OCP Baseline HW Mgmt Requirements Revison 1.1

Version: 1.0.0

Date: 2023-12-04

Document Status: Draft

Document Language: en-US

CONTENTS

1 Scope

This document contains requirements and provide usage examples for the OCP Baseline Hardware Management API v1.1. The baseline hardware management tasks are a common set of manageability from which OCP platforms can extend.

2 Requirements

As a Redfish-based interface, the required Redfish interface model elements are specified in a corresponding profile document. For the Baseline Hardware Management API v1.1 [4].

2.1 Validating conformance

The profile file can be read by the open-source conformance test, Redfish Interop Validator [3]. The validator will auto-generate tests, execute the tests against an implementation, and create a test report.

The following command executes the validator.

```
$> python RedfishInteropValidator.py <profileName> \--ip <host:port>
```

3 Management Tasks

The following table lists the baseline management tasks.

Use Case	Management Task	Requirement
Account Management	Get accounts	Mandatory
Service Management	Get sessions	Mandatory
Hardware Inventory	Get FRU info	Mandatory
	Get and Set the Asset Tag	Recommended
Hardware Location	Get location LED	Recommended
	Set location LED	Recommended
Status	Get Chassis status	Mandatory
Power	Get power state	If implemented, mandatory
	Get power usage	Recommended
	Get power limit	Recommended
Temperature	Get temperature	If implemented, mandatory
	Get temperature thresholds	If implemented, recommended
Cooling	Get fan speeds	If implemented, mandatory
	Get fan redundancy	If implemented, recommended
Log	Get log entry	Mandatory
	Clear system log	Recommended
Management Controller	Get firmware version	Mandatory
	Get controller status	Mandatory
	Get network info	Mandatory
	Reset controller	Mandatory

4 Use Cases

This section describes how each task is accomplished by interacting with the Redfish Interface.

4.1 Redfish model notes

Some aspects of the Redfish model should be comprehend before the interaction with the Redfish model shown below is understood.

Redfish models a managed node in terms of its physical, logical and manager aspects:

- The physical aspect is modeled via the Chassis resource
- The logical aspect is modeled via the ComputerSystem or Fabric resources
- The manager aspect is modeled via the Manager resource

The relationship between the above resources are specified by the Links property.

- On the Chassis resource, see the Links.ComputerSystems and Links.Storage, Links.Switches, and Links.ManagedBy array properties
- On the ComputerSystem resource, see the Links.Chassis and Links.ManagedBy array properties
- On the Manager resource, see the Links.ManagerForChassis, Links.ManagerForServers, Links.ManagerForSwitches and Links.ManagerForManagers array properties.

4.2 Get accounts

The accounts on the management controller is obtained from the AccountService resource.

```
GET /redfish/v1/AccountService
```

The response message contains the following fragment.

```
{
    "Accounts": { "@odata.id": "/redfish/v1/AccountService/Accounts" },
    "Roles": { "@odata.id": "/redfish/v1/AccountService/Roles" }
}
```

The Roles property specifies the path to the Roles collection resource. The Redfish specification specifies the Admin, Operator and ReadOnly roles be member resource.

The Account resource represents each account on the management controller and the role associated to the account.

```
Get /redfish/v1/AccountService/Accounts/1
```

The response message contains the following fragment.

```
{
    "Name": "User Account",
    "Enabled": true,
    "Password": null,
    "PasswordChangeRequired": false,
    "UserName": "Administrator",
    "RoleId": "Administrator",
    "Locked": false,
    "Links": {
        "Role": { "@odata.id": "/redfish/v1/AccountService/Roles/Administrator" }
    }
}
```

4.3 Get sessions

The sessions on the management controller is obtained from the SessionService resource.

```
GET /redfish/v1/SessionService
```

The response message contains the following fragment.

```
{
    "ServiceEnabled": true,
    "SessionTimeout": 30,
    "Sessions": { "@odata.id": "/redfish/v1/SessionService/Sessions" }
}
```

The Sessions property specifies the path to the Sessions collection resource. The Redfish service creates a Session resource for each session that is established. The following is a fragment of a Session resource.

```
{
    "@odata.id": "/redfish/v1/SessionService/Sessions/1234567890ABCDEF",
    "Id": "1234567890ABCDEF",
    "Name": "User Session",
    "UserName": "Administrator"
}
```

4.4 Get FRU info

The hardware FRU (field replaceable unit) information is obtained from the Chassis resource

```
GET /redfish/v1/Chassis/Ch-1
```

The response message contains the following fragment. The response contains the hardware inventory properties for manufacturer, model, SKU, serial number, and part number. The AssetTag properties is a client writeable property.

```
{
    "Id": "Ch-1",
    "ChassisType": "Node",
    "Manufacturer": "Contoso"
    "Model": "RackScale_Rack",
    "SKU": "..."
    "SerialNumber": "...",
    "PartNumber": "...",
    "AssetTag": null,
    "IndicatorLED": "Off"
}
```

4.5 Set the asset tag

The value of the asset tag is set by patching to AssetTag property in the Chassis resource.

```
PATCH /redfish/v1/Chassis/Ch-1
```

The PATCH request includes the following message.

```
{
    "AssetTag": "989846353530048"
```

```
}
```

On successful completion, the response message contains the Chassis resource.

4.6 Get the location LED

The state of the location LED is obtained by retrieving the Chassis resource.

```
GET /redfish/v1/Chassis/Ch-1
```

The response message contains one of the following two fragments.

```
{
    "IndicatorLED": "Lit"
}
```

Or

```
{
    "LocationIndicatorActive": True
}
```

4.7 Set the location LED

The state of the location LED is set by setting the IndicatorLED or the LocationIndicatorActive property in the Chassis resource. The IndicatorLED property has been deprecated in favor of the LocationIndicatorActive property.

```
PATCH /redfish/v1/Chassis/Ch-1
```

The PATCH request includes one of the following two messages, which corresponds to the property returned in the GET request.

```
{
    "IndicatorLED": "Lit"
}
```

Or

```
{
    "LocationIndicatorActive": True
}
```

4.8 Get chassis status

The status and health of the chassis is obtained by retrieving the Chassis resource.

```
GET /redfish/v1/Chassis/Ch-1
```

The response message contains the following fragment. The Status property contains the State and Health properties.

```
{
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    }
}
```

4.9 Get power state

The power state of the chassis is obtained from the Chassis resource.

```
GET /redfish/v1/Chassis/Ch-1
```

The response message contains the following fragment. The response contains the PowerState property.

```
{
    "PowerState": "On"
}
```

4.10 Get power usage

The power usage can obtained via the PowerSubsystem or Power resource, both are subordinate to the Chassis resource. The Power resource has been deprecated in favor of the PowerSubsystem resource.

The power usage is obtained via the PowerSubsystem resource by retrieving the EnvironmentMetrics resource.

```
GET /redfish/v1/Chassis/Ch-1/PowerSubsystem/EnvironmentMetrics
```

The response message contains the following fragment.

```
{
    "PowerWatts": "215"
}
```

The power usage is obtained via the Power resource by retrieving the Power resource.

```
GET /redfish/v1/Chassis/Ch-1/Power
```

The response message contains the following fragment. The PowerControl property contains the PowerConsumedWatts PowerCapacityWatts properties.

```
{
    "PowerControl": {
        "PowerConsumedWatts": "215"
    }
}
```

4.11 Get power limit

The power limit can be obtained via the PowerSubsystem or Power resource, both are subordinate to the Chassis resource. The Power resource has been deprecated in favor of the PowerSubsystem resource.

The power usage is obtained via the PowerSubsystem resource by retrieving the EnvironmentMetrics resource.

```
GET /redfish/v1/Chassis/Ch-1/PowerSubsystem/EnvironmentMetrics
```

The response message contains the following fragment.

```
{
    "PowerLimitWatts": "220"
}
```

The power limit is obtained via the Power resource by retrieving the Power resource.

```
GET /redfish/v1/Chassis/Ch-1/Power
```

The response message contains the following fragment. The PowerControl property contains the LimitInWatts and LimitException properties.

```
{
    "PowerControl": {
        "PowerLimit": {
            "LimitInWatts": "220"
        }
    }
}
```

4.12 Get temperature

The temperature can be obtained via the ThermalSubsystem or Thermal resource, both are subordinate to the Chassis resource. The Thermal resource has been deprecated in favor of the ThermalSubsystem resource.

The temperature is obtained via the ThermalSubsystem resource by retrieving the ThermalSubsystem resource.

```
GET /redfish/v1/Chassis/Ch-1/ThermalSubsystem
```

The response message contains the following fragment.

```
{
    "ThermalMetric": {
```

```
        "TemperatureSummayCelsius": {
            "Exhaust": 21
        }
    }
}
```

The temperature is be obtained via the Thermal resource by retrieving the Thermal resource.

```
GET /redfish/v1/Chassis/Ch-1/Thermal
```

The response message contains the following fragment. Temperatures is an array property. Each element of the array is identified by a Name property. The ReadingCelsius property contains the temperature.

```
{
    "Temperatures": [
        {
            "Name": "Chassis Exhaust Temp",
            "ReadingCelsius": 21
        }
    ]
}
```

4.13 Get temperature thresholds

The temperature thresholds are obtained from the Thermal resource which is subordinate to Chassis resource.

```
GET /redfish/v1/Chassis/Chassis_1/Thermal
```

The response message contains the following fragment. The threshold properties are optional.

```
{
    "Temperatures": [
        {
            "Name": "Chassis Exhaust Temp",
            "UpperThresholdFatal": "45",
            "UpperThresholdCritical": "40",
            "UpperThresholdNonCritical", "35"
        }
    ]
```

```
}
```

4.14 Get fan speeds

The fan speeds can be obtained via the ThermalSubsystem or Thermal resource, both are subordinate to the Chassis resource. The Thermal resource has been deprecated in favor of the ThermalSubsystem resource.

The fan speeds are obtained via the ThermalSubsystem resource by retrieving the Fans collection resource.

```
GET /redfish/v1/Chassis/Ch-1/ThermalSubsystem/Fans
```

The response message contains the following fragment.

```
{
    "Members@odata.count": 2,
    "Members": {
        {
            "@odata.id"://redfish/v1/Chassis/Ch-1/ThermalSubsystem/Fans/Bay1
        },
        {
            "@odata.id"://redfish/v1/Chassis/Ch-1/ThermalSubsystem/Fans/CPU1
        }
    }
}
```

The fan speed of a specific fan is obtained by retrieving the Fan resource

```
GET /redfish/v1/Chassis/Ch-1/ThermalSubsystem/Fans/Bay1
```

The response message contains the following fragment.

```
{
    "SpeedPercent": {
        "Reading": 45,
        "SpeedRPM": 2200"
    }
}
```

The fan speeds can be obtained from the Thermal resource by retrieving the Thermal resource.

```
GET /redfish/v1/Chassis/Ch-1/Thermal
```

The response message contains the following fragment. Within the Fans array property, each array member has a Reading and ReadingUnits property.

```
{
    "Fans": [
        {
            "Name": ""
            "Reading": 300
            "ReadingUnits": "RPM"
        }
    ]
}
```

4.15 Get fan redundancy

Fans which are configured in a redundancy set should be available via the resource model. The fan speeds can be obtained via the ThermalSubsystem or Thermal resource, both are subordinate to the Chassis resource. The Thermal resource has been deprecated in favor of the ThermalSubsystem resource.

The fan redundancy can be obtained via the ThermalSubsystem resource by retrieving the Fans collection resource.

```
GET /redfish/v1/Chassis/Ch-1/ThermalSubsystem/Fans
```

The response message contains the following fragment.

```
{
    "FanRedundancy": [
        {
            "RedundancyType: "NPlusM",
            "RedundancyGroup": {
                {
                    "@odata.id"://redfish/v1/Chassis/Ch-1/ThermalSubsystem/Fans/Bay1
                },
                {
                    "@odata.id"://redfish/v1/Chassis/Ch-1/ThermalSubsystem/Fans/Bay2
                }
            }
        }
    ]
```

```
}
```

The fans redundancy can be obtained from Thermal resource.

```
GET /redfish/v1/Chassis/Ch-1/Thermal
```

The response message contains the following fragment. The Redundancy array property contains as list of the redundancies. The redundancy contains a RedundancySet property which contains the members of the redundancy set.

```
{
    "Fans": [
        {
            "@odata.id": "/redfish/v1/Chassis/Ch-1/Thermal#/Fans/0",
            "MemberId": "0",
            "Reading": 300
            "ReadingUnits": "RPM"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/Ch-1/Thermal#/Fans/1",
            "MemberId": "1",
            "Reading": 300
            "ReadingUnits": "RPM"
        }
    ],
    "Redundancy": [
        {
            "@odata.id": "/redfish/v1/Chassis/Ch-1/Thermal#/Redundancy/0",
            "MemberId": "0",
            "RedundancySet": \[
                { "@odata.id": "/redfish/v1/Chassis/Ch-1/Thermal#/Fans/0" },
                { "@odata.id": "/redfish/v1/Chassis/Ch-1/Thermal#/Fans/1" }
            ],
            "Mode": "N+m",
            "Status": {
                "State": "Disabled",
                "Health": "OK"
            },
            "MinNumNeeded": 1,
            "MaxNumSupported": 2
        }
    ]
}
```

4.16 Get system log

The System's log is retrieved is obtained by retrieving the Log resource which represent the system log.

```
GET /redfish/v1/Systems/CS-1/LogService/Log
```

The response message contains the following fragment. The value of the Entries property is the collection resource for the entries in the log.

```
{
    "Name": "System Log",
    "Entries": {
        "@odata.id": "/redfish/v1/Systems/CS-1/LogServices/Log/Entries"
    }
}
```

The client can get each entry of the log.

```
GET /redfish/v1/Systems/CS-1/LogService/Log/Entries/1
```

The following fragment is

```
{
    "EntryType": "SEL",
    "Severity": "Critical",
    "Created": "2012-03-07T14:44:00Z",
    "Message": "Temperature threshold exceeded",
}
```

4.17 Clear system log

The System's log is retrieved is obtained by retrieving the Log resource which represent the node's log.

```
POST /redfish/v1/Systems/CS-1/LogService/Log/Actions/LogService.ClearLog
```

4.18 Get firmware version

The version of firmware for the management controller is obtained by retrieving the Manager resource which represents the management controller of interest.

```
GET /redfish/v1/Managers/BMC_1
```

The response contains the following fragment. The FirmwareVersion property contains the firmware version of the BMC.

```
{
    "FirmwareVersion": "1.00"
}
```

4.19 Get controller status

The status of the management controller is obtained by retrieving the Manager resource.

```
GET /redfish/v1/Managers/BMC
```

The response message contains the following fragment. The Status property contains the State and Health properties of the manager.

```
{
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    }
}
```

4.20 Get network info

The network information for the management controller is obtained by retrieving the EthernetInterface resource.

```
GET /redfish/v1/Managers/BMC/EthernetInterface
```

The response message contains the following fragment.

```
{
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    }.
    "MacAddress": "1E:C3:DE:6F:1E:24",
    "SpeedMbps": 100,
    "InterfaceEnabled": true,
    "LinkStatus": "LinkUp"
    "HostName": "MyHostName",
    "FQDN": "MyHostName.MyDomainName.com",
    "IPv4Addresses": [
        {
            "Address": "192.168.0.10",
            "SubnetMask": "255.255.252.0",
            "AddressOrigin": "DHCP",
            "Gateway": "192.168.0.1"
        }
    ],
    "NameServers": [
        "192.168.200.10",
        "192.168.150.1",
        "fc00:1234:100::2500"
    ]
}
```

The response message may also contain properties from the following fragment.

```
{
    "StaticNameServers": [
        "192.168.150.1",
        "fc00:1234:200:2500"
    ],
    "DHCPv4": {
        "DHCPEnabled": true,
        "UseDNSServers": true,
        "UseGateway": true,
        "UseNTPServers": false,
        "UseStaticRoutes": true,
        "UseDomainName": true,
        "FallbackAddress": "Static"
    },
```

```
    "IPv4StaticAddresses: [
        {
            "Address": "192.168.88.130",
            "SubnetMask": "255.255.0.0",
            "Gateway": "192.168.0.1"
        }
    ],
    "DHCPv6": {
        "OperatingMode": "Stateful",
        "UseDNSServers": true,
        "UseDomainName": false,
        "UseNTPServers": false,
        "UseRapidCommit": false
    },
    "IPv6Addresses": [
        {
            "Address": "2001:1:3:5::100",
            "PrefixLength": 64,
            "AddressOrigin": "DHCPv6",
            "AddressState": "Preferred"
        }
    ],
    "IPv6AddressPolicyTable": [
        {
            "Prefix": "::1/128",
            "Precedence": 50,
            "Label": 0
        }
    ],
    "IPv6StaticAddresses: [
        {
            "Address": "fc00:1234::a:b:c:d",
            "PrefixLength": 64
        }
    ],
    "IPv6StaticDefaultGateways": [
        {
            "Address": "fe80::fe15:b4ff:fe97:90cd",
            "PrefixLength": 64
        }
    ]
}
```

4.21 Reset controller

The management controller is reset by performing a POST action.

```
POST /redfish/v1/Manager/BMC/Actions/Manager.Reset
```

The POST request includes the following message. The ResetType property contains type of reset to perform.

```
{
    "ResetType": "ForceRestart"
}
```

5 References

[1] "Redfish API Specification"

[2] "Redfish Data Model Specification"

[3] "Redfish Interop Validator"

[4] "Baseline v1.1 Profile"

6 Revision

Revision	Date	Description
1.0	6/15/2021	Initial Baseline usage guide and profile contribution
1.1	TBD	Allow either PowerSubsystem or Power resource
		Allow either ThermalSubsystem or Thermal resource
		Require the StaticNameServers to be writable
		Require power limit to be writeable