Intel® Rack Scale Design Pod Manager (PODM)

API Specification
Software Version 2.2

December 12, 2017

Revision 001

Document Number: 336857-001

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and noninfringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services, and/or processes in development .All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications, and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents that have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting *http://www.intel.com/design/literature.htm*.

Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others

Copyright © 2017 Intel Corporation. All rights reserved.

Contents

December 13, 2017
Document Number: 336857-001

Figures

Tables

Revision History

Revision	Description	Date
001	Initial Release	December 13, 2017

§

1 Introduction

1.1 Scope

This document contains information about the Intel® Rack Scale Design Pod Manager (PODM) REST API, which has been designed and implemented for the Intel® Rack Scale Design Software v2.2 release.

The interfaces specified in this document are based on the *Distributed Management Task Force's Redfish* Interface Specification,* and *Schema*, DSP8010 v2016.3. (Refer to *dmtf.org*)

1.2 Intended audience

The intended audience for this document is designers and engineers working with the Rack Scale Design Software 2.2 release.

1.3 Terminology

Table 1 Terminology

Term	Definition
BMC	Baseboard Management Controller
CIMI	Cloud Infrastructure Management Interface
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
NIC	Network Interface Card
OCCI	Open Cloud Computing Interface
OData	Open Data Protocol
OVF	Open Virtualization Format
POD	A physical collection of multiple racks
PODM	POD Manager
PSME	Pooled System Management Engine
REST	Representational state transfer
Intel® TXT	Intel® Trusted Execution Technology
URI	Uniform resource identifier
UUID	Universally Unique Identifier
XML	Extensible Markup Language

1.4 References

Table 2 Reference Documents

Doc ID	Title	Location
336811	Intel® Rack Scale Design (RSD) Conformance and Software Reference Kit Getting Started Guide v2.2, Revision 001	*http://www.intel.com/intelRSD*
336814	Intel® Rack Scale Design Pod Manager (PODM) Release Notes, Software v2.2, Revision 001	
336815	Intel® Rack Scale Design Pod Manager (PODM) User Guide, Software v2.2, Revision 001	
336816	Intel® Rack Scale Design PSME Release Notes, Software v2.2, Revision 001	
336810	Intel® Rack Scale Design PSME User Guide, Software v2.2, Revision 001	
336855	Intel® Rack Scale Design PSME REST API Specification, Software v2.2, Revision 001	
336856	Intel® Rack Scale Design Storage Services API Specification, Software v2.2, Revision 001	
336858	Intel® Rack Scale Design Rack Management Module (RMM) API Specification, Software v2.2, Revision 001	
336859	Intel® Rack Scale Design Generic Assets Management Interface API Specification, Software v2.2, Revision 001	
336860	Intel® Rack Scale Design Firmware Extension Specification, Software v2.2, Revision 001	
336861	Intel® Rack Scale Design Architecture Specification, Software v2.2, Revision 001	
336862	Intel® RSD v2.2 Solid State Drive (SSD) Technical Advisory	
RFC2119	Key words for use in RFCs to Indicate Requirement Levels, March 1997	*https://www.ietf.org/rfc/rfc2119.txt*
SDP0266	Scalable Platforms Management API Specification v1.1.0	*https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.1.0.pdf*
DSP8010	Redfish Schema v2016.3	*https://www.dmtf.org/sites/default/files/standards/documents/DSP8010_2016.3.zip*

1.5 Notes and Symbol Convention

Symbol and note convention are similar to typographical conventions used in CIMI specification.

Notation used in JSON serialization description:

- Mandatory in italics indicate data types instead of literal Mandatory.
- Characters are appended to items to indicate cardinality:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
- Vertical bars, "|", denote choice. For example, "a|b" means a choice between "a" and "b".
- Parentheses, "(" and ")", are used to indicate the scope of the operators "?", "*", "+" and "|".
- Ellipses (i.e., "...") indicate points of extensibility.

Note: The lack of ellipses does not mean no extensibility point exists; rather it is just not explicitly called out.

§

2 PODM API

2.1 PODM API structure and relations

The PODM REST API provides the REST-based interface that allows full management of the Intel® Rack Scale Design POD including asset discovery, configuration, and composed node assembly.

2.1.1 PODM API physical resource hierarchy

Figure 1. PODM REST API Hierarchy

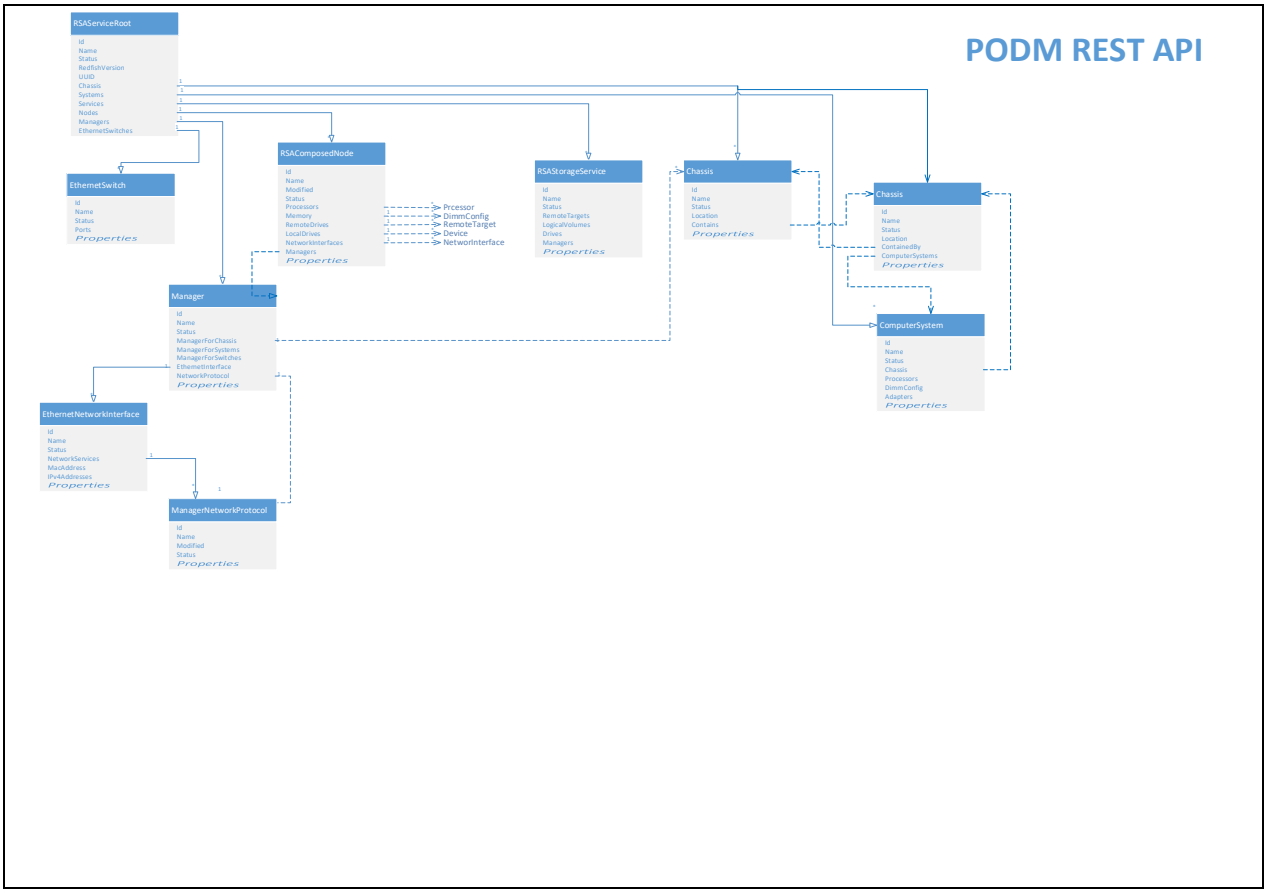


Table 3. Resources and URIs

Resource	URI
Service Root	/redfish/v1
Chassis Collection	/redfish/v1/Chassis
Chassis	/redfish/v1/Chassis/{chassisID}
Computer System Collection	/redfish/v1/Systems
Computer System	/redfish/v1/Systems/{systemID}
Processors Collection	/redfish/v1/Systems/{systemID}/Processors
Processor	/redfish/v1/Systems/{systemID /Processors/{processorID}
Memory Collection	/redfish/v1/Systems/{systemID}/Memory
Memory	/redfish/v1/Systems/{systemID}/Memory/{memoryID}

Resource	URI
Manager Collection	/redfish/v1/Managers
Manager	/redfish/v1/Managers/{managerID}
Network Protocol	/redfish/v1/Managers/{managerID}/NetworkProtocol
Network Interface Collection	/redfish/v1/Systems/{systemID}/EthernetInterfaces /redfish/v1/Managers/{managerID}/EthernetInterfaces
Network Interface	/redfish/v1/Systems/{systemID}/EthernetInterfaces/{nicID} /redfish/v1/Managers/{managerID}/EthernetInterfaces/{nicID}
Ethernet Switch Collection	/redfish/v1/EthernetSwitches
Ethernet Switch	/redfish/v1/EthernetSwitches/{switchID}
Ethernet Switch Port Collection	/redfish/v1/EthernetSwitches/{switchID}/Ports
Ethernet Switch Port	/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}
VLAN Network Interface Collection	/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}/VLANs /redfish/v1/Systems/{systemID}/EthernetInterfaces/{nicID}/VLANs /redfish/v1/Managers/{managerID}/EthernetInterfaces/{nicID}/VLANs
VLAN Network Interface	/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}/VLANs/{vlanID} /redfish/v1/Systems/{systemID}/EthernetInterfaces/{nicID}/VLANs/{vlanID} /redfish/v1/Managers/{managerID}/EthernetInterfaces/{nicID}/VLANs/{vlanID}
Storage Service Collection	/redfish/v1/Services
Storage Service	/redfish/v1/Services/{serviceID}
Remote Target Collection	/redfish/v1/Services/{serviceID}/Targets
Remote Target	/redfish/v1/Services/{serviceID}/Targets/{targetID}
Logical Drive Collection	/redfish/v1/Services/{serviceID}/LogicalDrives
Logical Drive	/redfish/v1/Services/{serviceID}/LogicalDrives/{driveID}
Physical Drive Collection	/redfish/v1/Services/{serviceID}/Drives
Physical Drive	/redfish/v1/Services/{serviceID}/Drives/{driveID}
Composed Node Collection	/redfish/v1/Nodes
Composed Node	/redfish/v1/Nodes/{nodeID}
Simple Storage Collection	/redfish/v1/Systems/{system1}/SimpleStorage
SimpleStorage	/redfish/v1/Systems/{system1}/SimpleStorage/{storageID}
PowerZone Collection	/redfish/v1/Chassis/{chassisID}/PowerZones
PowerZone	/redfish/v1/Chassis/{chassisID}/PowerZones/{powerzoneID}
ThermalZone Collection	/redfish/v1/Chassis/{chassisID}/ThermalZones
ThermalZone	/redfish/v1/Chassis/{chassisID}/ThermalZones/{thermalzoneID}
Power	/redfish/v1/Chassis/{chassisID}/Power
Thermal	/redfish/v1/Chassis/{chassisID}/Thermal
Storage Subsystem Collection	/redfish/v1/Systems/{systemID}/Storage
Storage Subsystem	/redfish/v1/Systems/{systemID}/Storage/{storageID}
Drives	/redfish/v1/Chassis/{chassisID}/Drives/{driveID}
Fabrics collection	/redfish/v1/Fabrics
Fabric	/redfish/v1/Fabrics/{fabricID}
Fabric Switch collection	/redfish/v1/Fabrics/{fabricID}/Switches
Fabric Switch	/redfish/v1/Fabrics/{fabricID}/Switches/{switchID}

Resource	URI
Fabric Switch Port collection	/redfish/v1/Fabrics/{fabricID}/Switches/{switchID}/Ports
Fabric Switch Port	/redfish/v1/Fabrics/{fabricID}/Switches/{switchID}/Ports/{portID}
Fabric Zone collection	/redfish/v1/Fabrics/{fabricID}/Zones
Fabric Zone	/redfish/v1/Fabrics/{fabricID}/Zones/{zoneID}
Endpoint Collection	/redfish/v1/Fabrics/{fabricID}/Endpoints
Endpoint	/redfish/v1/Fabrics/{fabricID}/Endpoints/{endpointID}
PCIeDevice	/redfish/v1/Chassis/{chassisID}/PCIeDevices/{deviceID}
PCIe Device Function	/redfish/v1/Chassis/{chassisID}/PCIeDevices/{deviceID}/Functions/{functionID}

§

3 PODM REST API Error Codes

This chapter describes all error codes that may be returned by the REST calls implemented in the PODM REST API of the Intel® Rack Scale Design software v2.2 release.

3.1 API error response

In the case of an error, the PODM REST API responds with an HTTP status code, as defined by the HTTP 1.1 specification and constrained by additional requirements defined in this specification.

HTTP response status codes alone often do not provide enough information to determine the error cause. The PODM REST API returns extended error information as a JSON object with a single property named “error”. The value of this property is the JSON object with the properties listed in Table 4.

Table 4. API Error Response Attributes

Attribute	Description
`MessageId`	String indicating a specific error or message (not to be confused with the HTTP status code). This code can be used to access a detailed message from a message registry.
`Message`	A human readable error message indicating the semantics associated with the error. This is the complete message, and not rely on substitution variables.
`MessageArgs`	An optional array of strings representing the substitution parameter values for the message. This is included in the response if a `MessageId` is specified for a parameterized message.
`Severity`	An optional string representing the severity of the error.
`Resolution`	An optional string describing recommended action(s) to take to resolve the error.
`RelatedProperties`	An optional array of JSON Pointers defining the specific properties within a JSON payload described by the message.

3.1.1 Example error JSON object

```
{
    "error": {
        "code": "Base.1.0.GeneralError",
        "message": "A general error has occurred. See ExtendedInfo for more
information.",
        "@Message.ExtendedInfo": [
            {
                "@odata.type" :
"/redfish/v1/$metadata#Message.v1_0_0.Message",
                "MessageId": "Base.1.0. MalformedJSON",
                "Message": "The request body submitted was malformed JSON and
could not be parsed by the receiving service",
                "Severity": "Error"
            }
            {
                "@odata.type" :
"/redfish/v1/$metadata#Message.v1_0_0.Message",
                "MessageId": "Base.1.0.PropertyNotWriteable",
                "RelatedProperties": [
                    "#/Name"
                ],
```

```
                "Message": "The property Name is a read only property and
cannot be assigned a value",
                "MessageArgs": [
                    "Name"
                ],
                "Severity": "Warning",
                "Resolution": "Remove the property from the request body and
resubmit the request if the operation failed"
            }
        ]
    }
}
```

3.2 API error codes

In general, if an error is not described in any of the following tables, it is to be mapped into HTTP 500 Internal Error code.

3.2.1 General error codes

For a detailed list of error codes, refer to the Redfish* specification, Section 6.5.2.

The client should be prepared to handle the error codes listed in Table 5.

Table 5 HTTP error status codes

HTTP Status Code	Description
400 Bad Request	The request could not be processed because it contains missing or invalid information (such as validation error on an input field, a missing required value, and so on). An extended error will be returned in the response body.
404 Not Found	The request specified a URI of a resource that does not exist.
405 Method Not Allowed	The HTTP verb specified in the request (e.g., DELETE, GET, HEAD, POST, PUT, PATCH) is not supported for this request URI. The response includes an Allow header which provides a list of methods that are supported by the resource identified by the Request-URI.
409 Conflict	A creation or update request cannot be applied given the state of the resource.
500 Internal Server Error	The server encountered an unexpected condition that prevented it from fulfilling the request. An extended error is returned in the response body.
501 Not Implemented	The server does not (currently) support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any resource.
503 Service Unavailable	The server is currently unable to handle the request due to temporary overloading or maintenance of the server.

3.2.2 PATCH method error codes

For the PATCH method, the Rack Scale Design service will conform to IETF RFC 5789.

The service will respond with following error codes in cases listed below:

- 400 Bad Request – Malformed JSON in request (values not in range, unknown property, etc.)
- 405 Method Not Allowed – Resource doesn't support PATCH method
- 409 Conflict – Update can't be executed at this moment. User might be able to resolve the conflict and resubmit the request.

- 501 Not Implemented – Resource supports PATCH method, but current implementation doesn't support such functionality)
- 500 Internal Server Error – All other situations where any of above codes do not fit (e.g., underlying HW doesn't allow to execute this particular request).

3.2.3 POST method error codes

The POST method is used to create a new resource (the POST request is submitted to the resource collection in which the new resource is to belong) or to initiate operation on the object (sending the POST method to the URI of the action).

The service will respond with following error codes in the cases listed below:

- 400 Bad Request – Malformed JSON in request (values not in range, unknown property, etc.)
- 405 Method Not Allowed – Resource doesn't support the POST method
- 409 Conflict – Update can't be executed at this moment. User might be able to resolve the conflict and resubmit the request.
- 501 Not Implemented – Resource supports the POST method, but the current implementation doesn't (e.g., underlying HW doesn't support such functionality)
- 500 Internal Server Error – All other situations where any of the above codes do not fit (e.g., underlying HW doesn't allow executing this particular request). Extended error info should provide information as to why the operation failed.

§

4 PODM REST API Definition

4.1 Odata support

Intel® Rack Scale Design supports Odata v4.0 as it is defined in the Redfish specification refered to in Table 2, Reference Documents.

All resources within this RESTful API are identified by a unique identifier property named “@odata.id”. Resource Identifiers are represented in JSON payloads as URI paths relative to the Redfish Schema portion of the URI. That is, they always start with "/redfish/". The resource identifier is the canonical URL for the resource and can be used to retrieve or edit the resource, as appropriate..

4.2 Protocol version

The protocol version is separate from the version of the resources or the version of the Redfish Schema supported by them.

Each version of the Redfish protocol is strongly typed. This is accomplished using the URI of the Redfish service in combination with the resource obtained at that URI, called the `ServiceRoot`.

The root URI for this version of the Redfish protocol is "/redfish/v1/".

While the major version of the protocol is represented in the URI, the major version, minor version and errata version of the protocol are represented in the Version property of the `ServiceRoot` resource, as defined in the Redfish Schema for that resource. The protocol version is a string of the form:

```
MajorVersion.MinorVersion.Errata
```

Where:

- MajorVersion = integer: something in the class changed in a backward incompatible way.
- MinorVersion = integer: a minor update. New functionality may have been added but nothing removed. Compatibility will be preserved with previous minorversions.
- Errata = integer: something in the prior version was broken and needed to be fixed.

Any resource discovered through links found by accessing the root service, or any service or resource referenced using references from the root service, will conform to the same version of the protocol supported by the root service.

4.2.1 Operations

4.2.1.1 GET

Request:

```
GET /redfish
Content-Type: application/json
```

Response:

```
{
       "v1": "/redfish/v1/"
}
```

4.3 Odata service document

This service document provides a standard format for enumerating the resources exposed by the service, enabling generic hypermedia-driven OData clients to navigate to the resources of the service.

4.3.1 Operations

4.3.1.1 GET

Request:

```
GET /redfish/v1/odata
Content-Type: application/json
```

Response:


```
{
    "@odata.context": "/redfish/v1/$metadata",
    "value": [
        {
            "name": "Service",
            "kind": "Singleton",
            "url": "/redfish/v1/"
        },
        {
            "name": "Systems",
            "kind": "Singleton",
            "url": "/redfish/v1/Systems"
        },
        {
            "name": "Chassis",
            "kind": "Singleton",
            "url": "/redfish/v1/Chassis"
        },
        {
            "name": "Managers",
            "kind": "Singleton",
            "url": "/redfish/v1/Managers"
        },
        {
            "name": "Nodes",
            "kind": "Singleton",
            "url": "/redfish/v1/Nodes"
        },
        {
            "name": "Services",
            "kind": "Singleton",
            "url": "/redfish/v1/Services"
        },
        {
            "name": "EthernetSwitches",
            "kind": "Singleton",
            "url": "/redfish/v1/EthernetSwitches"
        },
        {
            "name": "EventService",
            "kind": "Singleton",
            "url": "/redfish/v1/EventService"
        },
        {
            "name": "TelemetryService",
            "kind": "Singleton",
```

```
            "url": "/redfish/v1/TelemetryService"
        },
        {
            "name": "Fabrics",
            "kind": "Singleton",
            "url": "/redfish/v1/Fabrics"
        }
    ]
}
```

4.4 Intel® Rack Scale Design POD manager service root

Intel® Rack Scale Design POD Manager Service Root resource – entry point. Properties details are available in the `ServiceRoot.xml` metadata file.

4.4.1 Operations

4.4.1.1 GET

Request:

```
GET /redfish/v1
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#ServiceRoot.ServiceRoot",
    "@odata.id": "/redfish/v1/",
    "@odata.type": "#ServiceRoot.v1_1_1.ServiceRoot",
    "Id": "RootService",
    "Name": "Root Service",
    "Description": "description-as-string",
    "RedfishVersion": "1.1.0",
    "UUID": "92384634-2938-2342-8820-489239905423",
    "Systems": {
        "@odata.id": "/redfish/v1/Systems"
    },
    "Chassis": {
        "@odata.id": "/redfish/v1/Chassis"
    },
    "Managers": {
        "@odata.id": "/redfish/v1/Managers"
    },
    "EventService": {
        "@odata.id": "/redfish/v1/EventService"
    },
    "Fabrics": {
      "@odata.id": "/redfish/v1/Fabrics"
    },
    "TelemetryService": {
      "@odata.id": "/redfish/v1/TelemetryService"
    },
    "Oem": {
        "Intel_RackScale": {
            "@odata.type": "#Intel.Oem.ServiceRoot",
            "ApiVersion": "2.2.0",
```

```
            "Services": {
                "@odata.id": "/redfish/v1/Services"
            },
            "Nodes": {
                "@odata.id": "/redfish/v1/Nodes"
            },
            "EthernetSwitches": {
                "@odata.id": "/redfish/v1/EthernetSwitches"
            }
        }
    },
   "Links": {}
}
```

4.4.1.2 PUT

Operation is not allowed on this resource.

4.4.1.3 PATCH

Operation is not allowed on this resource.

4.4.1.4 POST

Operation is not allowed on this resource.

4.4.1.5 DELETE

Operation is not allowed on this resource.

4.5 Chassis collection

Table 6 Chassis collection attributes

Name	Chassis		
Type URI	/redfish/v1/Chassis		
Attribute	**Type**	**Required**	**Description**
Name	String	Yes	Name of collection
Members@odata.count	Number	No	Collection members count
Members	Array	No	Contains the members of this collection.

4.5.1 Operations

4.5.1.1 GET

Request:

```
GET /redfish/v1/Chassis
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#Chassis",
    "@odata.id": "/redfish/v1/Chassis",
    "@odata.type": "#ChassisCollection.ChassisCollection",
    "Name": "Chassis Collection",
    "Members@odata.count": 6,
    "Members": [
```

```
        {
            "@odata.id": "/redfish/v1/Chassis/Pod"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/Drawer1"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/FabricModule1"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/Sled1"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/Blade1"
        }
    ]
}
```

4.5.1.2 PUT

Operation is not allowed on this resource.

4.5.1.3 PATCH

Operation is not allowed on this resource.

4.5.1.4 POST

Operation is not allowed on this resource.

4.5.1.5 DELETE

Operation is not allowed on this resource.

4.6 Chassis

This is the schema definition for the Chassis resource. It represents the properties of physical components for any system. This one object is intended to represent racks, rackmount servers, blades, standalone, modular systems, enclosures, and all other containers. The non-CPU/device-centric parts of the schema are all accessed either directly or indirectly through this resource.

Details of this resource are described in the metadata file, *Chassis.xml*. OEM extension details are available in `IntelRackScaleOem.xml`.

4.6.1 Operations

4.6.1.1 GET

Request:

```
GET /redfish/v1/Chassis/1
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#Chassis/Members/$entity",
```

```
    "@odata.id": "/redfish/v1/Chassis/Rack1",
    "@odata.type": "#Chassis.v1_3_0.Chassis",
    "Id": "Rack1",
    "ChassisType": "RackMount",
    "Name": "name-as-string",
    "Description": "description-as-string",
    "Manufacturer": "Intel Corporation",
    "Model": "model-as-string",
    "SKU": "sku-as-string",
    "SerialNumber": "serial-number-as-string",
    "PartNumber": "part-number-as-string",
    "AssetTag": null,
    "IndicatorLED": "Unknown",
    "Status": {
        "State": "Enabled",
        "Health": "OK",
        "HealthRollup": null
    },
    "Oem": {
        "Intel_RackScale": {
            "@odata.type": "#Intel.Oem.RackChassis",
            "Location": {
                "Id": "Rack1",
                "ParentId": "Pod1"
            },
            "RMMPresent": true,
            "RackSupportsDisaggregatedPowerCooling": true,
            "UUID": "Unique ID",
            "GeoTag": "54.348103, 18.645172"
        }
    },
    "ThermalZones": {
        "@odata.id": "/redfish/v1/Chassis/Rack1/ThermalZones"
    },
    "PowerZones": {
        "@odata.id": "/redfish/v1/Chassis/Rack1/PowerZones"
    },
    "Thermal": {
      "@odata.id": "/redfish/v1/Chassis/Rack1/Thermal"
    },
    "Power": {
      "@odata.id": "/redfish/v1/Chassis/Rack1/Power"
    },
    "Links": {
        "@odata.type": "#Chassis.v1_2_0.Links",
        "Contains": [
            {
                "@odata.id": "/redfish/v1/Chassis/Drawer1"
            }
        ],
        "ContainedBy": null,
        "ComputerSystems": [],
        "ManagedBy": [
            {
                "@odata.id": "/redfish/v1/Managers/RMM"
```

```
            }
        ],
        "ManagersInChassis": [
            {
                "@odata.id": "/redfish/v1/Managers/RMM"
            }
        ],
        "Storage": [
        ],
        "Drives": [
        ],
        "Oem": {
          "Intel_RackScale": {
            "@odata.type": "#Intel.Oem.ChassisLinks",
            "Switches": []
          }
       }
    }
}
```

4.6.1.2 PUT

Operation is not allowed on this resource.

4.6.1.3 PATCH

The following properties can be updated by the PATCH operation:

Table 7. Chassis Properties

Attribute	Type	Required	Description
`AssetTag`	String	No	The user assigned asset tag for this chassis.
`Oem->Intel_RackScale->Location->Id`	String	No	The user assigned location id for this chassis. It can be changed only for Rack Chassis. Support for changing this attribute is not mandatory in PODM 2.2 API Specification but if implemented it needs to be aligned to this specification.

Request:

```
PATCH /redfish/v1/Chassis/Rack1
Content-Type: application/json
{
    "AssetTag": "Rack#1",
    "Oem": {
        "Intel_RackScale": {
            "Location": {
                "Id": "1234",
            },
        }
    },
}
```

Response:

```
HTTP/1.1 204 No Content
```

or

```
HTTP/1.1 200 OK
```

With full resource representation.

4.6.1.4 POST

Operation is not allowed on this resource.

4.6.1.5 DELETE

Operation is not allowed on this resource.

4.7 PowerZone collection

PowerZone collection resource.

Table 8. Powerzone Collection Attributes

Name	PowerZone Collection	
Type URI	/redfish/v1/Chassis/{chassisID}/PowerZones	
Attribute	**Type**	**Description**
Name	String	Name of collection
Members@odata.count	Number	Collection members count
Members	Array	Contains the members of this collection.

4.7.1 Operations

4.7.1.1 GET

Request:

```
GET /redfish/v1/Chassis/Rack1/PowerZones
Content-Type: application/json
```

Response:

```
{
    "@odata.context":
"/redfish/v1/$metadata#PowerZoneCollection.PowerZoneCollection",
    "@odata.id": "/redfish/v1/Chassis/Rack1/PowerZones",
    "@odata.type": "#PowerZoneCollection.PowerZoneCollection",
    "Name": "Power Zones Collection",
    "Members@odata.count": 1,
    "Members": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/PowerZones/Power1"
        }
    ]
}
```

4.7.1.2 PUT

Operation is not allowed on this resource.

4.7.1.3 PATCH

Operation is not allowed on this resource.

4.7.1.4 POST

Operation is not allowed on this resource.

4.7.1.5 DELETE

Operation is not allowed on this resource.

4.8 PowerZone

This resource is used to represent a power zone resource for an Intel® Rack Scale Design implementation. It contains Power Supplies and Power Control information.

Details of this resource are described in the metadata file, `PowerZone.xml`.

4.8.1 Operations

4.8.1.1 GET

Request:

```
GET /redfish/v1/Chassis/Rack1/PowerZones/Power1
Content-Type: application/json
```

Response:

```
{
    "@odata.context":
"/redfish/v1/$metadata#Chassis/Rack/PowerZones/Members/$entity",
    "@odata.id": "/redfish/v1/Chassis/Rack1/PowerZones/1",
    "@odata.type": "PowerZone.v1_0_0.PowerZone",
    "Id": "1",
    "Name": "power zone 1",
    "Description": "power zone 1",
    "Status": {
        "State": "Enabled",
        "Health": "OK",
        "HealthRollup": "OK"
    },
    "RackLocation": {
        "RackUnits": "OU",
        "XLocation": 0,
        "ULocation": 1,
        "UHeight": 8
    },
    "MaxPSUsSupported": 6,
    "Presence": "111111",
    "NumberOfPSUsPresent": 6,
    "PowerConsumedWatts": 2000,
    "PowerOutputWatts": 2000,
    "PowerCapacityWatts": 3000,
    "PowerSupplies": [
        {
            "Name": "Power supply 1",
            "Status": {
```

```
                "State": "Enabled",
                "Health": "OK",
                "HealthRollup": "OK"
            },
            "RackLocation": {
                "RackUnits": "OU",
                "XLocation": 0,
                "ULocation": 1,
                "UHeight": 8
            },
            "SerialNumber": "",
            "Manufacturer": "",
            "ModelNumber": "",
            "PartNumber": "",
            "FirmwareRevision": "",
            "PowerCapacityWatts": 300,
            "LastPowerOutputWatts": 48
        }
    ],
    "Actions": {},
    "Links": {}
}
```


4.8.1.2 PUT

Operation is not allowed on this resource.

4.8.1.3 PATCH

Operation is not allowed on this resource.

4.8.1.4 POST

Operation is not allowed on this resource.

4.8.1.5 DELETE

Operation is not allowed on this resource.

4.9 ThermalZone collection

ThermalZone collection resource.

Table 9. Thermalzone Collection Attributes

Name	**ThermalZone Collection**	
Type URI	/redfish/v1/Chassis/{`chassisID`}/ThermalZones	
Attribute	**Type**	**Description**
`Name`	String	Name of collection
`Members@odata.count`	Number	Collection members count
`Members`	Array	Contains the members of this collection.

4.9.1 Operations

4.9.1.1 GET

Request:

```
GET /redfish/v1/Chassis/Rack1/ThermalZones
Content-Type: application/json
```

Response:

```
{
    "@odata.context":
"/redfish/v1/$metadata#ThermalZoneCollection.ThermalZoneCollection",
    "@odata.id": "/redfish/v1/Chassis/Rack1/ThermalZones",
    "@odata.type": "#ThermalZoneCollection.ThermalZoneCollection",
    "Name": "Thermal Zones Collection",
    "Members@odata.count": 1,
    "Members": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/ThermalZones/Thermal1"
        }
    ]
}
```

4.9.1.2 PUT

Operation is not allowed on this resource.

4.9.1.3 PATCH

Operation is not allowed on this resource.

4.9.1.4 POST

Operation is not allowed on this resource.

4.9.1.5 DELETE

Operation is not allowed on this resource.

4.10 ThermalZone

This resource is used to represent a thermal zone resource for an Intel® Rack Scale Design implementation. It contains fans and temperature information.

Details of this resource are described in the metadata file, `ThermalZone.xml`.

4.10.1 Operations

4.10.1.1 GET

Request:

```
GET /redfish/v1/Chassis/Rack1/ThermalZones/Thermal1
Content-Type: application/json
```

Response:

```
{
    "@odata.context":
"/redfish/v1/$metadata#Chassis/Rack/ThermalZones/Members/$entity",
    "@odata.type": "ThermalZone.v1_0_0.ThermalZone",
    "@odata.id": "/redfish/v1/Chassis/Rack1/ThermalZones/1",
    "Id": "1",
    "Name": "thermal zone 1",
    "Description": "thermal zone 1",
```

```
    "RackLocation": {
        "RackUnits": "OU",
        "XLocation": 0,
        "ULocation": 1,
        "UHeight": 8
    },
    "Presence": "111100",
    "DesiredSpeedPWM": 50,
    "DesiredSpeedRPM": 3000,
    "MaxFansSupported": 6,
    "NumberOfFansPresent": 6,
    "VolumetricAirflow": 80,
    "Temperatures": [
        {
            "Name": "Inlet Temperature",
            "Status": {
                "State": "Enabled",
                "Health": "OK",
                "HealthRollup": null
            },
            "ReadingCelsius": 21,
            "PhysicalContext": "Intake"
        },
        {
            "Name": "Outlet Temperature",
            "Status": {
                "State": "Enabled",
                "Health": "OK",
                "HealthRollup": null
            },
            "ReadingCelsius": 35,
            "PhysicalContext": "Exhaust"
        }
    ],
    "Status": {
        "State": "Enabled",
        "Health": "OK",
        "HealthRollup": null
    },
    "Fans": [
        {
            "Name": "Fan 1",
            "Status": {
                "State": "Enabled",
                "Health": "OK",
                "HealthRollup": null
            },
            "ReadingRPM": 0,
            "RackLocation": {
                "RackUnits": "OU",
                "XLocation": 0,
                "ULocation": 1,
                "UHeight": 8
            }
        }
```

```
    ],
    "Actions": {},
    "Links": {} }
```

4.10.1.2 PUT

Operation is not allowed on this resource.

4.10.1.3 PATCH

Operation is not allowed on this resource.

4.10.1.4 POST

Operation is not allowed on this resource.

4.10.1.5 DELETE

Operation is not allowed on this resource.

4.11 Storage service collection

Intel® Rack Scale Design storage service collection resource – provides collection of available storage services. Table 10 lists the attributes.

Table 10. Storage Service Collection Attributes

Name	Storage service		
Type URI	/redfish/v1/Services		
Attribute	**Type**	**Mandatory**	**Description**
`Name`	String	Yes	Name of service collection.
`Members@odata.count`	Number	No	Collection members count
`Members`	Array	No	Contains the members of this collection.

4.11.1 Operations

4.11.1.1 GET

Request:

```
GET /redfish/v1/Services
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#Services",
    "@odata.id": "/redfish/v1/Services",
    "@odata.type": "#StorageServiceCollection.StorageServiceCollection",
    "Name": "Storage Services Collection",
    "Description": "Collection of Storage Services",
    "Members@odata.count": 1,
    "Members": [
        {
            "@odata.id": "/redfish/v1/Services/RSS1"
        }
    ]
}
```

4.11.1.2 PUT

Operation is not allowed on this resource.

4.11.1.3 PATCH

Operation is not allowed on this resource.

4.11.1.4 POST

Operation is not allowed on this resource.

4.11.1.5 DELETE

Operation is not allowed on this resource.

4.12 Composed node collection

Intel® Rack Scale Design Composed Node collection resource – provides collection of all logical nodes. Table 11 lists the attributes.

Table 11. Composed Node Collection Attributes

Name	Composed node collection		
Type URI	/redfish/v1/Nodes		
Attribute	**Type**	**Mandatory**	**Description**
`Name`	String	Yes	Name of collection
`Members@odata.count`	Number	No	Collection members count
`Members`	Array	No	Contains the members of this collection.
`Actions`	Object	No	Actions available: **Allocate** – this action is the first mandatory step to create a composed node. In response to this action, proper resources will be found and allocated for node composition. A node resource will be created and URL (link) of this node will be returned. To allocate a Composed Node using the PodM REST API, it is necessary to create a JSON template describing the requested resources. The JSON template may contain various details concerning resources to be used in the Composed Node. All JSON template elements are optional, but should not be mutually exclusive. It is possible to supply PodM with a JSON template containing no specific requirements (e.g., {} – a pair of empty curly braces in HTTP request body), thus allowing PodM to propose a Composed Node containing resources chosen arbitrarily by PodM. Format of the JSON template (action payload) is described in Table 12. For more information about node allocation and assembly, refer to the PODM Allocation Guide document.

Table 12. Composed Node Allocation Action Attributes

Attribute	**Type**	**Mandatory**	**Description**
`Name`	String	No	Name of the composed node. Note: Because `ComposedNode` is a Redfish resource, its Name field is mandatory, so an attempt to directly set a **null** value results in an expected error. PodM will set a default name for a newly-created `ComposedNode` resource only upon not supplying the Name attribute.

<table>
<tr><th>Attribute</th><th>Type</th><th>Mandatory</th><th>Description</th></tr>
<tr><td>Description</td><td>String</td><td>No</td><td>Description of the composed node</td></tr>
<tr><td>Processors</td><td>Array</td><td>No</td><td>Array of requirements for the processor for the composed node. Each processor requirement may contain one or more optional attributes:
<table>
<tr><th>Attribute</th><th>Type</th><th>Description</th></tr>
<tr><td>Model</td><td>String</td><td>Processor model that should be used for composed node (exact model name)</td></tr>
<tr><td>TotalCores</td><td>Number</td><td>Minimum number of processor cores</td></tr>
<tr><td>AchievableSpeedMHz</td><td>Number</td><td>Minimum achievable processor operating frequency.</td></tr>
<tr><td>InstructionSet</td><td>String</td><td>Processor supported instruction set.
“x86” – x86 32-bit
“x86-64” – x86 64-bit
“IA-64” – Intel IA-64
“ARM-A32” – ARM 32-bit
“ARM-A64” – ARM 64-bit
“MIPS32” – MIPS 32-bit
“MIPS64” – MIPS 64-bit
“OEM” – OEM-defined</td></tr>
<tr><td>Resource</td><td>Object</td><td>Reference to particular processor that should be used in composed node</td></tr>
<tr><td>Chassis</td><td>Object</td><td>Link to chassis object within this processor should be contained.</td></tr>
<tr><td>Brand</td><td>String</td><td>Brand of CPU that should be used to allocate node. Allowable values:
Intel® Xeon® family: E3, E5, E7
SoC/Atom® family: X3 (Avoton), X5 (Broadwell-DE), X7
Core family: I3, I5, I7
“Unknown” – processor doesn’t fit into any above categories</td></tr>
<tr><td>Capabilities</td><td>Array</td><td>Array of strings describing processor capabilities (like reported in /proc/cpuinfo flags), such as:
“sse” – Streaming SIMD Extensions
“avx” – Advanced Vector Extensions</td></tr>
<tr><td>ProcessorType</td><td>String</td><td>This property contains the string which identifies the type of processor:
“CPU”
“FPGA”
“GPU”
“DSP”
“Accelerator”
“OEM”</td></tr>
</table></td></tr>
<tr><td>Memory</td><td>Array</td><td>No</td><td>Array of requirements for memory for the composed node.
<table>
<tr><th>Attribute</th><th>Type</th><th>Description</th></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Type</th><th>Mandatory</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>CapacityMiB</td><td>Number</td><td>Minimum memory capacity requested for composed node</td></tr>
<tr><td>MemoryDeviceType</td><td>String</td><td>Type details of DIMM:
“DDR”
“DDR2”
“DDR3”
“DDR4”
“DDR4 SDRAM”
“DDR4E SDRAM”
“LPDDR4SDRAM”
“DDR3 SDRAM”
“LPDDR3 SDRAM”
“DDR2 SDRAM”
“DDR2 SDRAM-FB-DIMM”
“DDR2SDRAM-FB-DIMM PROBE”
“DDR SGRAM”
“DDR SDRAM”
“ROM”
“SDRAM”
“EDO”
“FastPageMode”
“PipelinedNibble”</td></tr>
<tr><td>SpeedMHz</td><td>Number</td><td>Minimum supported memory speed.</td></tr>
<tr><td>Manufacturer</td><td>String</td><td>Requested memory manufacturer.</td></tr>
<tr><td>DataWidthBits</td><td>Number</td><td>Requested memory data width in bits.</td></tr>
<tr><td>Resource</td><td>Object</td><td>Reference to particular memory module that should be used in composed node</td></tr>
<tr><td>Chassis</td><td>Object</td><td>Link to chassis object within this memory DIMM should be contained.</td></tr>
</table>
</td></tr>
<tr><td>RemoteDrives</td><td>Array</td><td>No</td><td>Array of requirements for remote drives that should be created/connected to the composed node
<table>
<tr><th>Attribute</th><th>Type</th><th>Description</th></tr>
<tr><td>CapacityGiB</td><td>Number</td><td>Minimum drive capacity requested for composed node</td></tr>
<tr><td>iSCSIAddress</td><td>String</td><td>Defines TargetIQN of Remote Target to be set for new Remote Target (should be unique in PodM).
Note: Master property is required for creating new target.</td></tr>
<tr><td>Master</td><td>Object</td><td>Defines master logical volume that should be taken to create new remote target. It contains properties described in Table 13.</td></tr>
</table>
</td></tr>
<tr><td>LocalDrives</td><td>Array</td><td>No</td><td>Array of requirements for local drives for the composed node.
<table>
<tr><th>Attribute</th><th>Type</th><th>Description</th></tr>
<tr><td>CapacityGiB</td><td>Number</td><td>Minimum drive capacity requested for composed node</td></tr>
<tr><td>Type</td><td>String</td><td>Drive type</td></tr>
</table>
</td></tr>
</table>

Attribute	Type	Mandatory	Description
			<table><tr><td></td><td></td><td>“HDD” “SSD”</td></tr><tr><td>MinRPM</td><td>Number</td><td>Minimum rotation speed of requested drive.</td></tr><tr><td>SerialNumber</td><td>String</td><td>Serial number of requested drive</td></tr><tr><td>Interface</td><td>String</td><td>Interface of requested drive: “SAS” “SATA” “NVMe”</td></tr><tr><td>Resource</td><td>Object</td><td>Reference to particular local drive that should be used in composed node</td></tr><tr><td>Chassis</td><td>Object</td><td>Link to chassis object within this drive should be contained.</td></tr><tr><td>FabricSwitch</td><td>Boolean</td><td>Determine if local drive should be connected using fabric switch or locally connected.</td></tr></table>
EthernetInterfaces	Array	No	Array of requirements for Ethernet interfaces of composed node. <table><tr><th>Attribute</th><th>Type</th><th>Description</th></tr><tr><td>SpeedMbps</td><td>Number</td><td>Minimum speed of Ethernet interface requested for composed node</td></tr><tr><td>VLANs</td><td>Array</td><td>Array of VLANs that should be configured on connected switch port for composed node for given Ethernet interface in following format: **VLANId** – number indicating VLAN Id **Tagged** – Boolean value describing if given VLAN is tagged. **Deprecated**: This is going to be removed in future releases of RSD. Equivalent functionality will be provided via CRUD operation on Ethernet Switch port VLANs.</td></tr><tr><td>PrimaryVLAN</td><td>Number</td><td>Primary VLAN ID that should be set for given Ethernet Interface **Deprecated**: This is going to be removed in future releases of RSD. Equivalent functionality will be provided via CRUD operation on Ethernet Switch port.</td></tr><tr><td>Resource</td><td>Object</td><td>Reference to particular Ethernet interface that should be used in composed node</td></tr><tr><td>Chassis</td><td>Object</td><td>Link to chassis object within this network interface should be contained.</td></tr></table>
TotalSystemMemory	Number	No	Minimum total amount of memory in composed node.
TotalSystemCoreCount	Number	No	Minimum total processor core count in composed node.

Attribute	Type	Mandatory	Description
Security	Object	No	Security requirements for Composed Node

Attribute	Type	Description
TpmPresent	Boolean	This is used to specify if composed node should have TPM module present.
TpmInterfaceType	String (enum)	This is the used to specify requested TPM interface type. It overrides TpmPresent attribute.
TxtEnabled	Boolean	This is used to specify if composed node should have TXT mode enabled.

Table 13. Remote Master Target Properties

Attribute	Type	Description
Type	String	Type of replication of master drive: Clone – volume should be cloned Snapshot – Copy on Write should be created from indicated volume
Resource	Object	Reference to logical volume that should be used as master for replication.

4.12.1 Operation

4.12.1.1 GET

Request:

```
GET /redfish/v1/Nodes
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#Nodes",
    "@odata.id": "/redfish/v1/Nodes",
    "@odata.type": "#ComposedNodeCollection.ComposedNodeCollection",
    "Name": "Composed Nodes Collection",
    "Members@odata.count": 1,
    "Members": [
        {
            "@odata.id": "/redfish/v1/Nodes/Node1"
        }
    ],
    "Actions": {
        "#ComposedNodesCollection.Allocate": {
            "target": "/redfish/v1/Nodes/Actions/Allocate"
        }
    }
}
```

4.12.1.2 PUT

Operation is not allowed on this resource.

4.12.1.3 PATCH

Operation is not allowed on this resource.

4.12.1.4 POST

Note: Currently a user can request allocation of a single node with a single request. Node components – CPU, memory, local storage, network interface – must be located on a single physical blade. Remote storage can be located anywhere in the pod.

The JSON listed below is just an example. For more details, refer to Section 6.1.

Request:

```
POST /redfish/v1/Nodes/Actions/Allocate
Content-Type: application/json
{
       "Name": "My first composed node",
       "Description": "Test node",
       "Processors": [{
                "Model": "Multi-Core Intel(R) Xeon(R) processor 7xxx Series",
                "TotalCores": 2,
                "AchievableSpeedMHz": 2000,
                "InstructionSet": "x86",
                "Oem": {
                      "Brand": "E5",
                      "Capabilities": [
                              "sse"
                      ],

                },
                "Resource": {
                      "@odata.id":
"/redfish/v1/Systems/System1/Processors/CPU1"
                }
       }],
       "Memory": [{
                "CapacityMiB": 16000,
                "MemoryDeviceType": "DDR3",
                "SpeedMHz": 1600,
                "Manufacturer": "Intel",
                "DataWidthBits": 64,
                "Resource": {
                      "@odata.id": "/redfish/v1/Systems/System1/Memory/Dimm1"
                },
                "Chassis": {
                      "@odata.id": "/redfish/v1/Chassis/Rack1"
                }
       }],
       "RemoteDrives": [{
                "CapacityGiB": 80,
                "iSCSIAddress": "iqn.oem.com:fedora21",
                "Master": {
                      "Type": "Snapshot",
                      "Resource": {
                              "@odata.id":
"/redfish/v1/Services/RSS1/LogicalDrives/sda1"
                      }
                }
       }],
       "LocalDrives": [{
```

```
                "CapacityGiB": 500,
                "Type": "HDD",
                "MinRPM": 5400,
                "SerialNumber": "12345678",
                "Interface": "SATA",
                "Resource": {
                        "@odata.id": "redfish/v1/Chassis/Blade1/Drives/Disk1"
                },
                "FabricSwitch": false
        }],
        "EthernetInterfaces": [{
                "SpeedMbps": 1000,
                "PrimaryVLAN": 100,
                "VLANs": [{
                        "VLANId": 100,
                        "Tagged": false
                }],
                "Resource": {
                        "@odata.id":
"/redfish/v1/Systems/System1/EthernetInterfaces/LAN1"
                }
        }],
        "Security": {
          "TpmPresent": true,
          "TpmInterfaceType": “TPM2_0”,
          “TxtEnabled”: false
        },
        "Oem": {
        },
        "TotalMemoryCapacityMiB": 32000,
        "TotalProcessorCoreCount": 2
}
```

Response:

```
HTTP/1.1 201 Created
Location: http://<IP>:<Port>/redfish/v1/Nodes/2
```

4.12.1.5 DELETE

Operation is not allowed on this resource.

4.13 Composed node

Composed node resource – provides detailed information about an assembled logical node identified by {`nodeID`}. Table 14 lists the attributes.

Table 14. Composed Node Attributes

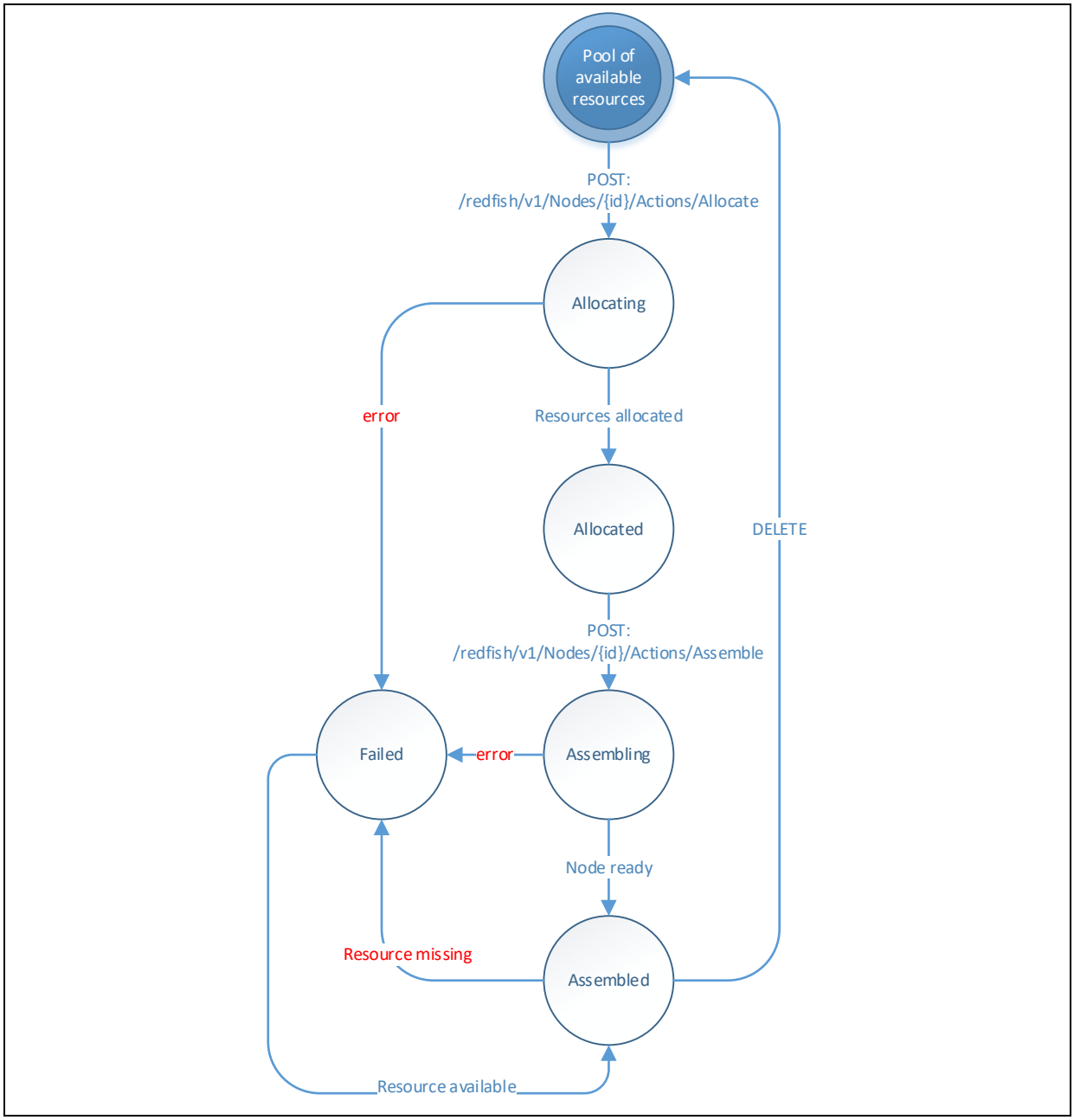
Name	Composed node		
Type URI	/redfish/v1/Nodes/{`nodeID`}		
Attribute	**Type**	**Mandatory**	**Description**
`Id`	String	Yes	Provides a ID of this resource
`Name`	String	Yes	Name of composed node
`Description`	String	No	User provided node description

Name	Composed node		
Type URI	/redfish/v1/Nodes/{nodeID}		
Attribute	Type	Mandatory	Description
UUID	String	No	UUID of computer system used as a base for this node.
PowerState	String (enum)	No	This is the current power state of the node "On" – The system is powered on. "Off" – The system is powered off, although some components may continue to have AUX power such as management controller. "PoweringOn" – A temporary state between Off and On. This temporary state can be very short. "PoweringOff" – A temporary state between On and Off. The power-off action can take time while the OS is in the shutdown process/
Processors	Object	Yes	**Name** / **Type** / **Mandatory** / **Description** Count / Number / No / Number of CPUs Model / String, Null / No / Basic information about processor model Status / Object / No / See Section 5.1 for status of resource
Memory	Object	Yes	**Name** / **Type** / **Mandatory** / **Description** TotalSystemMemoryGiB / Number / No / Amount of installed memory in GiB Status / Object / No / See Section 5.1 for status of resource
Status	Object, null	No	See Section 5.1 for status of resource.
ComposedNodeState	String (enum)	Yes	Current state of assembly process for this node. • Allocating: Allocating resources for node is in progress. Next state can be Allocated or Failed. • Allocated: Node resources has been allocated, but assembly not started yet. After ComposedNode.Assemble action state will progress to Assembling. • Assembling: Assembly process initiated, but not finished yet. When done it will change into Assembled. • Assembled: Node successfully assembled. • Failed: Allocation or assembly process failed, or in runtime one of composing components was removed or transitioned in error state.
Boot	Object	No	**Name** / **Type** / **Required** / **Description** BootSourceOverrideEnabled / String, Null / No / State of the Boot Source Override feature. Proper values: "Disabled" "Once" "Continuous" BootSourceOverrideTarget / String, Null / No / The current boot source to be used at next boot instead of the normal boot device, if BootSourceOverrideEnabled is true BootSourceOverrideTarget@Redfish.AllowableValues / Array / No / Array of supported boot sources

<table>
<tr><td>Name</td><td colspan="5">Composed node</td></tr>
<tr><td>Type URI</td><td colspan="5">/redfish/v1/Nodes/{nodeID}</td></tr>
<tr><td>Attribute</td><td>Type</td><td>Mandatory</td><td colspan="3">Description</td></tr>
<tr><td></td><td></td><td></td><td>BootSourceOverrideMode</td><td>String, Null</td><td>No</td><td>The BIOS Boot Mode (either Legacy or UEFI) to be used when BootSourceOverrideTarget boot source is booted from</td></tr>
<tr><td></td><td></td><td></td><td>BootSourceOverrideMode@Redfish.AllowableValues</td><td>Array</td><td>No</td><td>Array of supported boot modes</td></tr>
<tr><td>Oem</td><td>Object, Null</td><td>No</td><td colspan="3">OEM defined object</td></tr>
<tr><td>Links</td><td>Object</td><td>No</td><td colspan="3">Link section:
Name | Type | Required | Description
ComputerSystem | Object, null | Yes | Reference to ComputerSystem resource used to compose this node
Processors | Array | No | Array of references to Processor resources
Memory | Array | No | Array of references to Memory resources
RemoteDrives | Array | No | An array of references to the remote storage drives
LocalDrives | Array | No | An array of references to the computer system local storage drives
EthernetInterfaces | Array | No | Array of links to Ethernet Interface collection associated with this Composed Node
ManagedBy | Array | No | An array of references to Managers responsible for this Composed Node</td></tr>
<tr><td>Actions</td><td>Object</td><td>Yes</td><td colspan="3">Actions available for this node:
• Reset action with following values:
o On – Turn the system on.
o ForceOff – Turn the system off immediately (non-graceful) shutdown.
o GracefulRestart – Perform a graceful system shutdown followed by a restart of the system.
o ForceRestart – Perform an immediate (non-graceful) shutdown, followed by a restart of the system.
o Nmi – Generate a Diagnostic Interrupt (usually an NMI on x86 systems) to cease normal operations, perform diagnostic actions and typically halt the system.
o ForceOn – Turn the system on immediately.
o PushPowerButton - Simulate the pressing of the physical power button on this system.
o GracefulShutdown – Initiate a soft-shutdown of OS via ACPI.
• Assemble: Doesn't consume any parameters. Second step of creating a composed node (after Allocate Action on Nodes collection). That action will assemble a logical node – initiate ComposedNodeState change from Allocated state into Assembling state.</td></tr>
</table>

Name	Composed node		
Type URI	/redfish/v1/Nodes/{nodeID}		
Attribute	Type	Mandatory	Description
			After finishing assembly, composed node will stay in Off PowerState. To change its state, one needs to execute Reset action with "On" parameter (make sure that Boot is set properly). • AttachEndpoint – this action allows to attach endpoint (functional device) like PCI drives on PNC switch, from available pool to composed node. Action can be performed when Composed Node is Assembled or Failed. One of these parameters are required: o Resource – Link to endpoint (realized by PCI device function). Must be connected to PNC switch servicing ComputerSystem hosting this ComposedNode. o CapacityGiB – Requested capacity of PNC drive. PODM will find available drive in the pool and connect it to this node. **Note: Currently only PCI NVMe drives are supported by this action.** • DetachEndpoint – Action used to detach already connected PNC endpoint (functional device). In case of storage drive, if EraseOnDetach parameter of drive is set to *"true"*, this drive will be Secure Erased before returning to the pool. Action can be performed when Composed Node is Assembled or Failed. It takes one argument: o Resource – Link to PNC endpoint realized by PCI device function that needs to be detached.

Figure 2. Composed Node State Changes during Assembly Process



4.13.1 Operations

4.13.1.1 GET

Request:

```
GET /redfish/v1/Nodes/{nodeID}
Content-Type: application/json
```

Response:


```
{
    "@odata.context": "/redfish/v1/$metadata#Nodes/Members/$entity",
    "@odata.id": "/redfish/v1/Nodes/Node1",
    "@odata.type": "#ComposedNode.v1_0_0.ComposedNode",
    "Id": "Node1",
    "Name": "Composed Node",
    "Description": "Node #1",
    "UUID": "00000000-0000-0000-0000-000000000000 - the same as Computer
System",
    "PowerState": "On",
    "Status": {
        "State": "Enabled",
        "Health": "OK",
        "HealthRollup": "OK"
    },
    "Processors": {
        "Count": 2,
        "Model": "Multi-Core Intel(R) Xeon(R) processor 7xxx Series",
        "Status": {
            "State": "Enabled",
            "Health": "OK"
        }
    },
    "Memory": {
        "TotalSystemMemoryGiB": 32,
        "Status": {
            "State": "Enabled",
            "Health": "OK"
        }
    },
    "ComposedNodeState": "Allocated",
    "Boot": {
        "BootSourceOverrideEnabled": "Disabled",
        "BootSourceOverrideTarget": "None",
        "BootSourceOverrideTarget@Redfish.AllowableValues": [
            "None",
            "Pxe",
            "Hdd",
            "RemoteDrive"
        ] ,
    "BootSourceOverrideMode": "Legacy",
    "BootSourceOverrideMode@Redfish.AllowableValues":  ["Legacy",
      "UEFI"]
    },
    "Oem": {},
    "Links": {
        "ComputerSystem": {
            "@odata.id": "/redfish/v1/Systems/System1"
        },
        "Processors": [
            {
                "@odata.id": "/redfish/v1/Systems/System1/Processors/CPU1"
            }
        ],
```

```
        "Memory": [
            {
                "@odata.id": "/redfish/v1/Systems/System1/Memory/Dimm1"
            }
        ],
        "EthernetInterfaces": [
            {
                "@odata.id":
"/redfish/v1/Systems/System1/EthernetInterfaces/LAN1"
            }
        ],
        "LocalDrives": [
            {
                "@odata.id": "/redfish/v1/Chassis/Blade1/Drives/1"
            }
        ],
        "RemoteDrives": [
            {
                "@odata.id": "/redfish/v1/Services/RSS1/Targets/target1"
            }
        ],
        "ManagedBy": [
            {
                "@odata.id": "/redfish/v1/Managers/PODM"
            }
        ],
        "Oem": {}
    },
    "Actions": {
        "#ComposedNode.Reset": {
            "target": "/redfish/v1/Nodes/Node1/Actions/ComposedNode.Reset",
            "ResetType@Redfish.AllowableValues": [
                "On",
                "ForceOff",
                "GracefulRestart",
                "ForceRestart",
                "Nmi",
                "ForceOn",
                "PushPowerButton",
                "GracefulShutdown"
            ]
        },
        "#ComposedNode.Assemble": {
            "target": "/redfish/v1/Nodes/Node1/Actions/ComposedNode.Assemble"
        },
    "#ComposedNode.AttachEndpoint": {
      "target":
"/redfish/v1/Nodes/Node1/Actions/ComposedNode.AttachEndpoint",
      "Resource@Redfish.AllowableValues": [

{"@odata.id":"/redfish/v1/Chassis/PCIeSwitchChassis/Drives/Disk.Bay.1"},

{"@odata.id":"/redfish/v1/Chassis/PCIeSwitchChassis/Drives/Disk.Bay.2"}
      ]
    },
```

```
    "#ComposedNode.DetachEndpoint": {
      "target":
"/redfish/v1/Nodes/Node1/Actions/ComposedNode.DetachEndpoint",
      "Resource@Redfish.AllowableValues": [

{"@odata.id":"/redfish/v1/Chassis/PCIeSwitchChassis/Drives/Disk.Bay.3"}
      ]
    }
    }
}
```

4.13.1.2 PUT

Operation is not allowed on this resource.

4.13.1.3 PATCH

The following properties can be updated by the PATCH operation:

Table 15. Boot Override PATCH Operation

Attribute	Type	Required	Description
`Boot`	Object	No	Boot override properties, details in Table 16

Table 16 describes “Boot” properties that can be patched.

Table 16. Boot Override Update Properties

Attribute	Type	Required	Description
`BootSourceOverrideEnabled`	String	No	Describes the state of the Boot Source Override feature. Allowed values: “Disabled” - The system will boot as normal “Once” - On its next boot cycle, the system will boot (one time) to the Boot Source Override Target “Continuous” - The system will boot to the target specified in the `BootSourceOverrideTarget` until this property is set to Disabled
`BootSourceOverrideTarget`	String	No	The current boot source to be used at the next boot, instead of the normal boot device, if `BootSourceOverrideEnabled` is true. Supported values: “None” - Boot from the normal boot device “Pxe” - Boot from the Pre-Boot EXecution (PXE) environment “Hdd” - Boot from a hard drive
`BootSourceOverrideMode`	String	No	The BIOS Boot Mode (either Legacy or UEFI) to be used when the `BootSourceOverrideTarget` boot source is booted from: “Legacy” - The system will boot in non-UEFI boot mode to the Boot Source Override Target “UEFI” - The system will boot in UEFI boot mode to the Boot Source Override Target

Note: The Boot property represents only override values, not the current boot source configured on system. To make sure that the correct boot source/mode will be applied, it is recommended to send PATCH to the boot property after node assembly, and before powering it on.

```
PATCH /redfish/v1/Nodes/Node1
Content-Type: application/json
{
       “Boot”: {
              “BootSourceOverrideEnabled”: “Once”,
              “BootSourceOverrideTarget”: “Pxe”,
              “BootSourceOverrideMode”: “Legacy”
       }
```

```
}
```

Response:

```
HTTP/1.1 204 No Content
```

Or:

```
HTTP/1.1 200 OK
```

With full resource representation.

4.13.1.4 POST

4.13.1.4.1 Reset Action

Request:

```
POST /redfish/v1/Nodes/1/Actions/ComposedNode.Reset
Content-Type: application/json
{
       “ResetType”: “On”
}
```

Response:

```
HTTP/1.1 204 No Content
```

4.13.1.4.2 Assemble action

Request:

```
POST /redfish/v1/Nodes/1/Actions/ComposedNode.Assemble
```

Response:

```
HTTP/1.1 204 No Content
```

4.13.1.4.3 AttachEndpoint action (1) attaching specific endpoint (PNC drive) to existing composed node.

Request:

```
POST /redfish/v1/Nodes/1/Actions/ComposedNode.AttachEndpoint
Content-Type: application/json
{
    "Resource": {
        “@odata.id":"/redfish/v1/Chassis/PCIeSwitchChassis/Drives/Disk.Bay.1"
    }
}
```

Response:

```
HTTP/1.1 204 No Content
```

4.13.1.4.4 AttachEndpoint action (2) attaching endpoint meeting criteria provided in request body to existing composed node.

Request:

```
POST /redfish/v1/Nodes/1/Actions/ComposedNode.AttachEndpoint
Content-Type: application/json
{
    "CapacityGiB": 40
}
```

Response:

```
HTTP/1.1 204 No Content
```

4.13.1.4.5 DetachDrive action

Request:

```
POST /redfish/v1/Nodes/1/Actions/ComposedNode.DetachEndpoint
Content-Type: application/json
{
    “Resource”: {
       "@odata.id":"/redfish/v1/Chassis/PCIeSwitchChassis/Drives/Disk.Bay.3"
    }
}
```

Response:

```
HTTP/1.1 204 No Content
```

4.13.1.5 DELETE

Upon deletion (disassembly) of a Composed Node, several actions are performed:

- Force off request is sent to the computer system.
- All VLANs (except for reserved ones – see *Reserved VLANs*) are removed from Ethernet switch ports associated with the computer system's Ethernet interfaces.
- All PCIe devices connected to the node via a PCIe switch are detached.
- All drives attached via a PCIe switch with the property `EraseOnDetach` set to *true* are securely erased.

The DELETE action removes the affected components' reservation (deallocation) and puts them back to the resource pool.

In case when any resource cannot be removed from composed node, *ComposedNodeState* will be changed to *Failed*. All remaining resources will return to proper pools. Resending DELETE operation will remove node, without performing actions described above.

Request:

```
DELETE /redfish/v1/Nodes/1
```

Response:

```
HTTP/1.1 204 No Content
```

4.14 PSME and Storage Services resources

PODM supports PSME and Storage Services resources. Table 17 describes which resources and their operations are included as a part of the Intel® RSD PODM API Specification.

For those resources refer to the PSME and Storage Services specifications.

Table 17 PSME and Storage Services resources

Resource Name	Supported Operations				
	GET	PATCH	POST	DELETE	Actions
Computer System	X	X			X
ComputerSystemMetrics					
Processor	X				
ProcessorMetrics	X				
Memory	X				
MemoryMetrics	X				
Storage	X				
Drive	X	X			X
Network Interface	X				
VLAN	X		X	X	
Manager	X				
Network Protocol	X				
Ethernet Switch	X				
Ethernet Switch Port	X	X	X	X	
Ethernet Switch ACL	X	X	X	X	X
Ethernet Switch ACL rules	X	X	X	X	
Ethernet Switch Port static MACs	X	X	X	X	
Fabric	X				
Zone	X	X			
PCIe Device	X	X			
Fabric Switch	X				
PCIe Port	X				
Fabric Port Metrics					
PCIe Function	X				
Endpoint	X				
Storage Service	X				
Remote Target	X	X			
Logical Drive	X				
Physical Drive	X				
EventService	X				
EventSubscription	X		X	X	
NetworkInterface	X				
NetworkDeviceFunction	X	X			
MetricDefinition	X				

Resource Name	Supported Operations				
	GET	PATCH	POST	DELETE	Actions
Thermal	X				
Power	X				

4.15 Simple Storage collection

Table 18. Simple Storage Collection Attributes

Name	Simple storage	
Type URI	/redfish/v1/Systems/System1/SimpleStorage	
Attribute	**Type**	**Description**
`Name`	String	Name of collection
`Members@odata.count`	Number	Collection members count
`Members`	Array	Contains the members of this collection.

4.15.1 Operations

4.15.1.1 GET

Request:

```
GET /redfish/v1/Systems/System1/SimpleStorage
Content-Type: application/json
```

Response:

```
{
    "@odata.context":
"/redfish/v1/$metadata#SimpleStorageCollection.SimpleStorageCollection",
    "@odata.id": "/redfish/v1/Systems/System1/SimpleStorage",
    "@odata.type": "#SimpleStorageCollection.SimpleStorageCollection",
    "Name": "Simple Storage Collection",
    "Members@odata.count": 1,
    "Members": [
        {
            "@odata.id": "/redfish/v1/Systems/System1/SimpleStorage/Storage1"
        }
    ]}
```

4.15.1.2 PUT

Operation is not allowed on this resource.

4.15.1.3 PATCH

Operation is not allowed on this resource.

4.15.1.4 POST

Operation is not allowed on this resource.

4.15.1.5 DELETE

Operation is not allowed on this resource.

4.16 Simple storage

Simple storage devices associated with this system.

Details of this resource are described in the metadata file, `SimpleStorage.xml`.

4.16.1 Operations

4.16.1.1 GET

Request:

```
GET /redfish/v1/Systems/System1/SimpleStorage/Storage1
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#SimpleStorage.SimpleStorage",
    "@odata.id": "/redfish/v1/Systems/System1/SimpleStorage/Storage1",
    "@odata.type": "#SimpleStorage.v1_1_0.SimpleStorage",
    "Id": "Storage1",
    "Name": "Simple Storage Controller",
    "Description": "System SATA",
    "UefiDevicePath": "UEFI Device Path",
    "Status": {
        "State": "Enabled",
        "Health": "OK",
        "HealthRollup": "OK"
    },
    "Devices": [
        {
            "@odata.type": "#SimpleStorage.v1_1_0.Device",
            "Name": "Drive 1",
            "Manufacturer": "ACME",
            "Model": "Drive Model string",
           "CapacityBytes": 322122547200,
            "Status": {
                "State": "Enabled",
                "Health": "OK"
            }
        },
        {
            "@odata.type": "#SimpleStorage.v1_1_0.Device",
            "Name": "Drive 2",
            "Manufacturer": "SuperDuperSSD",
            "Model": "Drive Model string",
           "CapacityBytes": 68719476736,
            "Status": {
                "State": "Enabled",
                "Health": "OK"
            }
        },
        {
            "Name": "Drive 3",
            "Status": {
                "State": "Absent"
```

```
            }
        },
        {
            "Name": "Drive 4",
            "Status": {
                "State": "Absent"
            }
        }
    ]
}
```

4.16.1.2 PUT

Operation is not allowed on this resource.

4.16.1.3 PATCH

Operation is not allowed on this resource.

4.16.1.4 POST

Operation is not allowed on this resource.

4.16.1.5 DELETE

Operation is not allowed on this resource.

4.17 Power

Power metrics resource. It represents the properties for Power Consumption and Power Limiting.

Detailed info about the resource properties can be obtained from the metadata file, `Power.xml`.

4.17.1 Operations

4.17.1.1 GET

Request:

```
GET /redfish/v1/Chassis/Rack1/Power
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#Power.Power",
    "@odata.id": "/redfish/v1/Chassis/Rack1/Power",
    "@odata.type": "#Power.v1_1_0.Power",
    "Id": "Power",
    "Name": "Power",
    "Description": "PowerSubsystem",
    "PowerControl": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/Power#/PowerControl/0",
            "MemberId": "0",
               "Name": "System Power Control",
            "PowerConsumedWatts": 8000,
            "PowerRequestedWatts": 8500,
            "PowerAvailableWatts": 8500,
            "PowerCapacityWatts": 10000,
```

```
            "PowerAllocatedWatts": 8500,
            "PowerMetrics": {
                "IntervalInMin": 30,
                "MinConsumedWatts": 7500,
                "MaxConsumedWatts": 8200,
                "AverageConsumedWatts": 8000
            },
            "PowerLimit": {
                "LimitInWatts": 9000,
                "LimitException": "LogEventOnly",
                "CorrectionInMs": 42
               },
            "RelatedItem": [
                {"@odata.id": "/redfish/v1/Chassis/Drawer1"},
                {"@odata.id": "/redfish/v1/Systems/System1"}
            ],
             "Status": {
                   "State": "Enabled",
                   "Health": "OK",
                "HealthRollup": "OK"
               },
               "Oem": {}
        }
    ],
    "Voltages": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/Power#/Voltages/0",
            "MemberId": "0",
            "Name": "VRM1 Voltage",
            "SensorNumber": 11,
            "Status": {
                "State": "Enabled",
                "Health": "OK"
            },
            "ReadingVolts": 12,
            "UpperThresholdNonCritical": 12.5,
            "UpperThresholdCritical": 13,
            "UpperThresholdFatal": 15,
            "LowerThresholdNonCritical": 11.5,
            "LowerThresholdCritical": 11,
            "LowerThresholdFatal": 10,
            "MinReadingRange": 0,
            "MaxReadingRange": 20,
            "PhysicalContext": "VoltageRegulator",
            "RelatedItem": [
                {"@odata.id": "/redfish/v1/Systems/System1" }
            ]
        }
    ],
    "PowerSupplies": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/Power#/PowerSupplies/0",
            "MemberId": "0",
            "Name": "Power Supply Bay 1",
            "Status": {
```


```
                "State": "Enabled",
                "Health": "Warning"
            },
            "Oem": {},
            "PowerSupplyType": "DC",
            "LineInputVoltageType": "DCNeg48V",
            "LineInputVoltage": -48,
            "PowerCapacityWatts": 400,
            "LastPowerOutputWatts": 192,
            "Model": "499253-B21",
            "Manufacturer": "ManufacturerName",
            "FirmwareVersion": "1.00",
            "SerialNumber": "1z0000001",
            "PartNumber": "1z0000001A3a",
            "SparePartNumber": "0000001A3a",
               "InputRanges": [
               {
                   "InputType": "DC",
                   "MinimumVoltage": -47,
                   "MaximumVoltage": -49,
                   "OutputWattage": 400,
                   "MinimumFrequencyHz": 50,
                   "MaximumFrequencyHz": 60,
                   "Oem": {}
              }
            ],
            "IndicatorLED": "Off",
            "RelatedItem": [
              { "@odata.id": "/redfish/v1/Chassis/Rack1" }
            ],
            "Redundancy": [
              { "@odata.id": "/redfish/v1/Chassis/1/Power#/Redundancy/0" }
            ]
        }
    ],
    "Redundancy": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/Power#/Redundancy/0",
            "MemberId": "0",
            "Name": "PowerSupply Redundancy Group 1",
            "Mode": "Failover",
            "MaxNumSupported": 2,
            "MinNumNeeded": 1,
            "RedundancySet": [
                { "@odata.id": "/redfish/v1/Chassis/1/Power#/PowerSupplies/0"
}
            ],
            "Status": {
                "State": "Offline",
                "Health": "OK"
            }
        }
    ],
    "Oem": {}
}
```

4.17.1.2 PUT

Operation is not allowed on this resource.

4.17.1.3 PATCH

Operation is not allowed on this resource.

4.17.1.4 POST

Operation is not allowed on this resource.

4.17.1.5 DELETE

Operation is not allowed on this resource.

4.18 Thermal

Thermal metrics resource. It represents the properties for Temperature and Cooling.

Detailed info about the resource properties can be obtained from metadata file, `Thermal.xml`.

4.18.1 Operations

4.18.1.1 GET

Request:

```
GET /redfish/v1/Chassis/Rack1/Thermal
Content-Type: application/json
```

Response:

```
{
    "@odata.context": "/redfish/v1/$metadata#Thermal.Thermal",
    "@odata.id": "/redfish/v1/Chassis/Rack1/Thermal",
    "@odata.type": "#Thermal.v1_1_0.Thermal",
    "Id": "Thermal",
    "Name": "Thermal",
    "Description": "Thermal Subsystem",
    "Temperatures": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/Thermal#/Temperatures/0",
            "MemberId": "0",
            "Name": "Drawer inlet Temp",
            "SensorNumber": 42,
            "Status": {
                "State": "Enabled",
                "Health": "OK"
            },
            "ReadingCelsius": 21,
            "UpperThresholdNonCritical": 42,
            "UpperThresholdCritical": 42,
            "UpperThresholdFatal": 42,
            "LowerThresholdNonCritical": 42,
            "LowerThresholdCritical": 5,
            "LowerThresholdFatal": 42,
            "MinReadingRange": 0,
            "MaxReadingRange": 200,
```


```
            "PhysicalContext": "Intake",
            "RelatedItem": [
                {"@odata.id": "/redfish/v1/Chassis/Drawer1" }
            ]
        }
    ],
    "Fans": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/Thermal#/Fans/0",
            "MemberId": "0",
            "Name": "BaseBoard System Fan",
            "PhysicalContext": "Backplane",
            "Status": {
                "State": "Enabled",
                "Health": "OK"
            },
            "Reading": 2100,
            "ReadingUnits": "RPM",
            "UpperThresholdNonCritical": 42,
            "UpperThresholdCritical": 4200,
            "UpperThresholdFatal": 42,
            "LowerThresholdNonCritical": 42,
            "LowerThresholdCritical": 5,
            "LowerThresholdFatal": 42,
            "MinReadingRange": 0,
            "MaxReadingRange": 5000,
            "Redundancy" : [
                {"@odata.id":
"/redfish/v1/Chassis/Rack1/Thermal#/Redundancy/0"}
            ],
            "RelatedItem" : [
                {"@odata.id": "/redfish/v1/Chassis/Rack1" }
            ]
        }
    ],
    "Redundancy": [
        {
            "@odata.id": "/redfish/v1/Chassis/Rack1/Thermal#/Redundancy/0",
            "MemberId": "0",
            "Name": "BaseBoard System Fans",
            "RedundancyEnabled": false,
            "RedundancySet": [
                { "@odata.id": "/redfish/v1/Chassis/1/Thermal#/Fans/0" }
            ],
            "Mode": "N+m",
            "Status": {
                "State": "Disabled",
                "Health": "OK"
            },
            "MinNumNeeded": 1,
            "MaxNumSupported": 2
        }
    ]
}
```

4.18.1.2 PUT

Operation is not allowed on this resource.

4.18.1.3 PATCH

Operation is not allowed on this resource.

4.18.1.4 POST

Operation is not allowed on this resource.

4.18.1.5 DELETE

Operation is not allowed on this resource.

§

5 Common Property Description

5.1 Status

Table 19. Status Attributes

Attribute	Type	Nullable	Description
`State`	String	Yes	This indicates the known state of the resource, such as if it is enabled. Refer to status ->State.
`Health`	String	Yes	This represents the health state of this resource in the absence of its dependent resources. Allowed values: Refer to Section 5.3 for allowed values.
`HealthRollup`	String	Yes	This represents the overall health state from the view of this resource. Allowed values: Refer to Section 5.3 for allowed values.

5.2 Status -> State

- Enabled: This function or resource has been enabled.
- Disabled: This function or resource has been disabled.
- StandbyOffline: This function or resource is enabled, but awaiting an external action to activate it.
- StandbySpare: This function or resource is part of a redundancy set and is awaiting a failover or other external action to activate it.
- InTest: This function or resource is under test.
- Starting: This function or resource is starting.
- Absent: This function or resource is not installed.
- UnavailableOffline: This function or resource is present but cannot be used.
- Deferring: The element will not process any commands but will queue new requests.
- Quiesced: The element is enabled but only processes a restricted set of commands.
- Updating: The element is updating and may be unavailable or degraded.

5.3 Status -> Health

- OK: Normal.
- Warning: A condition exists that requires attention.
- Critical: A critical condition exists that requires immediate attention.

5.4 ComputerSystem.Reset

- On: Turn the system on.
- ForceOff: Turn the system off immediately (nongraceful) shutdown.
- GracefulRestart: Perform a graceful system shutdown followed by a restart of the system.
- ForceRestart: Perform an immediate (non-graceful) shutdown, followed by a restart of the system.
- Nmi: Generate a nonmaskable interrupt to cause an immediate system halt.
- ForceOn: Turn the system on immediately.
- PushPowerButton: Simulate the pressing of the physical power button on this system.
- GracefulShutdown: Perform a graceful system shutdown and power off.

5.5 BootSourceOverrideTarget/Supported

- None: Boot from the normal boot device.
- Pxe: Boot from the preboot execution (PXE) environment.
- Floppy: Boot from the floppy disk drive.
- Cd: Boot from the CD/DVD disc.
- USB: Boot from a USB device as specified by the system BIOS.
- Hdd: Boot from a hard drive.
- BiosSetup - Boot to the BIOS Setup Utility.
- Utilities: Boot the manufacturer's Utilities programs.
- Diags: Boot the manufacturer's Diagnostics program.
- UefiShell: Boot to the UEFI Shell.
- UefiTarget: Boot to the UEFI Device specified in the UefiTargetBootSourceOverride property.
- SDCard: Boot from an SD Card.
- UefiHttp: Boot from a UEFI HTTP network location.
- RemoteDrive: Boot from a remote drive (e.g., iSCSI).

§

6 Appendix

6.1 Creating new Composed Node – explanation

6.1.1 Creating Composed Node using JSON template

To create a Composed Node using the Pod Manager REST API, it is necessary to create a JSON template describing the requested resources. Supply the template to the Pod Manager by performing a HTTP POST request on the Composed Node Collection Action URI, located at "/redfish/v1/Nodes/Actions/Allocate" on the Pod Manager service.

The JSON template may contain various details of resources to be used in Composed Node. All JSON template elements are optional, but each requirement should be coherent itself. It is possible to supply Pod Manager with a JSON template containing no specific requirements (e.g., `{}` – a pair of empty curly braces in HTTP request body) thus allowing Pod Manager to propose a Composed Node containing resources chosen arbitrarily by Pod Manager.

6.1.2 Specifying requirements for a Composed Node

The JSON template contains requirements for a single Composed Node. Basic customization covers setting a "Name" and "Description" of such a system (both being of type *String*). As the "Name" parameter is required by Redfish for all resources, if it is not supported, Pod Manager uses the default name. The example below will allocate a single Composed Node with the requested name and description:

```
{
       "Name": "Customized Composed Node name",
       "Description": "Description of a customized Composed Node"
}
```

Note: The JSON template may contain requirements for processors, memory, remote drives, local drives, and Ethernet interfaces. To specify requirements for those resources, a proper section must appear in the JSON template.

6.1.3 General assumptions for allocation

Requirements are treated as minimal required values, so the resulting Composed Node may have better parameters than requested. The Composed Node customization and resource customization sections described below can be used jointly.

Each resource type description has an associated table containing details about specific requirements. **Key** is the JSON object field; **JSON type** contains a data type as defined by json.org; **Allowed values** contains additional restrictions to JSON type or hints (e.g., for enumerations or `Boolean` values); **Nullable** indicates whether a ***null*** value can be passed for a specified key; **Notes, limitations** provides additional hints about a specific requirement.

6.1.3.1 Location requirements

Processor, Memory, Local Drive, and Ethernet Interface sections may contain Resource and Chassis objects. Resource must contain the Pod Manager URI (presented as "@odata.id") of each discovered resource (processor's URI in Processor section, URI to memory resource in Memory section, and so on). Chassis must contain the Pod Manager URI of the discovered Chassis in which applicable resources will be looked for.

6.1.4 Specifying Processor requirements

The JSON template may contain requirements for multiple Processors. The example below specifies requirements for a single Processor to be used in Composed Node.

```
{
	"Processors": [{
		"Model": "Multi-Core Intel(R) Xeon(R) processor 7xxx Series",
		"TotalCores": 2,
		"AchievableSpeedMHz": 3700,
		"InstructionSet": "x86-64",
		"Oem": {
			"Brand": "X7"",
			"Capabilities": [
				"sse"
			]
		},
		"Resource": {
			"@odata.id": "/redfish/v1/Systems/1/Processors/1"
		},
		"Chassis": {
			"@odata.id": "/redfish/v1/Chassis/1"
		},
		 "ProcessorType": "CPU"
	}]
}
```

Table 20. Processor Requirement Attributes

Attribute	Type	Allowed values	Nullable	Description
`Model`	String		Yes	String representing Processor model.
`TotalCores`	Number		Yes	Positive integer value expected
`AchievableSpeedMHz`	Number		Yes	Positive integer value expected
`InstructionSet`	String	"x86", "x86-64", "IA-64", "ARM-A32", "ARM-A64", "MIPS32", "MIPS64", "OEM"	Yes	One of allowed, enumerated values
`Oem`	Object		Yes	
`Oem` → `Brand`	String	"E3", "E5", "E7", "X3", "X5", "X7", "I3", "I5", "I7", "Unknown","Silver", "Gold", Platinum"	Yes	One of allowed, enumerated values
`Oem->Capabilities`	Array	CPU capabilities string	Yes	List of processor capabilities (like "sse")
`Resource`	Object	Exact location of a single Processor.	Yes	Refer to Section 6.1.3.1
`Chassis`	Object	Exact location of a single chassis.	Yes	Refer to Section 6.1.3.1

Attribute	Type	Allowed values	Nullable	Description
ProcessorType	String	“CPU”, “FPGA”, “GPU”, “DSP”, “Accelerator”, ”OEM”	Yes	One of allowed enumerated values

The template can also provide a requirement for the number of processor cores available in a composed node:

```
        "TotalSystemCoreCount": 32
```

Allocation assumptions:

- Which processors will meet supplied requirements?
  - located on the same computer system as other resources
  - with exact match on Model
  - with exact match on Brand
  - with at least TotalCores
  - with at least AchievableSpeedMHz
  - with exact match on InstructionSet
  - with exact match on ProcessorType
  - with superset of processor capabilities specified in the Capabilities array

If a computer system contains processors with cores numbering at least TotalSystemCoreCount, it will meet the requirements.

6.1.5 Specifying Memory requirements

The JSON template may contain requirements for multiple Memory Modules. The example below specifies requirements for a single Memory Module to be used in Composed Node.

```
{
        "Memory": [{
                "CapacityMiB": 16000,
                "MemoryDeviceType": "DDR3",
                "SpeedMHz": 1600,
                "Manufacturer": "Intel",
                "DataWidthBits": 64,
                "Resource": {
                        "@odata.id": "/redfish/v1/Systems/1/Memory/1"
                },
                "Chassis": {
                        "@odata.id": "/redfish/v1/Chassis/1"
                }
        }]
}
```

Table 21. Memory Requirement Attributes

Attribute	Type	Allowed values	Nullable	Notes, limitations
CapacityMiB	Number		Yes	Positive value expected

Attribute	Type	Allowed values	Nullable	Notes, limitations
MemoryDeviceType	String	"DDR", "DDR2", "DDR3", "DDR4", "DDR4_SDRAM", "DDR4E_SDRAM", "LPDDR4_SDRAM", "DDR3_SDRAM", "LPDDR3_SDRAM", "DDR2_SDRAM", "DDR2_SDRAM_FB_DIMM", "DDR2_SDRAM_FB_DIMM_PROBE", "DDR_SGRAM", "DDR_SDRAM", "ROM", "SDRAM", "EDO", "FastPageMode", "PipelinedNibble"	Yes	One of allowed, enumerated values
SpeedMHz	Number		Yes	Positive integer value expected
Manufacturer	String		Yes	String representing Memory Module manufacturer name
DataWidthBits	Number		Yes	Positive integer value expected.
Resource	Object	Exact location of a single Memory Module.	Yes	Refer to Section 6.1.3.1
Chassis	Object	Exact location of a single chassis.	Yes	Refer to Section 6.1.3.1

The template can also provide requirement for total memory available in composed node, without dividing it into memory modules:

```
        "TotalSystemMemoryMiB": 32000
```

Allocation assumptions:

- Which Memory Modules will meet supplied requirements?
  - with at least CapacityMiB
  - located on the same computer system as other resources
  - with exact match on MemoryDeviceType
  - with at least SpeedMHz
  - with exact match on Manufacturer
  - with at least DataWidthBits
- If a computer system contains Memory Modules of a total size of at least TotalSystemMemory, it will meet the requirements.

6.1.6 Specifying Remote Drive requirements

The JSON template may contain requirements for multiple Remote Drives, but currently only one set of requirements is supported. The example below specifies requirements for a single Remote Drive to be used in Composed Node.

```
{
        "RemoteDrives": [{
                "CapacityGiB": 80,
                "iSCSIAddress": "iqn.oem.com:fedora21",
                "Master": {
                        "Type": "Snapshot",
                        "Resource": {
                                "@odata.id":
"/redfish/v1/Services/1/LogicalDrives/1"
                        }
                }
        }]
}
```

Table 22. Remote Drive Requirement Attributes

Attribute	Type	Allowed values	Nullable	Description
`CapacityGiB`	Number		Yes	Positive value expected, required if Master Drive supplied. Should be at least the size of Logical Drive used as Master Drive.
`iSCSIAddress`	String		No	Required. Defines `TargetIQN` of `RemoteTarget`. When no Master Drive supplied, defines IQN of an existing target. Otherwise defines IQN to be set for new Remote Target (should be unique in Pod Manager).
`Master`	Object		Yes	
`Master → Type`	String	"Snapshot", "Clone"	No	One of the allowed, enumerated values. Required if Master Drive supplied
`Master → Address`	Object		No	Pod Manager URI of discovered Logical Volume. Required if Master Drive supplied.

6.1.6.1 Using existing Remote Drive

To use an existing Drive, it is necessary to

- Set `iSCSIAddress` to `TargetIQN` of existing target,
- Not provide Master, or set it to null

```
{
	"RemoteDrives": [{
		"iSCSIAddress": "iqn.oem.com:fedora21"
	}]
}
```

6.1.6.2 Using a Master Drive for fresh Remote Drive creation

To use a fresh Drive created from the Master Drive, it is necessary to

- Set `CapacityGiB` to define capacity of the new Remote Drive that is at least of Master Drive's size
- Set Address to an IQN that is unique in Pod Manager
- Set Master → Type to "Snapshot" or "Clone"
- Set Master → Resource to a valid Pod Manager URI of the Logical Drive to be used as the source Drive

```
{
	"RemoteDrives": [{
		"CapacityGiB": 80,
		"iSCSIAddress": "iqn.oem.com:fedora21",
		"Master": {
			"Type": "Snapshot",
			"Resource": {
				"@odata.id":
"/redfish/v1/Services/1/LogicalDrives/1"
			}
		}
	}]
}
```

6.1.7 Specifying Local Drive requirements

The JSON template may contain requirements for multiple Local Drives (represented by Drive resources). The example below specifies requirements for a single Local Drive to be used in Composed Node.

```
{
	"LocalDrives": [{
		"CapacityGiB": 100,
		"Type": "HDD",
		"MinRPM": 5400,
		"SerialNumber": "12345678",
		"Interface": "SATA",
		"Resource": {
			"@odata.id": "redfish/v1/Chassis/Blade1/Drives/Disk1"
		},
		"Chassis": {
			"@odata.id": "/redfish/v1/Chassis/Blade1"
		},
		"FabricSwitch": false
	}]
}
```

Table 23. Local Drive Requirement Attributes

Attribute	Type	Allowed values	Nullable	Description
`CapacityGiB`	Number		Yes	Positive value expected
`Type`	String	"HDD", "SSD"	Yes	One of allowed, enumerated values
`MinRPM`	Number		Yes	Positive integer value expected
`SerialNumber`	String		Yes	
`Interface`	String	“SAS”, “SATA”, “NVMe”	Yes	One of allowed, enumerated values
`Resource`	Object	Exact location of a single Device.	Yes	Refer to Section 6.1.3.1
`Chassis`	Object	Exact location of a single Chassis.	Yes	Refer to Section 6.1.3.1
`FabricSwitch`	Boolean	True, false	Yes	Determines if drive should be connected using fabric switch (PNC) or directly attached to computer system.

Allocation assumptions:

- Which Local Drives will meet supplied requirements?
  - located on the same computer system as other resources
  - with at least `CapacityGiB`
  - with exact match on Type
  - with at least `MinRPM`
  - with exact `SerialNumber`
  - with exact Interface

6.1.7.1 Pooled NVMe Cotroller (PNC) drives

If PNC is available in POD Manager, and there is no system fulfilling Local Drive requirements, PNC drives will be attached to the Composed Node from the pool of available PNC drives.

Note: PNC drives that were detached from a Composed Node resource (via a Detach action or DELETE operation on the node) and have the property "`EraseOnDetach`" set to `false` (or to `null`), won't be erased. Their property, "`DriveErased`", won't change to `true`, and because of that won't be available in the pool of PNC drives ready for node composition. Those drives need to be selected via Resource property, or erased by action SecureErase on the drive resource.

Drives with the property "`DriveErased`" set to `true` or `null` are available for composition without the need to specify their URI in the Resource property.

The following example shows a request that will allocate a node with a PNC drive:

```
{
       "LocalDrives": [{
              "CapacityGiB": 100,
              "Type": "SSD",
              "Interface": "NVMe",
              "Chassis": {
                     "@odata.id": "/redfish/v1/Chassis/PCIeSwitchChassis"
              }
       }]
}
```

After node allocation and assembly (in "Assembled" and "Failed" `ComposedNodeState`), the user is able to attach and remove PNC devices (drives) using `AttachEndpoint` and `DetachEndpoint` actions.

This example shows a request that will attach a drive to an existing node:

```
POST /redfish/v1/Nodes/1/Actions/ComposedNode.AttachEndpoint
Content-Type: application/json
{
    "Resource": {
       "@odata.id":"/redfish/v1/Chassis/PCIeSwitchChassis/Drives/Disk.Bay.2"
    }
}
```

This example shows a request to remove a drive from an existing node:

```
POST /redfish/v1/Nodes/1/Actions/ComposedNode.DetachEndpoint
Content-Type: application/json
{
    "Resource": {
        "@odata.id":"/redfish/v1/Chassis/PCIeSwitchChassis/Drives/Disk.Bay.3"
    }
}
```

6.1.8 Specifying Ethernet Interface requirements

The JSON template may contain requirements for multiple Ethernet Interfaces .The example below specifies requirements for a single Ethernet Interface to be used in Composed Node.

```
{
       "EthernetInterfaces": [{
                "SpeedMbps": 1000,
                "PrimaryVLAN": 100,
                "VLANs": [{
                      "VLANId": 100,
                      "Tagged": false
                }],
                "Resource": {
                      "@odata.id":
"/redfish/v1/Systems/1/EthernetInterfaces/1"
                },
                "Chassis": {
                      "@odata.id": "/redfish/v1/Chassis/1"
                }
       }]
}
```

Table 24. Ethernet Interface Requirement Attributes

Attribute	Type	Allowed values	Nullable	Description
`SpeedMbps`	Number		Yes	Positive integer value expected
`PrimaryVLAN`	Number		Yes	Positive integer value expected
`VLANs`	Array[Object]		Yes	Null value will be interpreted as absence of this key. Empty array [ ] will clear all existing vlans, excluding Reserved VLANs.
`VLANs` → `VLANId`	Number		No	Positive integer value expected
`VLANs` → `Tagged`	Boolean	true, false	No	Boolean value
`Resource`	Object	Exact location of a single Ethernet Interface.	Yes	Refer to Section 6.1.3.1
`Chassis`	Object	Exact location of a single Chassis.	Yes	Refer to Section 6.1.3.1

Allocation assumptions:

- Which Ethernet Interfaces will meet supplied requirements?
  - located on the same Computer System as other resources
  - with at least `SpeedMbps`
  - ones that are connected with `SwitchPorts` (when VLANs section is provided)

6.1.8.1 Reserved VLANs

There is a possibility to restrict usage of some VLANs by changing the configuration file located in `/etc/pod-manager/allocation.json`.

An example file looks like this:

```
{
  "ReservedVlanIds": [1, 170, 4088, 4091, 4094]
}
```

where 1, 170, 4088, 4091, and 4094 are VLANs which are reserved. Reserved VLANs have following implications:

- Allocation JSON cannot contain such VLANs and such requests result in an error
- Reserved VLANs are not deleted during allocation
- Reserved VLANs are not deleted during disassembly

6.1.9 Specifying Security requirements

The JSON template may contain requirements for security of the Composed Node. The example below specifies requirements for a Trusted Platform Module version 2.0 present in Composed Node.

```
{
       "Security": {
         "TpmPresent": true,
         "TpmInterfaceType": “TPM2_0”
       }
}
```

Table 25. Security Requirement Attributes

Attribute	Type	Allowed values	Nullable	Description
`TpmPresent`	Boolean		Yes	Boolean expected
`TpmInterfaceType`	String		Yes	String value (enum) expected. Possible values for this field are defined in metadata file `ComputerSystem.xml` as a `InterfaceType`
`TxtEnabled`	Boolean		Yes	Boolean expected

Pod Manager will allocate a Composed Node that meets the security requirements listed in the Security section.

- `TpmPresent` – Determines whether the Composed Node should be equipped with a TPM module.
- `TpmInterfaceType` – Overrides the `TpmPresent` parameter (if specified TPM module expected). The system must be equipped with defined TPM interface types only.
- `TxtEnabled` – Determines whether Composed Node should have Intel® Trusted Execution Technology (Intel® TXT) mode enabled.

6.1.10 Allocation algorithm

Node composition starts with an HTTP POST request of the JSON template on the `/redfish/v1/Nodes/Actions/Allocate` Composed Node Collection Action URI on Pod Manager Service. If the JSON template is well-formed, and contains a supported set of requirements, the allocation process starts. Four major scenarios are currently supported:

- Allocating resources for Composed Node to be booted from Local Drive.
- Allocating resources for Composed Node to be booted from existing Remote Drive.
- Allocating resources for Composed Node to be booted from Remote Drive that need to be created.
- Allocating resources for Composed Node with VLAN requirements specified. This scenario is used with one of the other three.

The allocation process is preceded by a general verification of the JSON template that checks whether the requested node can be realized by available resources, and consists of the following:

- Selecting and allocating a computer system that contains resources matching template requirements for processors, memory, local drives and Ethernet interfaces.
- Selecting or creating a remote drive to be used with a previously-selected computer system, and allocating it.

6.1.10.1 Detailed process of selecting and allocating a Computer System for a Composed Node

- Find all Computer Systems that are not yet allocated (not used by any other allocated Composed Node) with Status Enabled and Health OK.
- Filter Computer Systems by specified Resource and Chassis (if supplied in template)
- Filter Computer Systems by Processors: Return all Computer Systems that contain at least requested quantity of Processors that meet requirements (if supplied in template):
  - Exactly matching requested model,
  - Exactly matching requested brand,
  - With at least requested number of cores,
  - With at least requested frequency,
  - Exactly matching requested instruction set.
- Filter Computer Systems by Memory: Return all Computer Systems with at least total requested size of memory located on Memory Modules that each of them meet requirements (if supplied in template):
  - Memory of exactly requested dimm device type
  - With at least requested speed MHz
  - With exact requested manufacturer
  - With at least requested data width bits
- Filter Computer Systems by Local Drives: Return all Computer Systems that contain for each requested Drive one distinct Device meeting requirements (if supplied in template):
  - With at least requested capacity specified
  - Exactly matching requested Drive type
  - With at least requested min RPM
  - With exact requested serial number
  - With exact Interface

- Filter Computer Systems by Ethernet Interfaces: Return all Computer Systems that contain for each requested Ethernet Interface one distinct Ethernet Interface meeting requirements (if supplied in template):
  - With at least requested speed.
  - If the VLANs section is provided, then Computer Systems with Ethernet Interfaces which are not connected with `EthernetSwitchPorts` are filtered out (as described below).
- A first Computer System from the resulting filtered collection is then allocated to be used in Composed Node.

6.1.10.2 Connection between Computer System's EthernetInterface and EthernetSwitchPort

In order to enable particular VLAN usage on a Composed Node, there is a need to map the Ethernet Switch Port and the Computer System's Ethernet Interface. This mapping is done using a MAC address as an identifier. The following fields are used for this mapping:

- `NeighborMAC` on `EthernetSwitchPort` resource
- `MacAddress` on `EthernetInterface` resource

If those two properties contain the same value, Computer System's Ethernet Interface and Ethernet Switch Port are treated as connected. Only Computer Systems with Ethernet Interfaces that are connected to Ethernet Switch Ports could be used in allocation with the specified VLAN requirement.

6.1.10.3 Detailed process of selecting Remote Drives

- Determine what type of Remote Drive is requested,
- When requesting existing Remote Drive:
  - Find all Targets that are not yet allocated (not used by any allocated Composed Node).
  - Find first Target that exactly matches requested IQN and allocate it to be used in Composed Node.
- When requesting a new Remote Drive,
  - Check if Target does not exist with requested IQN to be set for newly created target.
  - Check if Logical Drive requested as Master Drive exists on Storage Service handled by Pod Manager, and select this Storage Service to handle new Target creation.
  - Find all Logical Volume Groups meeting requirements:
    - Located on selected Storage Service
    - Having free space of at least requested capacity for a new Remote Drive
  - A first Logical Volume Group from the resulting filtered collection is selected as a placement for new Logical Volume, which will be exposed as a new Target (Remote Drive).
  - A new Logical Volume is created on the selected Logical Volume Group (as a snapshot or as a clone).
  - A new Target is created on top of newly created Logical Volume.
  - Newly created Target is allocated to be used in Composed Node.

6.1.10.4 Post-allocation scenarios

A Composed Node is created as a new REST resource at `/redfish/v1/Nodes/{`*`NodeId`*`}` when a proper Computer System is found and successfully allocated. The state of the Composed Node is set to `Allocated`. An `Allocated` Composed Node is a Pod Manager proposition that can be either accepted or rejected.

- If accepted, the user has to send a HTTP POST request on the `ComposedNode.Assemble` action of the proposed Composed Node to assemble it.
  - If no Remote Drive was requested, a Composed Node's state is set to `Assembled`.
  - When a Remote Drive is requested, the Composed Node remains `Assembling` until Target creation finishes. When the Target is successfully assembled to be used with the Composed Node, the node's state is set to `Assembled`.

  - The assembly process doesn't end with sending a power-on request, so it's necessary to perform a `ComposedNode.Reset` action to power on a Composed Node after assembly.
- If rejected, the user can continue sending HTTP POST requests of the JSON template on `/redfish/v1/Nodes/Actions/Allocate` to create more proposals to pick from. When finding the right pick, it is recommended to send HTTP DELETE on all rejected proposals of Composed Nodes to free the resources allocated by them.

§