# 

# Edgecore’s Device Management

V2.10

Author: Edgecore

# Table of Contents

[Table of Contents 2](#_Toc51522980)

[1. License 3](#_Toc51522981)

[2. Scope & Overview 4](#_Toc51522982)

[3. Architecture 5](#_Toc51522983)

[4. API Details 7](#_Toc51522984)

[4.1 Authentication 7](#_Toc51522985)

[4.1.1 Enable/Disable Session Management 7](#_Toc51522986)

[4.1.2 Create a New Session 7](#_Toc51522987)

[4.1.3 Delete an Existing Session 8](#_Toc51522988)

[4.2 Account Service 8](#_Toc51522989)

[4.2.1 View Accounts 8](#_Toc51522990)

[4.2.2 Create a New Account 9](#_Toc51522991)

[4.2.3 Delete an Account 9](#_Toc51522992)

[4.3 Event Service 10](#_Toc51522993)

[4.3.1 Add Subscriptions 10](#_Toc51522994)

[4.3.2 View Subscriptions 11](#_Toc51522995)

[4.3.3 Delete Subscriptions 11](#_Toc51522996)

[4.4 Hardware Information 12](#_Toc51522997)

[4.4.1 Chassis information 12](#_Toc51522998)

[4.4.2 Thermal Sensor Information 13](#_Toc51522999)

[4.4.3 PSU Power Information 14](#_Toc51523000)

[4.4.4 QSFP/XFP Information 15](#_Toc51523001)

[4.4.4.1 Collection of Ports 15](#_Toc51523002)

[4.4.4.2 Port Information 15](#_Toc51523003)

[4.5 Enable/Disable Log Service 16](#_Toc51523004)

[4.5.1 Enable/Disable Log Service 16](#_Toc51523005)

[4.5.2 Get Log Entries of the Target Device 17](#_Toc51523006)

[4.5.3 Clear Log Entries of the Target Device 17](#_Toc51523007)

[4.6 System Management 17](#_Toc51523008)

[4.6.1 System Reset 17](#_Toc51523009)

[4.6.2 Configure Boot Source 18](#_Toc51523010)

[4.6.3. Configure Default OS in GRUB 18](#_Toc51523023)

[4.6.4. Update System Firmware 19](#_Toc51523024)

[4.6.5. Check the System Firmware reflash status 19](#_Toc51523025)

[4.6.6. Update NOS Image 20](#_Toc51523026)

[4.6.7. Check the NOS Update status 20](#_Toc51523027)

[5. Potential Future Enhancements 21](#_Toc51523028)

[5.1 In-band IP Management 21](#_Toc51523029)

# License

Contributions to this proposal are made under the terms and conditions set forth in Open Compute Project Contribution License Agreement (“OCP CLA”) (“Contribution License”) by:

Edgecore Networks

Usage of this Specification is governed by the terms and conditions set forth in Open Compute Project Hardware License – Permissive (“OCPHL Permissive.)

Note: The following clarifications, which distinguish technology licensed in the Contribution License and/or Specification License from those technologies merely referenced (but not licensed), were accepted by the Incubation Committee of the OCP:

[None].

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Scope & Overview

Scope:

Edgecore’s Device Management can provide a method for identification and management of target devices. It can also provide data that can be used to analyze and detect possible future device health or out of state issues. This enables the entire infrastructure devices (including networking, storage, server) to be monitored to provide High Availability.

The work, described in this paper, provides means to manage devices though a well-defined data model. It uses the Edgecore’s Device Manager to collect and manage device status from all devices, and sends to an Output Bus, using RESTFUL APIS. The functionality also provides support for monitoring certain device states and sending event notifications.

All APIs and functionality set forth in this document will be available on both CPU and BMC sides, unless otherwise noted. The document also introduces Edgecore BMC daughter-card and motherboard reference design providing support to the new features, which is be discussed in detail separately. At the current time, the proposed changes are only planned for vOLT devices.

Overview:

Scalability is one of the paramount problems in today’s modern infrastructures. The growing needs of the end customers forces the infra operators to employ horizontal, scale-out solutions, often requiring them to deploy additional devices to their systems. This ever-growing infrastructure requires non-traditional thinking in which a more automated device management functionality must be employed.

Designed to manage large systems through a modern platform, DMTF’s Redfish is an open industry standard specification enabling customers to monitor their infrastructure through a well-defined interface.

In its core, each device is monitored and managed through an agent, PSME running on OLT (or OpenBMC running on BMC daughter card, if available), to

1. retrieve specific parameters to be sent through its northbound interfaces.
2. receive notifications on certain device state changes.[[1]](#footnote-1)
3. manage device state and reboot options.

Edgecore’s Device Manager uses the ONLP driver (or BMC when available) to manage and retrieve information about device, and makes data available at a configurable Output Bus. It uses a RESTFUL API to communicate with PSME at certain intervals to retrieve the data .

Other devices on the system, such as Aggregation switches and ONUs, can also use Redfish to report to Edgecore’s Device Manager, the same way vOLT does.

To provide uniform functionality across an infrastructure employing a non-uniform device list provided by multiple vendors, Redfish (and OCP) exchanges information using a hardware management specification, employing a JSON schema. This proposal uses standard OCP baseline profile. If needed, we may enhance the baseline profile with attributes specific for TelCo in the future.

This will also work on BMC controller running OpenBMC standard interface.

The following is an example of OCP defined schema

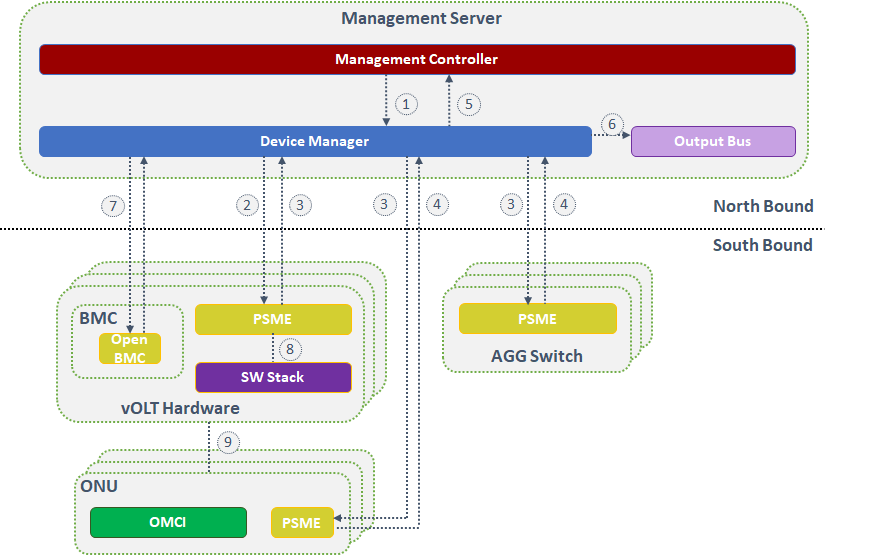
|  |
| --- |
| Redfish >> v1 >> Chassis  “@odata.type”: “#ChassisCollection.ChassisCollection”,  “Name”: “Chassis Collection”,  “Members@odata.count”: 2,  “Members”: [  {  “@odata.id”: “/redfish/v1/Chassis/Switch1”  } ,  {  “@odata.id”: “/redfish/v1/Chassis/Switch2”  }  ],  “odata.context”: “redfish/v1/$metadata#ChassisCollection.ChassisCollection”,  “odata.id”; “redfish/v1/Chassis” |

# Architecture

Edgecore’s Device Manager will use a standard interface and dataflow to manage and monitor for deployment in the field. Each device in the system will communicate North Bound with Device Manager to transmit device data and notifications, using PSME (or OpenBMC in BMC), adhering Redfish standards. Device Manager will collect device data and notifications from each device, and make the data available on a predetermined Output Bus for consumers. The consumers of this data could then use it for various purposes, including

* inventory management
* status monitoring
* device updates
* reboots

Following diagram illustrates the overall architecture and the interaction between the components at a high level.



① : The first step is the notification from Management Controller to add the IP address of the device to be monitored to the device list (Device Manager can also use Ref ⑤ to refresh all its data.) Device Manager can also subscribe to events from the device by sending a HTTP POST to the URL of the Resource Collection for "Subscriptions" in the Event Service. At this point, Device Manager can query status parameters with HTTP GET method to PSME (Pooled System Management Engine) periodically. The default polling interval is set to five seconds.

② : Device Manager is an alert Receiver. When the Redfish service interface is responding to Device Manager, it uses RESTful API. If PSME detects a change in one of the monitored hardware states, it will send an alert for the subscribed event to the Device Manager. A particular issue is reported only once until the state of the hardware changes.

③ : Device Manager retrieves data using HTTP GET method from devices periodically.

④ : Switch responds using HTTP RESPONSE to Device Manager.

⑤ : The Management Controller will provide the active device list to Device Manager which needs to be monitored. When the device becomes available, the Device Manager uses this IP address to connect PSME (or OpenBMC) using HTTP protocol.

⑥ : Device Manager publishes the device alarm/status into Output Bus.

⑦ : If exists, BMC can also be used to retrieve device status and manage the target device. However, BMC may not provide the full functionality offered by PSME, including periodic data collection or alarms.

⑧ : PSME inspects hardware status at regular intervals to detect hardware failure/status through Network OS’s SW stack.

# API Details

## Authentication

Redfish Service uses session management to implement authentication.

## Enable/Disable Session Management

Enable secure access to resources.

**POST**: /redfish/v1/SessionService

|  |  |  |
| --- | --- | --- |
| {  **"ServiceEnabled": true,**  **"SessionTimeout": 300**  }  - Response : 200 OK | | |
|  | | |
| Property | Requirement | Value |
| ServiceEnabled | Mandatory | true, false |
| SessionTimeout | Mandatory | ≥1 [in seconds] |

Following is a mockup of get the Session Service State

**GET**: /redfish/v1/SessionService

|  |
| --- |
| {  **"ServiceEnabled": true,**  **"SessionTimeout": 300**  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **SetSessionService** (DeviceAccount) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port.  **UserToken** (String): device user (Administrator) login permission code.  **SessionEnabled** (Bool): true(enable session service), false(disable session service).  **SessionTimeout** (uint64): What are the many timeouts for device session service? |
| gRPC API return | None |

Create a New Session

Create a new session (Login) by using the login credential of an existing account (default username "admin" and Password "redfish"), enable Session Management, first. Include the token is generated in the https request header to perform any operations on the secured resources.

**POST**: /redfish/v1/SessionService/Sessions

|  |  |  |
| --- | --- | --- |
| {  **"UserName": "admin",**  **"Password": "redfish"**  }  - Response : 201 CREATED | | |
|  | | |
| Property | Requirement | Value |
| UserName | Mandatory |  |
| Password | Mandatory |  |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **LoginDevice** (DeviceAccount) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port.  **ActUsername** (String): login this user name.  **ActPassword** (String): login this user password. |
| gRPC API return | DeviceAccount (message):  **Httptoken**: device user login permission code. |

Following is a mockup of the session token generated

|  |
| --- |
| {  “**X-Auth-Token**”: “5HHYVsfNOb9Dm4X0PVmQcK9aJnWUMQXU”  } |

## Delete an Existing Session

Delete an existing session (Logout) by using default UserName "admin" and Password "redfish". Following example assumes a session with ID “1” has already been created.

**DELETE**: /redfish/v1/SessionService/Sessions/<session\_id>

|  |
| --- |
| - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **LogoutDevice** (DeviceAccount) |
| gRPC Arguments | **IpAddress**(String): device IP Address and port  **UserToken**(String): device user login permission code. The user logout follows the permission by this rule.  . Administrator: logout all users.  . Operator: logout “Operator” and “ReadOnlyUser” users.  . ReadOnlyUser: only logout “ReadOnlyUser” user.  **ActUsername**(String): logout this user name. |
| gRPC API return | None |

## Account Service

Account Service contains properties common to all user accounts, such as password requirements, and control features such as account lockout. It also contains links to the collections of Manager Accounts and Roles.

## View Accounts

List default accounts.

**GET**: /redfish/v1/AccountService/Accounts

Following is a mockup of the accounts list

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#AccountService.AccountService",  "@odata.id": "/redfish/v1/AccountService/Accounts",  "@odata.type": "#ManagerAccountCollection.ManagerAccountCollection",  “Name": "Accounts Collection",  "Members@odata.count": 1,  "**Members**": [  {  "@odata.id": "/redfish/v1/AccountService/Accounts/admin"  } ]  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **ListDeviceAccounts** (DeviceAccount) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken**(String): device user (Administrator) login permission code. |
| gRPC API return | DeviceAccountList (message):  **Account**: device user list. |

## Create a New Account

Create a new account.

**POST**: /redfish/v1/AccountService/Accounts

|  |  |  |
| --- | --- | --- |
| {  "Name":"Name\_1",  **"UserName":"User\_Name\_1",**  **"Password":"User\_Password\_1",**  **"RoleId":"Administrator",**  **"Enabled":true ,**  **"Locked":false**  }  - Response : 201 CREATED | | |
|  | | |
| Property | Requirement | Value |
| UserName | Mandatory |  |
| Password | Mandatory |  |
| RoleId | Mandatory | Administrator, Operator, ReadOnlyUser |
| Enabled | Mandatory | true, false |
| Locked | Mandatory | true, false |

Following is an account generated

**GET**: /redfish/v1/AccountService/Accounts/User\_Name\_1

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#ManagerAccount.ManagerAccount",  "@odata.id": "/redfish/v1/AccountService/Accounts/User\_Name\_1",  "@odata.type": "#ManagerAccount.v1\_0\_0.ManagerAccount",  "Id": "User\_Name\_1",  "Name": "Name\_1",  "Description": null,  **"UserName": "User\_Name\_1",**  "Password": null,  "Locked": false,  "Enabled": true,  **"RoleId": "Administrator",**  "Links":{  "Role":{  "@odata.id": "/redfish/v1/AccountService/Roles/1"  }  }  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **CreateDeviceAccount** (DeviceAccount) |
| gRPC Arguments | **IpAddress**(String): device IP Address and port.  **UserToken**(String): device user (Administrator) login permission code.  **ActUsername**(String): add this user name.  **ActPassword** (String): add this user password.  **Privilege** (String): add supported privilege (Administrator/Operator/ReadOnlyUser). |
| gRPC API return | None |

## Delete an Account

Delete an existing account.

**DELETE**: /redfish/v1/AccountService/Accounts/<user\_name>

|  |
| --- |
| - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **RemoveDeviceAccount** (DeviceAccount) |
| gRPC Arguments | **IpAddress**(String): device IP Address and port.  **UserToken**(String): device user (Administrator) login permission code.  **ActUsername**(String): delete this user name. |
| gRPC API return | None |

## Event Service

The event service is a new alert mechanism for Redfish. This alert will be sent out through HTTP or HTTPS to a web service that is subscribed to the service. All subscriptions added will remain on the device across reboots. 4 event types are supported as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Event Types | Resource Added | Resource Removed | Alert | Resource Updated |
| PSU Module Being Plugged In | **x** |  |  |  |
| PSU Module Being Pulled Out |  | **x** |  |  |
| FAN Module Being Pluged In | **x** |  |  |  |
| FAN Module Being Pulled Out |  | **x** |  |  |
| FAN Module Stopped Spinning |  |  | **x** |  |
| CPU/Main Board Thermal Sensor Exceeding Critical Temperature |  |  | **x** |  |
| CPU/Main Board Thermal Sensor Exceeding Fatal Temperature |  |  | **x** |  |
| QSFP or XFP Transceiver Being Plugged In | **x** |  |  |  |
| QSFP or XFP Transceiver Being Pulled Out |  | **x** |  |  |
| Multiple Firmware update state |  |  |  | **x** |
| NOS update state |  |  |  | **x** |

## Add Subscriptions

Add subscriptions and inform Redfish intended destination for the events. Subscriptions to different events are added separately.

**POST**: /redfish/v1/EventService/Subscriptions

|  |  |  |
| --- | --- | --- |
| {  "Name": "**EventSubscription ResourceAdded TEST**",  "Destination": "https://172.17.10.60:8889",  "EventTypes": [  "**ResourceAdded**"  ],  "Context": "**ResourceAdded notification message**",  "Protocol": "Redfish"  }  - Response : 201 CREATED | | |
|  | | |
| Property | Requirement | Value |
| Name | Mandatory |  |
| Destination | Mandatory |  |
| EventTypes | Mandatory | ResourceAdded ResourceRemoved Alert ResourceUpdated |
| Protocol | Mandatory | Redfish |

Following is a mockup of an event triggered

|  |
| --- |
| {  "@odata.context": "/rest/v1/$metadata#EventService/Members/Events/1",  "@odata.id": "/rest/v1/EventService/Events/1",  "@odata.type": "#EventService.1.0.0.Event",  "Id": "1",  "Name": " **EventSubscription ResourceAdded TEST** ",  "Description": "Events",  "**Events**": [ {  "EventType": "ResourceAdded",  "EventId": "12f03ec2-714f-11e9-ba1f-ef16238bd344",  "EventTimestamp": "2019-05-08T05:06:44+00:00",  "Severity": "OK",  "Message": "**Port 1 Plug In.**",  "MessageId": "Base.1.0.0.Success",  "MessageArgs": [],  "OriginOfCondition": "",  "Context": "**ResourceAdded notification message**"  }  } |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **SubscribeGivenEvents** (GivenEventList) |
| gRPC Arguments | **IpAddress**(String): device IP Address and port.  **UserToken**(String): device user (Administrator or Operator) login permission code.  **Events** (String): The event supported ResourceAdd/ResourceRemoved/Alert/ResourceUpdated.  **EventServerAddr** (String): The device event you want to notify to this address.  **EventServerPort** (String): This device event you want to notify the port. |
| gRPC API return | None |

## View Subscriptions

List current subscriptions.

**GET** : URL: /redfish/v1/EventService/Subscriptions

Following is a mockup of list of events registered

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#EventService.EventService",  "@odata.id": "/redfish/v1/EventService/Subscriptions",  "@odata.type": "#EventDestinationCollection.EventDestinationCollection",  "Name": "Event Subscriptions Collection",  "Description": "Collection of Event Subscriptions",  "Members@odata.count": 4,  "Members":[  {  "@odata.id": "/redfish/v1/EventService/Subscriptions/3"  },  {  "@odata.id": "/redfish/v1/EventService/Subscriptions/1"  },  {  "@odata.id": "/redfish/v1/EventService/Subscriptions/2"  },  {  "@odata.id": "/redfish/v1/EventService/Subscriptions/4"  }  ]  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GetCurrentDevices** () |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken**(String): device user (Administrator) login permission code. |
| gRPC API return | DeviceListByIp (message):  **IpAddress**: device IP address and port list. |

## Delete Subscriptions

Delete a subscription from Redfish service.

**DELETE** : /redfish/v1/EventService/Subscriptions/<subscription\_id>

|  |
| --- |
| - Response : 204 NO\_CONTENT |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **UnsubscribeGivenEvents** (GivenEventList) |
| gRPC Arguments | **IpAddress**(String): device IP Address and port.  **UserToken**(String): device user (Administrator or Operator) login permission code.  **Events** (String): The event supported ResourceAdd/ResourceRemoved/Alert/ResourceUpdated. |
| gRPC API return | None |

## Hardware Information

Hardware inventory and health state information, such as information about temperature sensors, fans, PSU, HDD, NIC, can be obtained using Redfish API’s.

## Chassis information

Retrieve chassis information.

**GET**: /redfish/v1/Chassis/<chassis\_id>

Following is a mockup of chassis info retrieved for Edgecore’s GPON device x86-64-accton-asgvolt64-r0

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#Chassis.Chassis",  "@odata.id": "/redfish/v1/Chassis/1",  "@odata.type": "#Chassis.v1\_3\_0.Chassis",  "Id": "1",  "ChassisType": "Drawer",  "Name": "Chassis",  "Description": "ASGvOLT64-O-AC-F",  "PowerState": "On",  "Manufacturer": "Accton",  "Model": "x86\_64-accton\_asgvolt64-r0",  "SKU": null,  "SerialNumber": "EC1921003307",  "PartNumber": "FN1EC0964000Z",  "AssetTag": “”,  "IndicatorLED": "Lit",  "Status":{  "State": "Enabled",  "Health": "OK",  "HealthRollup": "OK"  },  "@odata.type": "#Intel.Oem.Chassis", {  "Location": {  "Id": null,  "ParentId": null  },  "Links": {  "@odata.type": "#Chassis.v1\_2\_0.Links",  "Contains":[],  "ComputerSystems": [ {  "@odata.id": "/redfish/v1/Systems/1",  "Actions": {  "#Chassis.Reset":{"target": "/redfish/v1/Chassis/1/Actions/Chassis.Reset",  "ResetType@Redfish.AllowableValues": ["ForceOff",¡K}, "Thermal":{"@odata.id": "/redfish/v1/Chassis/1/Thermal"}, "Power":{"@odata.id": "/redfish/v1/Chassis/1/Power" } ]  …… }  ……  }  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GetDeviceData** (Device) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken**(DeviceAccount)(String): device user login permission code.  **RedfishAPI** (String): /redfish/v1/Chassis/<chassis\_id>. |
| gRPC API return | DeviceData (message):  **DeviceData**: device data by “RedfishAPI”. |

## Thermal Sensor Information

Fan speeds can be read from "Reading" property in payload of Fans object. PSU power thermal sensor can be read from "ReadingCelsius" property in payload of Temperatures object.

**GET**: /redfish/v1/Chassis/<chassis\_id>/Thermal

Following is a mockup of thermal snsor info retrieved for Edgecore’s XGSPON device x86-64-accton-asxvolt16-r0

|  |
| --- |
| {  "Description": "Collection of Thermal Sensors",  "Redundancy":[],  "Temperatures":[  {  "@odata.id": "/redfish/v1/Chassis/1/Thermal",  "MemberId": "1",  "Name": "System CPU Thermal Sensor Temperature",  "PhysicalContext": "CPU",  "SensorNumber": 1,  "Status":{"HealthRollup": "**OK**", "State": "**Enabled**"},  **"ReadingCelsius": 49,**  "UpperThresholdNonCritical": 83,  "UpperThresholdCritical": 93,  "UpperThresholdFatal": 105,  "RelatedItem":[{"@odata.id": "/redfish/v1/Chassis/1" }]  },  {  "@odata.id": "/redfish/v1/Chassis/1/Thermal",  "MemberId": "2",  "Name": "Chassis Thermal Sensor Temperature",  "PhysicalContext": "SystemBoard",  "SensorNumber": 2,  "Status":{"HealthRollup": "**OK**", "State": "**Enabled**"},  **"ReadingCelsius": 35,**  "UpperThresholdNonCritical": 83,  "UpperThresholdCritical": 93,  "UpperThresholdFatal": 105,  "RelatedItem":[{"@odata.id": "/redfish/v1/Chassis/1" }]  },…  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GetDeviceData** (Device) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken**(DeviceAccount)(String): device user login permission code.  **RedfishAPI** (String): /redfish/v1/Chassis/<chassis\_id>/Thermal. |
| gRPC API return | DeviceData (message):  **DeviceData**: device data by “RedfishAPI”. |

## PSU Power Information

Power consumption information can be read from the "**PowerConsumedWatts**" property in the payload of the PowerControl object. PSU presence information can be read from the “State” property in the payload of “Status” object.

**GET**: /redfish/v1/Chassis/<chassis\_id>/Power

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#Power.Power",  "@odata.id": "/redfish/v1/Chassis/1/Power",  "Id": "Power",  "@odata.type": "#Power.v1\_1\_0.Power",  "Name": "Power Collection",  "Description": "Collection of Power",  "PowerControl":[  {"@odata.id": "/redfish/v1/Chassis/1/Power", "MemberId": "1", "Name": "**System Power Control**",…},  {  "@odata.id": "/redfish/v1/Chassis/1/Power",  "MemberId": "2",  "Name": "System Power Control",  **"PowerConsumedWatts": 146,**  "Status":{"Health": "OK", "State": "Enabled"},  "PowerRequestedWatts": null,  "PowerAvailableWatts": null,  "PowerCapacityWatts": 0,  "PowerAllocatedWatts": null,  "PowerLimit":{"LimitInWatts": null, "LimitException": null, "CorrectionInMs": null},  "RelatedItem":[{"@odata.id": "/redfish/v1/Chassis/1" }],  "Oem":{}  }  ],  ….  }  ],  "Oem":{}  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GetDeviceData** (Device) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken**(DeviceAccount)(String): device user login permission code.  **RedfishAPI** (String): /redfish/v1/Chassis/<chassis\_id>/Power. |
| gRPC API return | DeviceData (message):  **DeviceData**: device data by “RedfishAPI”. |

## QSFP/XFP Information

## Collection of Ports

Retrieve the list of ports

**GET**: /redfish/v1/EthernetSwitches/<ethernetswitch\_id>/Ports

Following is a mockup of list of ports

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#EthernetSwitchPortCollection.EthernetSwitchPortCollection",  "@odata.id": "/redfish/v1/EthernetSwitches/1/Ports",  "@odata.type": "#EthernetSwitchPortCollection.EthernetSwitchPortCollection",  "Name": "Ethernet Switch Port Collection",  "Description": "Collection of Ethernet Switch Ports",  "Members@odata.count": 4,  "**Members**":[  {"@odata.id": "/redfish/v1/EthernetSwitches/1/Ports/1"},  {"@odata.id": "/redfish/v1/EthernetSwitches/1/Ports/2"},  {"@odata.id": "/redfish/v1/EthernetSwitches/1/Ports/3"},  {"@odata.id": "/redfish/v1/EthernetSwitches/1/Ports/4"}  ]  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GetDeviceData** (Device) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken**(DeviceAccount)(String): device user login permission code.  **RedfishAPI** (String): /redfish/v1/EthernetSwitches/<ethernetswitch\_id>/Ports. |
| gRPC API return | DeviceData (message):  **DeviceData**: device data by “RedfishAPI”. |

## Port Information

Retrieve detailed information about a port, if **present**, and if supported transceivers are detected, their detailed information.

**GET**: /redfish/v1/EthernetSwitches/<ethernetswitch\_id>/Ports/<port\_id>

Following is a mockup of port details

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#EthernetSwitchPort.EthernetSwitchPort",  "@odata.id": "/redfish/v1/EthernetSwitches/1/Ports/1",  "@odata.type": "#EthernetSwitchPort.v1\_0\_0.EthernetSwitchPort",  "Id": "1",  "Name": "Port1",  "Description": "Ethernet Switch Port description",  "PortId": "Port ID",  "Status":{"State": "**Enabled**", "Health": "OK", "HealthRollup": "OK"},  "LinkType": "Ethernet",  "OperationalState": null,  "AdministrativeState": null,  "LinkSpeedMbps": null,  "NeighborInfo":{"SwitchId": null, "PortId": null, "CableId": null},  "NeighborMAC": null,  "FrameSize": null,  "Autosense": null,  "FullDuplex": null,  "MACAddress": null,  "PortClass": null,  "PortMode": null,  "PortType": null,  "Oem":{},  "IPv4Addresses":[],  "IPv6Addresses":[],  "VLANs":{"@odata.id":"/redfish/v1/EthernetSwitches/1/Ports/1/VLANs"},  "StaticMACs":{"@odata.id":"/redfish/v1/EthernetSwitches/1/Ports/1/StaticMACs"},  "Links":{  …  }  - Response : 200 OK |

|  |  |  |
| --- | --- | --- |
| **Transceiver** | **Vendor** | **Part No** |
| QSFP | Precision | QSFP28AOC03 |
| XFP (XGSPON) | Source Photonics | XPPXG2N1CDFA |
| XFP (XGSPON) | Hisense | LTH7226-PC+ |
| SFP (GPON) | Hisense | LTE3680M-BH+ |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GetDeviceData** (Device) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken**(DeviceAccount)(String): device user login permission code.  **RedfishAPI** (String): /redfish/v1/EthernetSwitches/<ethernetswitch\_id>/Ports/<port\_id>. |
| gRPC API return | DeviceData (message):  **DeviceData**: device data by “RedfishAPI”. |

## Enable/Disable Log Service

This resource represents the log service for the resource or service to which it is associated. It records all ONL peripheral Add/Remove/Alert events, such as, fan plugging-in and out, thermal sensor exceeding fatal critical threshold temperature, …, etc.

## Enable/Disable Log Service

**PATCH**: /redfish/v1/Managers/<manager\_id>/LogServices/<log\_id>

|  |  |  |
| --- | --- | --- |
| {  "ServiceEnabled": true  }  - Response : 200 OK | | |
|  | | |
| Property | Requirement | Value |
| ServiceEnabled | Mandatory | true, false |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **EnableLogServiceState** (LogService) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator or Operator) login permission code.  **LogServiceEnabled** (Bool): true: enable, false: disable |
| gRPC API return | None |

## Get Log Entries of the Target Device

**GET** : /redfish/v1/Managers/<manager\_id>/LogServices/<log\_id>/Entries

Following is a mockup of logs retrieved from the device

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#LogEntryCollection.LogEntryCollection",  "@odata.id": "/redfish/v1/Managers/1/LogServices/1/Entries",  "@odata.type": "#LogEntryCollection.LogEntryCollection",  "Name": "Log Service Collection",  "Description": "Collection of Logs for this System",  "Members@odata.count": 3,  "**Members**":[  {"@odata.id": "/redfish/v1/Managers/1/LogServices/1/Entries/0","Id": "0"},  {"@odata.id": "/redfish/v1/Managers/1/LogServices/1/Entries/1","Id": "1"},  {"@odata.id": "/redfish/v1/Managers/1/LogServices/1/Entries/2","Id": "2"}  ]  }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **EnableLogServiceState** (LogService) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator or Operator) login permission code.  **LogServiceEnabled** (Bool): true: enable, false: disable |
| gRPC API return | None |

## Clear Log Entries of the Target Device

**POST**:/redfish/v1/Managers/<manager\_id>/LogServices/<log\_id>/Actions/LogService.Reset

|  |
| --- |
| { }  - Response : 200 OK |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **ResetDeviceLogData** (LogService) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator or Operator) login permission code. |
| gRPC API return | None |

## System Management

### System Reset

Reset the system.

**POST**: URL: /redfish/v1/Systems/<*system\_id*>[[2]](#footnote-2)/Actions/ComputerSystem.Reset

|  |  |  |
| --- | --- | --- |
| {  "**ResetType**":” GracefulRestart”,  }  - Response : 200 OK | | |
|  | | |
| Property | Requirement | Value |
| ResetType | Mandatory | GracefulShutdown, GracefulRestart |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **ResetDeviceSystem** (SystemBoot) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator or Operator) login permission code.  **ResetType** (String): GracefulShutdown/GracefulRestart |
| gRPC API return | None |

## Configure Boot Source

Change the boot source for the system

PATCH: /redfish/v1/Systems/<*system\_id>*

|  |  |  |
| --- | --- | --- |
| {  "**BootSourceOverrideEnabled**":”Once”,  "**BootSourceOverrideTarget**":”Pxe”,  } | | |
|  | | |
| Property | Requirement | Value |
| BootSourceOverrideEnabled | Mandatory | Disabled, Once, Continuous |
| BootSourceOverrideTarget | Mandatory | Pxe, Hdd, BiosSetup |

Following is a mockup of the reconfigured boot source

|  |
| --- |
| {  "**BootSourceOverrideEnabled**":”Once”,  "**BootSourceOverrideTarget**":”Pxe”  } |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **ConfigureBootSource** (SystemBoot) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator) login permission code.  **BootSourceOverrideEnabled** (String): Disable/Once/Continuous  **BootSourceOverrideTarget** (String): Pxe/Hdd/BiosSetup |
| gRPC API return | None 🡪 **(This function is not supported now)** |



### Configure Default OS in GRUB (OEM)

Set the default boot OS in UEFI grub system.

Only support devices that use UEFI BIOS booting system.

To get available bootable partitions

**GET**: /redfish/v1/Systems/<*system\_id>*

|  |
| --- |
| {  ...  ...  "Oem": {  "Intel\_RackScale": {  "@odata.type": "#Intel.Oem.ComputerSystem",  "PciDevices": [],  "PCIeConnectionId": [],  "ProcessorSockets": 1,  "MemorySockets": 2,  "Accton\_Oem": {  "Actions": {  "#ComputerSystem.GrubDefault": {  "target": "/redfish/v1/Systems/1/Actions/ComputerSystem.GrubDefault",  "GrubDefault@Redfish.AllowableValues": [  **"Open Network Linux",**  **"ONIE:diag",**  **"ONIE:embed",**  **"ONIE:install",**  **"ONIE:rescue",**  **"ONIE:uninstall",**  **"ONIE:update"**  ]  }  },  **"GrubDefault": "Open Network Linux"**  }  }  }  ...  ...  }  - Response : 200 OK |

To set "ONIE:update" as default boot partitions

**POST**: /redfish/v1/Systems/<*system\_id>*/Actions/ComputerSystem.GrubDefault

|  |  |  |
| --- | --- | --- |
| {  **"GrubDefault":"ONIE:update"**  } | | |
|  | | |
| Property | Requirement | Value |
| GrubDefault | Mandatory | Depend on GrubDefault@Redfish.AllowableValues |

## If user set default boot as ONIE environment and want back to NOS, user need change BIOS order in BIOS to return back to NOS again.

Following is a mockup of the reconfigured grub source

|  |
| --- |
| {  **"GrubDefault": "Open Network Linux"**  } |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **SetDeviceDefaultBoot** (SystemBoot) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator) login permission code.  **DefaultBoot** (String): keyword of default boot name |
| gRPC API return | None |

### Update System Firmware

Update service can use MU(Multiple Updater) procedures of Edge-core to upgrade "multiple firmware" at the same time.

MU only support devices that use UEFI BIOS booting system.

MU needs to use "**wget**" command to retrieve the MU images from the remote Http(s) server.

Please note that under SONiC OS system doesn't have this application pre-install. You need to install this application first before you use this feature.

And you need to add a subscription of "**ResourceUpdated**" in event service to receive the related “**ResourceUpdated**” notification during the update process.

Trigger MU(Multiple Updater) update process.

The operator needs to put the MU firmware image in Http(s) server and provide the **ImageURI** path in

**POST**: /redfish/v1/UpdateService/FirmwareInventory/MU

|  |  |  |
| --- | --- | --- |
| {  "**ImageURI**":"https://172.17.8.103/asgvolt64\_pak\_va.0.0.2.updater"  }  - Response Payload :  {  "@odata.context": "/redfish/v1/$metadata#SoftwareInventory.SoftwareInventory",  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/MU",  "Id": "Multipal Updater",  "@odata.type": "#SoftwareInventory.v1\_2\_1.SoftwareInventory",  "Name": "Edgecore Multiple Updater information",  "Description": "Update Service",  "Oem":{  "Intel\_RackScale":{  "@odata.type": "#Intel.Oem.Chassis",  "Accton\_Oem":{  "UpdateState": "**Downloading**"  }  }  }  }  - Response : 200 OK | | |
|  | | |
| Property | Requirement | Value |
| ImageURI | Mandatory |  |

If not support UEFI environment.

|  |
| --- |
| - Response : 503 Service Unavailable |

If not enough memory/disk space

|  |
| --- |
| - Response : 507 Insufficient Storage |

If update image file not found

|  |
| --- |
| - Response : 400 Bad Request |

If under processing MU or NOS update process.

|  |
| --- |
| - Response : 403 Forbidden |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **SendDeviceSoftwareDownloadURI** (SoftwareUpdate) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator or Operator) login permission code.  **SoftwareDownloadType** (String): “MU”  **SoftwareDownloadURI** (String): The software download URI (included “http://” / “https://” / “tftp://” ) |
| gRPC API return | None |

### Check the System Firmware update status

Retrieves the status of a reflash action started previously

**GET**: /redfish/v1/UpdateService/FirmwareInventory/MU

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#SoftwareInventory.SoftwareInventory",  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/MU",  "@odata.type": "#SoftwareInventory.v1\_2\_1.SoftwareInventory",  "Description": "Update Service",  "Id": "Multipal Updater",  "Name": "Edgecore Multiple Updater information",  "Oem": {  "Intel\_RackScale": {  "@odata.type": "#Intel.Oem.Chassis",  "Accton\_Oem": {  "**UpdateState**": "**LastCompletedAt 2020-08-01T16:51:42+00:00**"  }  }  }  }  - Response : 200 OK |

**UpdateState**: (OEM)

|  |  |
| --- | --- |
| **None** | The Device does not have any update process before |
| **Downloading** | Firmware image is downloading |
| **InProcessing** | Preparing update process |
| **Rebooting** | Reboot into ONIE to start update process |
| **LastCompletedAt** | The last success update time |
| **Failure** | The last time update result is failure |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GenericDeviceAccess** (Device) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (DeviceAccount)(String): device user login permission code.  **RedfishAPI** (String): /redfish/v1/UpdateService/FirmwareInventory/MU |
| gRPC API return | HttpData(message):  **ResultData**: device data |

You can get following notification event message strings by JSON format during MU update process in your Http(s) event listener:

After trigger POST method of update process and start to download MU image.If under processing MU or NOS update process.

|  |
| --- |
| ('EventType is ', **u'ResourceUpdated'**)  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**DownloadStart'**)  ('EventId is ', u'9e8a02f8-1a8e-11eb-85a8-cfceb00f2447')  ('EventTimestamp is ', u'2020-10-30T09:02:21+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[**172.17.8.41**][**Multiple firmware update**] **asgvolt64\_pak\_vb.0.0.2\_zz.updater Downloading...'**)  ('MessageArgs is ', [u'172.17.8.41', u'Multiple firmware update', u'asgvolt64\_pak\_vb.0.0.2\_zz.updater'])  ('OriginOfCondition is ', u'**/redfish/v1/UpdateService/FirmwareInventory/MU**') |

Retrieve MU image successfully from server.

|  |
| --- |
| ('EventType is ', **u'ResourceUpdated'**)  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**DownloadSuccessful'**)  ('EventId is ', u'aa907474-1a8e-11eb-a168-f72386daf63c')  ('EventTimestamp is ', u'2020-10-30T09:02:41+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[172.17.8.41][Multiple firmware update] asgvolt64\_pak\_vb.0.0.2\_zz.updater Download Successful.')  ('MessageArgs is ', [u'172.17.8.41', u'Multiple firmware update', u'asgvolt64\_pak\_vb.0.0.2\_zz.updater'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/MU') |

Retrieve MU image failure from server.

|  |
| --- |
| ('EventType', **'ResourceUpdated'**)  ('MessageId', 'AcctonFirmwareUpdateRegistry.1.0.0.**DownloadFailure'**)  ('EventId', 'ea4a6362-1855-11eb-8617-e72e9aabcb38')  ('EventTimestamp', '2020-10-27T13:11:24+00:00')  ('Severity', 'OK')  ('Message', '[172.17.8.41][Multiple firmware update] asgvolt64\_pak\_va.0.0.2.updater Download Failure.')  ('Context': 'ResourceUpdated notification messages')  ('OriginOfCondition', u'/redfish/v1/UpdateService/FirmwareInventory/MU')  ('MessageArgs', ["172.17.8.41","Multiple firmware update","no\_file\_asgvolt64\_pak\_va.0.0.2.updater"])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/MU') |

**Preparing update process** and ready to **reboot** into ONIE environment to update.

|  |
| --- |
| ('EventType is ', **u'ResourceUpdated'**)  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**UpdateInProgress'**)  ('EventId is ', u'61840396-1a92-11eb-a903-7794124645f8')  ('EventTimestamp is ', u'2020-10-30T09:29:16+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[172.17.8.41][Multiple firmware update] InProgress.')  ('MessageArgs is ', [u'172.17.8.41', u'Multiple firmware update', u'asgvolt64\_pak\_vb.0.0.1\_zz.updater'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/MU')  ('EventType', 'ResourceUpdated')  ('MessageId', 'AcctonFirmwareUpdateRegistry.1.0.0.**Reboot'**)  ('EventId', 'ea4a6362-1855-11eb-8617-e72e9aabcb38')  ('EventTimestamp', '2020-10-27T13:11:24+00:00')  ('Severity', 'OK')  ('Message', '[172.17.8.41][Multiple firmware update] Rebooting. ')  ('Context': 'ResourceUpdated notification messages')  ('OriginOfCondition', u'/redfish/v1/UpdateService/FirmwareInventory/MU')  ('MessageArgs', ["172.17.8.41","Multiple firmware update","asgvolt64\_pak\_va.0.0.2.updater"]) |

Preparing update process has **error**.

|  |
| --- |
| ('EventType is ', **u'ResourceUpdated'**)  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**BadPackage'**)  ('EventId is ', u'618f25a0-1a92-11eb-9c6a-3f592171e9b5')  ('EventTimestamp is ', u'2020-10-30T09:29:16+00:00')  ('Severity is ', u'Warning')  ('Message is ', u'[172.17.8.41][Multiple firmware update] asgvolt64\_pak\_vb.0.0.1\_zz.updater InProcess Failure.')  ('MessageArgs is ', [u'172.17.8.41', u'Multiple firmware update', u'asgvolt64\_pak\_vb.0.0.1\_zz.updater'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/MU') |

Update result is **successful** in ONIE environment and return back to NOS.

|  |
| --- |
| ('EventType', **'ResourceUpdated'**)  ('MessageId', 'AcctonFirmwareUpdateRegistry.1.0.0.**UpdateSuccessful'**)  ('EventId', 'ea4a6362-1855-11eb-8617-e72e9aabcb38')  ('EventTimestamp', '2020-10-27T13:11:24+00:00')  ('Severity', 'OK')  ('Message', '[172.17.8.41][Multiple firmware update] asgvolt64\_pak\_va.0.0.2.updater Update Successful. ')  ('Context': 'ResourceUpdated notification messages')  ('MessageArgs', ["172.17.8.41","Multiple firmware update","asgvolt64\_pak\_va.0.0.2.updater"])  ('OriginOfCondition', u'/redfish/v1/UpdateService/FirmwareInventory/MU') |

Update result is **failure** in ONIE environment and return back to NOS.

|  |
| --- |
| ('EventType', **'ResourceUpdated'**)  ('MessageId', 'AcctonFirmwareUpdateRegistry.1.0.0.**UpdateFailed'**)  ('EventId', 'ea4a6362-1855-11eb-8617-e72e9aabcb38')  ('EventTimestamp', '2020-10-27T13:11:24+00:00')  ('Severity', 'OK')  ('Message', '[172.17.8.41][Multiple firmware update] asgvolt64\_pak\_va.0.0.2.updater Update Failure. ')  ('Context': 'ResourceUpdated notification messages')  ('MessageArgs', ["172.17.8.41","Multiple firmware update","asgvolt64\_pak\_va.0.0.2.updater"])  ('OriginOfCondition', u'/redfish/v1/UpdateService/FirmwareInventory/MU') |

### Update NOS Image

Please note that under SONiC OS system doesn't have “wget” command pre-install.

To use "wget" command to retrieve the NOS images from the remote Http(s) server. You need to install this application first before you use this feature.

And you need to add a subscription of "**ResourceUpdated**" in event service to receive the related “**ResourceUpdated**”notification during the update process.

**POST**: /redfish/v1/UpdateService/FirmwareInventory/NOS

|  |
| --- |
| {  "**ImageURI**":"https://172.17.8.103/ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER"  }  - Response Payload :  {  "@odata.context": "/redfish/v1/$metadata#SoftwareInventory.SoftwareInventory",  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/NOS",  "Id": "NOS Updater",  "@odata.type": "#SoftwareInventory.v1\_2\_1.SoftwareInventory",  "Name": "Edgecore NOS Updater information",  "Description": "Update Service",  "Oem":{  "Intel\_RackScale":{  "@odata.type": "#Intel.Oem.Chassis",  "Accton\_Oem":{  "UpdateState": "Downloading"  }  }  }  }  - Response : 200 OK |

|  |  |  |
| --- | --- | --- |
| Property | Requirement | Value |
| ImageURI | Mandatory |  |

If not support UEFI environment.

|  |
| --- |
| - Response : 503 Service Unavailable |

If not enough memory/disk space

|  |
| --- |
| - Response : 507 Insufficient Storage |

If update image file not found

|  |
| --- |
| - Response : 400 Bad Request |

If under processing MU or NOS update process.

|  |
| --- |
| - Response : 403 Forbidden |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **SendDeviceSoftwareDownloadURI** (SoftwareUpdate) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (String): device user (Administrator or Operator) login permission code.  **SoftwareDownloadType** (String): “NOS”  **SoftwareDownloadURI** (String): The software download URI (included “http://” / “https://” / “tftp://” ) |
| gRPC API return | None |

### Check the NOS Update status

Retrieves the status of a reflash action started previously

**GET**: /redfish/v1/UpdateService/FirmwareInventory/NOS

|  |
| --- |
| {  "@odata.context": "/redfish/v1/$metadata#SoftwareInventory.SoftwareInventory",  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/**NOS**",  "@odata.type": "#SoftwareInventory.v1\_2\_1.SoftwareInventory",  "Description": "Update Service",  "Id": "**NOS Updater**",  "Name": "Edgecore NOS Updater information",  "Oem": {  "Intel\_RackScale": {  "@odata.type": "#Intel.Oem.Chassis",  "Accton\_Oem": {  UpdateState": "LastCompletedAt 2020-08-01T16:51:42+00:00"  }  }  }  }  - Response : 200 OK |

**UpdateState**: (OEM)

|  |  |
| --- | --- |
| **None** | The Device does not have any update process before |
| **Downloading** | N image is downloading |
| **InProcessing** | Preparing update process |
| **Rebooting** | Reboot into ONIE to start update process |
| **LastCompletedAt** | The last success update time |
| **Failure** | The last time update result is failure |

|  |  |
| --- | --- |
| Device Management gRPC definition | |
| gRPC API name | **GenericDeviceAccess** (Device) |
| gRPC Arguments | **IpAddress** (String): device IP Address and port  **UserToken** (DeviceAccount)(String): device user login permission code.  **RedfishAPI** (String): /redfish/v1/UpdateService/FirmwareInventory/NOS |
| gRPC API return | HttpData(message):  **ResultData**: device data |

You can get following notification event message strings by JSON format during NOS update process in your https event listener:

Trigger POST method of update process and start to download NOS image.

|  |
| --- |
| ('EventType is ', u'**ResourceUpdated**')  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**DownloadStart'**)  ('EventId is ', u'66c13a46-1a87-11eb-ab41-2725e197bb59')  ('EventTimestamp is ', u'2020-10-30T08:10:41+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[172.17.8.41][NOS update] ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER Downloading...')  ('MessageArgs is ', [u'172.17.8.41', u'NOS update', u'ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/**NOS'**) |

Retrieve NOS image successfully from server.

|  |
| --- |
| ('EventType is ', u'ResourceUpdated')  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**DownloadSuccessful'**)  ('EventId is ', u'8672a50a-1a87-11eb-965a-f74ff1553eb1')  ('EventTimestamp is ', u'2020-10-30T08:11:34+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[172.17.8.41][NOS update] ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER Download Successful.')  ('MessageArgs is ', [u'172.17.8.41', u'NOS update', u'ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/NOS') |

Retrieve NOS image failure from server..

|  |
| --- |
| ('EventType is ', u'ResourceUpdated')  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**DownloadFailure'**)  ('EventId is ', u'59a2bc50-1a8b-11eb-8f5f-bf4c9dde0cdd')  ('EventTimestamp is ', u'2020-10-30T08:38:57+00:00')  ('Severity is ', u'Warning')  ('Message is ', u'[172.17.8.41][NOS update] NO\_FILE\_ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER Download Failure.')  ('MessageArgs is ', [u'172.17.8.41', u'NOS update', u'NO\_FILE\_ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/NOS') |

Preparing update process and ready to reboot into ONIE environment to update...

|  |
| --- |
| ('EventType is ', u'ResourceUpdated')  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**UpdateInProgress'**)  ('EventId is ', u'8672a50a-1a87-11eb-965a-f74ff1553eb1')  ('EventTimestamp is ', u'2020-10-30T08:11:34+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[172.17.8.41][NOS update] InProgress.')  ('MessageArgs is ', [u'172.17.8.41', u'NOS update', u'ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/NOS')  ('EventType is ', u'ResourceUpdated')  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.Reboot')  ('EventId is ', u'8672a50a-1a87-11eb-965a-f74ff1553eb1')  ('EventTimestamp is ', u'2020-10-30T08:11:34+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[172.17.8.41][NOS update] Rebooting.')  ('MessageArgs is ', [u'172.17.8.41', u'NOS update', u'ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/NOS') |

Preparing update process has error.

|  |
| --- |
| ('EventType is ', u'ResourceUpdated')  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**BadPackage'**)  ('EventId is ', u'ac23d4e8-1a89-11eb-8362-577ad9a9199b')  ('EventTimestamp is ', u'2020-10-30T08:26:56+00:00')  ('Severity is ', u'Warning')  ('Message is ', u'[172.17.8.41][NOS update] ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER InProcess Failure.')  ('MessageArgs is ', [u'172.17.8.41', u'NOS update', u'ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER'])  ('OriginOfCondition is ', u'/redfish/v1/UpdateService/FirmwareInventory/NOS') |

Update result is successful in ONIE environment and return back to NOS.

|  |
| --- |
| ('EventType is ', u'ResourceUpdated')  ('MessageId is ', u'AcctonFirmwareUpdateRegistry.1.0.0.**UpdateSuccessful'**)  ('EventId is ', u'04fc74c8-1a88-11eb-aae4-0b51238e5edf')  ('EventTimestamp is ', u'2020-10-30T08:15:06+00:00')  ('Severity is ', u'OK')  ('Message is ', u'[NOS update] ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER Update Successful.')  ('MessageArgs is ', [u'172.17.8.41', u'NOS update', u'ONL-HEAD\_ONL-OS8\_2020-09-03.1004-f0bcb23\_AMD64\_INSTALLED\_INSTALLER'])  ('OriginOfCondition is ', u'') |

Event messages during MU and NOS update use Accton's extended MessageID define in

**GET**: redfish/v1/Registries/AcctonFirmwareUpdateRegistry

|  |
| --- |
| {  "@Redfish.Copyright": "Copyright 2019-2020 Accton. All rights reserved.",  "@odata.type": "#MessageRegistry.v1\_0\_0.MessageRegistry",  "Description": "This registry defines the messages for Accton Firmware Update Registry",  "Id": "AcctonFirmwareUpdateRegistry.v1.0.0",  "Language": "en",  "Messages":{  "**BadPackage**":{"Description": "Indicates that software update failure because some error in preparing update procedure.", "Message": "[%1][%2] %3 InProcess Failure.", "NumberOfArgs": 3,,  "**DownloadFailure**":{"Description": "Indicates that software update download failure.", "Message": "[%1][%2] %3 Download Failure.", "NumberOfArgs": 3,,  "**DownloadStart**":{"Description": "Indicates that software update download Successful.", "Message": "[%1][%2] %3 Downloading...", "NumberOfArgs": 3,,  "**DownloadSuccessful**":{"Description": "Indicates that software update download Successful.", "Message": "[%1][%2] %3 Download Successful.", "NumberOfArgs": 3,,  "**Reboot**":{"Description": "Indicates that device is going to reboot and enter ONIE environment and enter to update mode.", "Message": "[%1][%2] %3 Rebooting.", "NumberOfArgs": 3,,  "**UpdateFailed**":{"Description": "Indicates that software update result is failure.", "Message": "[%1][%2] %3 Update Failure.", "NumberOfArgs": 3,,  "UpdateInProgress":{"Description": "Indicates that device is parparing update procedure.", "Message": "[%1][%2] %3 InProcess.", "NumberOfArgs": 3,,  "**UpdateSuccessful**":{"Description": "Indicates that software update result is successful.", "Message": "[%1][%2] %3 Update Successful.", "NumberOfArgs": 3,  },  "Name": "Accton FirmwareUpdate Registry Message Registry",  "OwningEntity": "Accton",  "RegistryPrefix": "AcctonFirmwareUpdateRegistry",  "RegistryVersion": "1.0.0"  }  - Response : 200 OK |

# Potential Future Enhancements

## In-band IP Management

Add a tiny switch between BMC and the switch chip connected using the ethernet interface on the motherboard to gather system information in the event CPU is not functioning.

A screenshot of a cell phone

Description automatically generated

1. Device Manager will not support notifications setup through BMC. [↑](#footnote-ref-1)
2. system\_id is a placeholder for the system resource identifier. [↑](#footnote-ref-2)