post]: <https://IP/redfish/v1/EventService/Actions/EventService.SendTestEvent> {"EventType": "Alert"}

User can delete events using the Delete service.

[DELETE]: [https://IP/redfish/v1/EventService/Subscriptions/1 \(num\)](https://IP/redfish/v1/EventService/Subscriptions/1 (num))

3.4 Registries

</redfish/v1/Registries/Base.1.0.0>

Registry defines the base messages for Redfish. It represents properties for the registries themselves.

The Message Id is formed per the Redfish specification. It consists of the RegistryPrefix concatenated with the version concatenated with the unique identifier for the message registry entry.

3.5 Jschema

</redfish/v1/JsonSchemas>

The JSON Schema File resource describes the location (URI) of a particular Redfish schema definition being implemented or referenced by a Redfish service.

3.6 OEM APIs

3.6.1 SMTP

SMTP is implemented under <redfish/v1/Managers/1/SMTP>, Supported operations: Get/Patch

PATCH:

A: SMTP SSL authentication Disabled:

```
{
  "SmtpServer": "mailserver_ip or mailserver_name",
  "SmtpPortNumber": server_port,
  "SmtpUserName": "",
  "SmtpPassword": "",
  "SmtpSenderAddress": "sender_email_address"
}
```

B: SMTP SSL authentication Enabled:

```

{
  "SmtpSslEnabled": true,
  "SmtpServer": "mailserver_ip or mailserver_name",
  "SmtpPortNumber": server_port,
  "SmtpUserName": "user_name",
  "SmtpPassword": "user_password",
  "SmtpSenderAddress": "sender_email_address"
}

```

After applying the configurations, generate any system event to check if email-alert is received.

3.6.2 FanMode

It is implemented under /redfish/v1/Managers/1/FanMode

Allowable patch values: {"Standard", "FullSpeed", "PUE2", "HeavyIO"}

Example: Use the Patch operation and parse the following payload for your system.

```

{
  "Oem": {
    "OemFan": {
      "FanMode": "PUE2"} } }

```

3.6.3 Active Directory

AD is implemented under redfish/v1/Managers/1/ActiveDirectory

Method supported: Get/Patch/Post/Delete

- You can patch following properties in order to configure ActiveDirectory

```

{
  "@odata.context": "/redfish/v1/$metadata#ActiveDirectory.ActiveDirectory",
  "@odata.type": "#ActiveDirectory.ActiveDirectory",
  "@odata.id": "/redfish/v1/Managers/1/ActiveDirectory",
  "Id": "Active Directory",
  "Name": "Active Directory",
  "AuthenticationEnabled": false,
  "AuthenticationOverSslEnabled": false,
  "PortNumber": 389,
  "UserDomainName": "",
  "Timeout": 0,
  "DCSAddress1": "0.0.0.0",
  "DCSAddress2": "0.0.0.0",
  "DCSAddress3": "0.0.0.0",
  "RoleGroups": {
    "@odata.id": "/redfish/v1/Managers/1/ActiveDirectory/RoleGroups"
  }
}

```

- GET/POST: "redfish/v1/Managers/1/ActiveDirectory/RoleGroups"
You can perform post operation with following payload:
{"RoleGroupName":"xxx", "RoleGroupDomain":"xxx", "RoleGroupPrivilege":"Operator"}
- GET/PATCH/DELETE: "redfish/v1/Managers/1/ActiveDirectory/RoleGroups/ [number]"

3.6.4 Get/Set iKVM Mouse Mode

It is implemented under redfish/v1/Managers/1/MouseMode

Method supported: Get/Patch

Allowable values: "Absolute", "Relative", "Single"

3.6.5 Reset to Factory Default

It is implemented under `redfish/v1/Managers/1/Actions/Oem/ManagerConfig.Reset`

Method supported: Post

3.6.6 NTP

It is implemented under `redfish/v1/Managers/1/NTP`

Method supported: Get/Patch

Patch: "NTPEnable", "PrimaryNTPServer", "SecondaryNTPServer", "DaylightSavingTime"

3.6.7 RADIUS

It is implemented under `redfish/v1/Managers/1/RADIUS`

Method supported: Get/Patch

Patch: RadiusEnabled", "RadiusServerIP", "RadiusPortNumber", "RadiusSecret"

3.6.8 LDAP

It is implemented under `redfish/v1/Managers/1/LDAP`

Method supported: Get/Patch

Patch:

"LDAPEnabled", "LDAPAuthOverSSL", "LDAPPortNumber", "LDAPServerIP", "LDAPPassword", "LDAPDN", "LDAPSearchbase"

3.6.9 Snooping

[GET]: <https://x.x.x.x/redfish/v1/Managers/1/Snooping>

3.6.10 IP Access Control

It is implemented under `redfish/v1/Managers/1/IPAccessControl`

Method supported: Get/Patch/Post

<https://x.x.x.x/redfish/v1/Managers/1/IPAccessControl>

PATCH: {"ServiceEnabled": true}

<https://x.x.x.x/redfish/v1/Managers/1/IPAccessControl/FilterRule>

POST: {"Address": "10.136.176.0", "PrefixLength": 24, "Policy": "Accept"}

<https://x.x.x.x/redfish/v1/Managers/1/IPAccessControl/FilterRule/1>

PATCH: {"Address": "10.136.176.0", "PrefixLength": 24, "Policy": "Drop"}

3.6.11 SMCRAKP

It is implemented under `redfish/v1/Managers/1/SMCRAKP`

Method supported: Get/Patch

Example: PATCH - Raw data :{"Mode": "Enabled"}

3.6.12 SNMP

It is implemented under redfish/v1/Managers/1/SNMP, Method supported: Get/Patch

[Get]: <https://x.x.x.x/redfish/v1/Managers/1/SNMP>

[Patch]:

```
{"SnpEnabled":true}
```

```
{"SnpEnabled":false}
```

[Get]: <https://x.x.x.x/redfish/v1/Managers/1/SNMP/SNMPv2>

[Patch]: {"Snpv2Enabled":true,"ROCommunity":"rtest","RWCommunity":"wtest"}

[Get]: <https://x.x.x.x/redfish/v1/Managers/1/SNMP/SNMPv3>

[Patch]: {"Snpv3Enabled":true,"UserName":"administrator","AuthProtocol":"SHA1", "PrivateProtocol":"DES", "AuthKey":"Test1234", "PrivateKey":"Test1234"}

3.6.13 Syslog

It is implemented under redfish/v1/Managers/1/Syslog

Method supported: Get/Patch

Enable

[PATCH]: {"Enable Syslog": true,"Syslog PortNumber": 514,"Syslog ServerIP": "10.136.176.16"}

Disable

[PATCH]: {"Enable Syslog": false,"Syslog PortNumber": 514,"Syslog ServerIP": "10.136.176.16"}

3.6.14 Chassis Intrusion

It is implemented under redfish/v1/redfish/v1/Chassis/1

Method supported: Get/Patch

- Clear Chassis Intrusion- [PATCH]: {"PhysicalSecurity":{"IntrusionSensor": "Normal"}}

3.6.15 RAID Management Reference Examples

Create LSI3108 Volume	<p>URL: \${BMC_IP}/redfish/v1/Systems/1/Storage/HA-RAID/Actions/Oem/Storage.CreateVolume</p> <p>Method: post</p> <p>Example Body: { "ControllerId": 0, "Raid": "RAID0", "Span": 1, "PhysicalDrives": ["HA-RAID.0.Disk.0", "HA-RAID.0.Disk.1"], "UsePercentage": 100, "LogicalDriveCount": 1, "StripSizePerDDF": "256K", "LdReadPolicy": "NoReadAhead", "LdWritePolicy": "WriteBack", "LdIOPolicy": "DirectIO", "AccessPolicy": "ReadWrite", "DiskCachePolicy": "Unchanged", "InitState": "NoInit" }</p>
Locate physical Hdd	<p>URL: \${BMC_IP}/redfish/v1/Chassis/HA-RAID.[contoller_num].StorageEnclosure.[enclosure_num]/Drives/Disk.Bay.[disk_num]/Volume.Indicate</p> <p>Method: post</p> <p>Example Body: { "Active": "true" }</p>
Locate logical volume Hdd	<p>URL: \${BMC_IP}/redfish/v1/Systems/1/Storage/HA-RAID/HA-RAID.[contoller_num].Volumes/[volume_num]/Actions/OEM/Volume.Indicate</p> <p>Method: post</p> <p>Example Body: { "Active": "true" }</p>
Delete logical volume	<p>URL: \${BMC_IP}/redfish/v1/Systems/1/Storage/HA-RAID/HA-RAID.[contoller_num].Volumes/[volume_num]/Actions/OEM/Volume.Delete</p> <p>Method: post</p> <p>Example Body: { }</p>
Clear all logical volumes	<p>URL: \${BMC_IP}/redfish/v1/Systems/1/Storage/HA-RAID/Storage.ClearVolumes</p> <p>Method: post</p> <p>Example Body: { "ControllerId": 0 }</p>

Save HA-Raid controller config	<p>URL: \${BMC_IP}/redfish/v1/Systems/1/Storage/HA-RAID/Actions/Oem/HARAIDController.Save</p> <p>Method: patch</p> <p>Example Body: { "ControllerId": 0, "BIOSBootMode": "PauseOnError" }</p>
--------------------------------	---

3.6.16: IKVM

Description: Launch HTML5 iKVM using Redfish

1. [GET] URL: [\\${BMC_IP}/redfish/v1/Managers/1/IKVM](#)
2. Use the replied property, "URI", above to prepend "[https://\\${BMC_IP}](https://${BMC_IP})" and paste this complete URL in browser to render HTML5 iKVM

Example of launching URL: https://{BMC_IP}/redfish/Kk1D4UVATDja0Jw.IKVM

3.7 BIOS Configurations: Configure BIOS over Redfish

BIOS registry will show Menu of key (Menus), Keys (Attributes) and Keys' dependency (Dependencies)

[https://\\$BMC_IP/redfish/v1/Registries/BiosAttributeRegistry.v1_0_0](https://$BMC_IP/redfish/v1/Registries/BiosAttributeRegistry.v1_0_0)

```

1  | {
2  |   "@Redfish.Copyright": "Copyright 2016 Distributed Management Task Force, Inc. (DMTF). All rights reserved.",
3  |   "@odata.type": "#AttributeRegistry.v1_0_0.AttributeRegistry",
4  |   "Description": "This registry defines a representation of BIOS Attribute instances",
5  |   "Id": "BiosAttributeRegistry.v1_0_0",
6  |   "Language": "en",
7  |   "Name": "BIOS Attribute Registry",
8  |   "OwningEntity": "SMCI",
9  |   "RegistryVersion": "1.0.0",
10 |   "SupportedSystems": [
11 |     {
12 |       "ProdictName": "SuperMicroServer"
13 |     }
14 |   ],
15 |   "RegistryEntries": {
16 |     "Attributes": [{}],
6723 |     "Menus": [{}],
7117 |     "Dependencies": [{}]
8600 |   }
8601 | }

```

Attributes: containing the attributes and their possible values.

```

{
  "CurrentValue": "Force BIOS",
  "DisplayName": "Option ROM Messages",
  "HelpText": "Set display mode for Option ROM",
  "MenuPath": "./Advanced/BootFeature",
  "AttributeName": "OptionROMMessages",
  "IsFunCallback": false,
  "ReadOnly": false,
  "GrayOut": false,
  "Hidden": false,
  "Type": "Enumeration",
  "Value": [{
    "ValueDisplayName": "Force BIOS"
  },
  {
    "ValueDisplayName": "Keep Current"
  }]
},

```

Menu: containing the attributes menus and their hierarchy

```

{
  "DisplayName": "PCIe|PCI|PnP Configuration",
  "DisplayOrder": 26,
  "MenuPath": "./Advanced/PCIe|PCI|PnPConfiguration",
  "MenuName": "PCIe|PCI|PnPConfiguration",
  "Hidden": false,
  "ReadOnly": false
},

```

Dependencies: a list of dependencies of attributes on this component

```

{
  "Dependency": {
    "MapFrom": [{
      "MapFromAttribute": "PowerTechnology",
      "MapFromCondition": "NEQ",
      "MapFromProperty": "CurrentValue",
      "MapFromValue": "Custom",
      "MapTerms": "AND"
    },
    {
      "MapFromAttribute": "PowerPerformanceTuning",
      "MapFromCondition": "EQU",
      "MapFromProperty": "CurrentValue",
      "MapFromValue": "OS Controls EPB"
    }
  ],
  "MapToAttribute": "ENERGY_PERF_BIAS_CFGmode",
  "MapToProperty": "GrayOut",
  "MapToValue": true
},
  "DependencyFor": "ENERGY_PERF_BIAS_CFGmode",
  "Type": "Map"
},

```

Ex. If (PowerTechnology's CurrentValue != "Custom" **AND** PowerPerformanceTuning's CurrentValue == "OS Controls EPB") ENERGY_PERF_BIAS_CFGmode's GrayOut = true

Modify attributes:

[https://\\$BMC_IP/redfish/v1/Systems/1/Bios](https://$BMC_IP/redfish/v1/Systems/1/Bios)

→ User can GET current setting and PATCH/PUT desired settings

```

{
  @odata.context : "/redfish/v1/$metadata#Bios.Bios",
  @odata.type : "#Bios.v1_0_0.Bios",
  @odata.id : "/redfish/v1/Systems/1/Bios",
  Id : "Bios",
  Name : "BIOS Configuration Current Settings",
  AttributeRegistry : "BiosAttributeRegistry.v1_0_0",
  Description : "BIOS Configuration Current Settings",
  @Redfish.Settings : { @odata.type : "#Settings.v1_0_0.Settings", ETag : "SMC_TAG", Time : "Thu Feb 5 22:37:03 2015",...},
  Actios : { #Bios.ResetBios : { target : "/redfish/v1/Systems/1/Bios/Actions/Bios.ResetBios"...},
  Attributes : {
    A7Mode : "Enable",
    ACPI-States : "Enable",
    AES-NI : "Enable",
    AOMCPU1PCI-E3.0X160PROM : "Legacy",
    ASPMSupport : "Disabled",
    Above4GDecoding : "Disabled",
    AddOnROMDisplayMode : "Force BIOS",
    AddOnROMDisplayMode$2 : "Force BIOS",
    AdjacentCachePrefetch : "Enable",
    Azalia : "Auto",
    AzaliaPMEEnable : "Disabled",
  }
}

```

View pending settings:

[https://\\$BMC_IP/redfish/v1/Systems/1/Bios/SD](https://$BMC_IP/redfish/v1/Systems/1/Bios/SD)

→ User can view any pending setting after PATCH/PUT.

After PATCH/PUT, please reset system to set values to BIOS.

```

1  {
2    @odata.context : "/redfish/v1/$metadata#Bios.Bios",
3    @odata.type : "#Bios.v1_0_0.Bios",
4    @odata.id : "/redfish/v1/Systems/1/Bios/SD",
5    Id : "SD",
6    Name : "BIOS Configuration Pending Settings",
7    AttributeRegistry : "BiosAttributeRegistry.v1_0_0",
8    Description : "BIOS Configuration Pending Settings. These settings will be applied on next system reboot.",
9    Attributes : {
10     ASPMSupport : "Auto"
11   }
12 }

```

Factory default:

[https://\\$BMC_IP/redfish/v1/Systems/1/Bios/Actions/Bios.ResetBios](https://$BMC_IP/redfish/v1/Systems/1/Bios/Actions/Bios.ResetBios)

→ POST a reset of the BIOS attributes to default values

After POST, please reset system to set values to BIOS

Change BIOS booting Password:

[https://\\$BMC_IP/redfish/v1/Systems/1/Bios/Actions/Bios.ChangePassword](https://$BMC_IP/redfish/v1/Systems/1/Bios/Actions/Bios.ChangePassword)

→ POST with "PasswordName", "OldPassword", "NewPassword" to change password.

After POST, please reset system to set values to BIOS

4 UpdateService

4.1 Update SSL certificate and key

Description: Update SSL certificate and key for secure web server connection.

[POST] https://<BMC_IP>/redfish/v1/UpdateService/SSLCert/Actions/SSLCert.Upload

1. Change the type to “form-data”
2. Select cert_file and key_file as keys and browse respective files to upload-> send

4.2 BIOS Update

Description:

Update BIOS through Redfish API. In the current implementation, the content-type must be “multipart/form-data” while uploading the BIOS image.

4.2.1 Enter the BIOS update mode by posting the following request and expect to receive a “Successfully Completed Request” response.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.EnterBIOSUpdateMode>

Note: the following screenshots are from the Restlet Chrome based app.

The screenshot shows the Restlet Chrome interface for configuring a POST request. The URL is `https://10.138.160.98/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.EnterBIOSUpdateMode`. The headers section is expanded, showing `Authorization: Basic QURNSU46QURNSU4=` and `Content-Type: application/json`. The body section is expanded, showing a JSON object: `{}`. The body type is set to 'Text'.

4.2.2 Upload the BIOS image by posting the following request and expect to receive a “Successfully Completed Request” response. The content type must be “multipart/form-data”.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.UploadBIOS>

The screenshot shows the Restlet Chrome interface for configuring a POST request. The URL is `https://10.138.160.98/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.UploadBIOS`. The headers section is expanded, showing `Authorization: Basic QURNSU46QURNSU4=` and `Content-Type: multipart/form-data`. The body section is expanded, showing a form parameter `bbb [File] = X11SPI7.622`. The body type is set to 'Form'.

4.2.3 Update BIOS by posting the following request with the following payload and expect to receive a “Successfully Completed Request” response.

Payload: PreserveME, PreserveNVRAM and PreserveSMBIOS are required in the request body.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.UpdateBIOS>

REQUEST

METHOD: POST

URL: https://10.138.160.98/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem/FirmwareInventory.UpdateBIOS

HEADERS:

- Authorization: Basic QURNSU46QURNSU4=
- Content-Type: application/json

BODY:

```
1 {
2   "PreserveME": true,
3   "PreserveNVRAM": true,
4   "PreserveSMBIOS": true
5 }
```

4.2.4 Check the BIOS update status by issuing the following request with the GET method and expect to receive a response with the BIOS information.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BIOS/>

REQUEST

METHOD: GET

URL: https://10.138.160.98/redfish/v1/UpdateService/FirmwareInventory/BIOS/

HEADERS:

- Authorization: Basic QURNSU46QURNSU4=

RESPONSE

200 OK

HEADERS:

- Content-Length: 632 bytes
- OData-Version: 4.0
- Content-Type: application/json
- Date: 2017 Sep 30 18:08:29 -4d 20h

BODY:

```
{
  @odata.context: "/redfish/v1/$metadata#FirmwareInventory.FirmwareInventory",
  @odata.type: "#FirmwareInventory.v1_0_0.FirmwareInventory",
  @odata.id: "/redfish/v1/UpdateService/FirmwareInventory/BIOS",
  Id: "FirmwareInventory",
  Name: "SUPERMICRO BMC BIOS",
  Description: "Description of SUPERMICRO BMC BIOS",
  Updateable: true,
  VersionSequence: "BIOS Date: <Date> Rev <major>.<minor>",
  Version: "BIOS Date: 6/22/2017 Rev 1.0a",
  UpdateStatus: "Updated",
  Actions: {
    Oem: {
      #FirmwareInventory.EnterBIOSUpdateMode: {
        target: "/redfish/v1/UpdateService/FirmwareInventory/BIOS/Actions/Oem."
      }
    }
  },
  Oem: {}
}
```


4.3 BMC Firmware Update

Description:

Update BMC firmware through the Redfish API. In the current implementation, the content-type must be “multipart/form-data” while uploading the BMC image.

4.3.1 Enter BMC update mode by posting the following request and expect to receive a “Successfully Completed Request” response.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.EnterBMCUpdateMode>

The screenshot shows a REST client interface with the following details:

- REQUEST**
- METHOD:** POST
- URL:** https://10.138.160.98/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.EnterBMCUpdateMode
- QUERY PARAMETERS:** None
- HEADERS:**
 - Authorization: Basic QURNSU46QURNSU4=
 - Content-Type: application/json
- BODY:** A JSON object with three empty fields: { } (length: 4 bytes)

4.3.2 Upload the BMC image by issuing the following request with the POST method and expect to receive a “Successfully Completed Request” response. The content type must be “multipart/form-data”.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.UploadBMC>

The screenshot shows a REST client interface with the following details:

- REQUEST**
- METHOD:** POST
- URL:** https://10.138.160.98/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.UploadBMC
- QUERY PARAMETERS:** None
- HEADERS:**
 - Authorization: Basic QURNSU46QURNSU4=
 - Content-Type: multipart/form-data
- BODY:** A multipart form-data body with a file parameter named bbbCCC (type: application/octet-stream, size: 3355443 bytes)

4.3.3 Update the BMC by posting the following request with the following payload and expect to receive a “Successfully Completed Request” response.

Payload: PreserveCfg, PreserveSdr, PreserveSsl and UpdateBootLdr are required in the request body.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.UpdateBMC>

REQUEST

METHOD: POST | SCHEME://HOST[:PORT]/[PATH][?QUERY] | length: 108 bytes

URL: https://10.138.160.98/redfish/v1/UpdateService/FirmwareInventory/BMC/Actions/Oem/FirmwareInventory.UpdateBMC

HEADERS:

- Authorization: Basic QURNSU46QURNSU4=
- Content-Type: application/json

BODY:

```

1 {
2   "PreserveCfg": true,
3   "PreserveSdr": true,
4   "PreserveSsl": true,
5   "UpdateBootLdr": true
6 }

```

4.3.4 Check the BMC firmware information by issuing the following request with the GET method and expect to receive a response with BMC information.

<https://<IP>/redfish/v1/UpdateService/FirmwareInventory/BMC/>

5 Examples

Users can integrate current APIs into their software and applications in order to receive all services provided by Redfish APIs.

5.1 Posting an action:

POST | https://BMC IP/redfish/v1/Systems/1/Actions/ComputerSystem.Reset | Params

Authorization | Headers (1) | Body | Pre-request Script | Tests

form-data | x-www-form-urlencoded | raw | binary | Text

Body | Cookies | Headers (6) | Test Results | Status: 200 OK

Request Body:

```

1 {
2   "ResetType": "On"
3 }

```

Response Body:

```

1 {
2   "Success": {
3     "code": "Base.v1_0_0.Success",
4     "Message": "Successfully Completed Request."
5   }
6 }

```

5.2 Getting mac address from AOC

The screenshot shows a REST client interface with a GET request to `https://BMC IP/redfish/v1/Systems/1/EthernetInterfaces/5`. The request headers include an Authorization token. The response body is displayed in JSON format, showing details for the AOC LAN 1 interface, including its name, description, status (Disabled), health (OK), and MAC address (00:25:90:5e:6d:a0).

Key	Value
Authorization	Basic QURNSU46QURNSU4=
New key	Value

```
1 {
2   "@odata.context": "/redfish/v1/$metadata#EthernetInterface.EthernetInterface",
3   "@odata.type": "#EthernetInterface.v1_0_0.EthernetInterface",
4   "@odata.id": "/redfish/v1/Systems/1/EthernetInterfaces/5",
5   "Id": "5",
6   "Name": "AOC LAN 1",
7   "Description": "AOC-S25G-i2S #1",
8   "Status": {
9     "State": "Disabled",
10    "Health": "OK"
11  },
12  "MACAddress": "00:25:90:5e:6d:a0",
13  "SpeedMbps": 0,
```

5.3 Memory info through Redfish API:

```
1 {
2   "@odata.context": "/redfish/v1/$metadata#Memory.Memory",
3   "@odata.type": "#Memory.v1_1_0.Memory",
4   "@odata.id": "/redfish/v1/Systems/1/Memory/5",
5   "Id": "5",
6   "Name": "Memory",
7   "RankCount": 5,
8   "Description": "Memory",
9   "CapacityMiB": 16384,
10  "DataWidthBits": 64,
11  "BusWidthBits": 72,
12  "MemoryMedia": [
13    "DRAM"
14  ],
15  "MemoryType": "DRAM",
16  "MemoryDeviceType": "DDR4",
17  "OperatingSpeedMhz": 2133,
18  "DeviceLocator": "P2-DIMMB1",
19  "MemoryLocation": {
20    "Socket": 1,
21    "MemoryController": 0,
22    "Channel": 1,
23    "Slot": 0
24  },
25  "Manufacturer": "Micron Technology",
26  "SerialNumber": "101A73C1",
27  "PartNumber": "36ASF2G72PZ-2G1A2",
28  "Status": {
29    "State": "Enabled",
30    "Health": "OK"
31  }
32 }
```

5.4 Redfish API Response for drive connected to 3108 controller

```
GET https://BMC IP/redfish/v1/Chassis/HA-RAID.0.StorageEnclosure.0/Drives/Disk.Bay.1

Pretty Raw Preview JSON
```

```
1 {
2   "@odata.context": "/redfish/v1/$metadata#Drive.Drive",
3   "@odata.type": "#Drive.v1_1_0.Drive",
4   "@odata.id": "/redfish/v1/Chassis/HA-RAID.0.StorageEnclosure.0/Drives/Disk.Bay.1",
5   "Name": "Disk.Bay.1",
6   "Id": "1",
7   "Manufacturer": "ATA",
8   "SerialNumber": "WD-WX11E83C3344",
9   "Model": "WDC WD5000BHTZ-0",
10  "Revision": "6A00",
11  "StatusIndicator": "OK",
12  "FailurePredicted": false,
13  "MediaType": "HDD",
14  "CapacityBytes": 499558383616,
15  "BlockSizeBytes": 512,
16  "CapableSpeedGbs": 6,
17  "IndicatorLED": "Off",
18  "Status": {
19    "State": "Enabled",
20    "Health": "OK"
21  },
22  "Links": {
23    "Volumes": [
24      {
25        "@odata.id": "/redfish/v1/Systems/1/Storage/HA-RAID/Volumes/Controller.0.Volume.0"
26      }
27    ]
28  },
29  "Actions": {
30    "Oem": {
31      "#Drive.Indicate": {
32        "target": "/redfish/v1/Chassis/HA-RAID.0.StorageEnclosure.0/Drives/Disk.Bay.1/Actions/Oem/Drive.Indicate",
33        "Active@Redfish.AllowableValues": [
34          "true",
35          "false"
36        ]
37      }
38    }
39  }
40 }
```

5.5 Python Code for Redfish API Response

```
base_url = 'https://IP/redfish/v1/Managers/1/SerialInterfaces/1'  
dict_host = requests.get(base_url).json()  
print (json.dumps(dict_host, indent=2))
```

Output:

```
{  
  "@odata.type": "#SerialInterface.1.0.0.SerialInterface",  
  "Parity": "None",  
  "Name": "SerialInterfaces",  
  "DataBits": "8",  
  "@odata.id": "/redfish/v1/Managers/1/SerialInterfaces/1",  
  "@odata.context":  
  "/redfish/v1/Managers/1/SerialInterfaces/1/$metadata#Managers/Links/Members/1/Links/SerialInterfaces/$entity",  
  "FlowControl": "None",  
  "SignalType": "Rs232",  
  "StopBits": "8",
```

6 Reference Links

Supermicro Redfish:

<https://www.supermicro.com/solutions/Redfish.cfm>

Supermicro YouTube:

<https://www.youtube.com/watch?v=anppU663kUs>

DMTF Redfish:

<http://www.dmtf.org/standards/redfish>

<http://redfish.dmtf.org/>

Mockups:

<http://redfish.dmtf.org/redfish/v1>

Contact:

Supermicro Technical Support