Redfish

Table of contents

Redfish provides a RESTful interface designed to manage IT infrastructure and is implemented using a modern toolchain (HTTP(s)/TLS/JSON).

Redfish supports the operations listed in this section.

BIOS Configuration Schema

The BIOS schema contains properties related to the BIOS attribute registry. The attribute registry describes the system-specific BIOS attributes and actions for changing to BIOS settings. It is likely that a client finds the `@Redfish.Settings` term in this resource, and if it is found, the client makes requests to change BIOS settings by modifying the resource identified by the `@Redfish.Settings` annotation.

URI	`/redfish/v1/Systems/{ComputerSystemId}/Bios`
Schema file	`http://redfish.dmtf.org/schemas/v1/Bios.v1_1_1.json`
Operations	GET; PATCH

Example command and response:

```
$ curl -sku $BMC_USER:$BMC_PASSWORD -X GET
https://$BMC_IP/redfish/v1/Systems/Bluefield/Bios
{
  "@Redfish.Settings": {
    "@odata.type": "#Settings.v1_3_5.Settings",
    "SettingsObject": {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Bios/Settings"
    }
  },
  "@odata.id": "/redfish/v1/Systems/Bluefield/Bios",
  "@odata.type": "#Bios.v1_2_0.Bios",
  "Actions": {
    "#Bios.ChangePassword": {
      "target":
"/redfish/v1/Systems/Bluefield/Bios/Actions/Bios.ChangePassword"
    },
```

```
    "#Bios.ResetBios": {
      "target":
"/redfish/v1/Systems/Bluefield/Bios/Actions/Bios.ResetBios"
    }
  },
  "Attributes": {
    "BootPartitionProtection": false,
    "CeThreshold": 5000,
    "CurrentUefiPassword": "",
    "DateTime": "2024-10-17T19:47:04Z",
    "DefaultPasswordPolicy": true,
    "DisableHEST": false,
    "DisableI2c1": false,
    "DisablePCIe": false,
    "DisableSPMI": false,
    "DisableTMFF": false,
    "EmmcWipe": false,
    "Enable2ndeMMC": false,
    "EnableDdr5600": false,
    "EnableOPTEE": false,
    "EnableSMMU": true,
    "FieldMode": false,
    "ForcePxeRetryDisable": false,
    "HostPrivilegeLevel": "Restricted",
    "InternalCPUModel": "Embedded",
    "L3CachePartitionLevel": 0,
    "LegacyPasswordEnable": false,
    "NicMode": "DpuMode",
    "NvmeWipe": false,
    "OsArgs": "",
    "ResetEfiVars": false,
    "SPCR_UART": "Disabled",
    "UefiArgs": "",
    "UefiPassword": ""
  },
  "Description": "BIOS Configuration Service",
```

```
  "Id": "BIOS",
  "Links": {
    "SoftwareImages": [
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF"
      },
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_BOARD"
      },
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_BSP"
      },
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_NIC"
      },
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_NODE"
      },
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_OFED"
      },
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_OS"
      },
      {
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_SYS_IMAGE"
      },
      {
```

```
        "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_UEFI"
      }
    ],
    "SoftwareImages@odata.count": 9
  },
  "Name": "BIOS Configuration",
  "ResetBiosToDefaultsPending": false
}
```

The following table explains each of the attributes listed in the code:

Attribute	Description
`Boot Partition Protection`	See description in section "System Configuration"
`CurrentUefiPassword`	See "Set Password" in section "System Configuration"
`DateTime`	See "Set RTC" in section "System Configuration"
`DefaultPasswordPolicy`	See "Password Settings" in section "System Configuration"
`Disable PCIe`	See description in section "System Configuration"
`Disable SPMI`	See description in section "System Configuration"
`Disable TMFF`	See description in section "System Configuration"
`Disable HEST`	See description in section "System Configuration"
`EmmcWipe`	See description in section "System Configuration"
`Enable 2nd eMMC`	See description in section "System Configuration"
`Enable OP-TEE`	See description in section "System Configuration"
`Enable SMMU`	See description in section "System Configuration"
`Field Mode`	See description in section "System Configuration"
`Host Privilege Level`	See "BlueField Modes" in section "System Configuration"

Attribute	Description
`Internal CPU Model`	See "BlueField Modes" in section "System Configuration"
`LegacyPasswordEnable`	See "Password Settings" in section "System Configuration"
`NicMode`	See "BlueField Modes" under section "System Configuration"
`NvmeWipe`	See description in section "System Configuration"
`OsArgs`	Arguments to pass to the OS kernel
`ResetEfiVars`	See "Reset EFI Variables" in section "System Configuration"
`SPCR UART`	See " Select SPCR UART " in section "System Configuration"
`UefiArgs`	Arguments to pass to the UEFI
`UefiPassword`	See "Set Password" in section "System Configuration"

Info

To change the configuration of any of these BIOS attributes, refer to section "Changing BIOS Attributes Value" in the *BMC Software User Manual*.

Example of Setting BIOS Attributes

The following is an example of fetching and setting a BlueField BIOS attribute:

1. Check UEFI attributes and their values by doing a GET on `Bios` URI. Look for the `Attributes` property.

```
$ curl -sk -X GET -u $BMC_USER:$BMC_PASSWORD
https://$BMC_IP/redfish/v1/Systems/Bluefield/Bios | jq
```

```
'.Attributes'
{
  "BootPartitionProtection": false,
  "CeThreshold": 5000,
  "CurrentUefiPassword": "",
  "DateTime": "2024-10-17T19:47:04Z",
  "DefaultPasswordPolicy": true,
  "DisableHEST": false,
  "DisableI2c1": false,
  "DisablePCIe": false,
  "DisableSPMI": false,
  "DisableTMFF": false,
  "EmmcWipe": false,
  "Enable2ndeMMC": false,
  "EnableDdr5600": false,
  "EnableOPTEE": false,
  "EnableSMMU": true,
  "FieldMode": false,
  "ForcePxeRetryDisable": false,
  "HostPrivilegeLevel": "Restricted",
  "InternalCPUModel": "Embedded",
  "L3CachePartitionLevel": 0,
  "LegacyPasswordEnable": false,
  "NicMode": "DpuMode",
  "NvmeWipe": false,
  "OsArgs": "",
  "ResetEfiVars": false,
  "SPCR_UART": "Disabled",
  "UefiArgs": "",
  "UefiPassword": ""
}
```

Note

For Security reasons, the `CurrentUefiPassword` and `UefiPassword` strings may appear as empty.

2. The following example updates the UEFI password. Perform PATCH to the pending `Bios` settings URI as follows:

```
curl -vk -X PATCH -d '{"Attributes":{"CurrentUefiPassword":
"$CURRENTPASSWD", "UefiPassword": "$NEWPASSWORD"}}' -u
$BMC_USER:$BMC_PASSWORD
https://<bmc_ip>/redfish/v1/Systems/SystemId/Bios/Settings |
jq '.Attributes'
```

Note

To update the password, both the current password and the new password (requesting) should be specified as demonstrated above. Otherwise, the change does not work. To modify other attributes no password is required.

3. To confirm whether the PATCH request is successful, perform a GET to the pending `Bios` settings URI:

```
curl -vk -X GET -u -u $BMC_USER:$BMC_PASSWORD
https://<bmc_ip>/redfish/v1/Systems/SystemId/Bios/Settings |
jq '.Attributes'
```

4. For requests to take effect, reboot BlueField. If the `CurrentUefiPassword` is correct, then the UEFI password is updated during the UEFI Redfish phase of boot.

> **Info**
>
> The UEFI password is only required to enter the UEFI menu using the serial console.

BlueField Platform Inventory

The NVIDIA® BlueField® networking platform (DPU or SuperNIC) provides inventory information in the `ComputerSystemCollection` schema. To identify the BlueField `ComputerSystem` instance, fetch the `ComputerSystemCollection` first.

BlueField devices are identified with the `SystemType` attribute `DPU`. The BlueField instance identifier value (`DPU.Embedded.1_NIC.Slot.2` in this case) differs from one server vendor to another but will uniquely identify BlueField in all cases.

The following is a simple example of fetching Redfish inventory information from a server's BMC:

```
root@localhost:~$ python3 /usr/local/bin/redfishtool.py -r
<bmc_ip> -u <USER> -p <PASSWORD> raw GET /redfish/v1/Systems/
{
    "@odata.context":
"/redfish/v1/$metadata#ComputerSystemCollection.ComputerSystemColle
    "@odata.id": "/redfish/v1/Systems",
    "@odata.type":
"#ComputerSystemCollection.ComputerSystemCollection",
    "Description": "Collection of Computer Systems",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Systems/System.Embedded.1"
        },
        {
            "@odata.id":
"/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2"
```

```
        }
    ],
    "Members@odata.count": 2,
    "Name": "Computer System Collection"
}

root@localhost:~$ python3 /usr/local/bin/redfishtool.py -r
<bmc_ip> -u <USER> -p <PASSWORD> raw GET
/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2
{
    "@odata.context":
"/redfish/v1/$metadata#ComputerSystem.ComputerSystem",
    "@odata.id": "/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2",
    "@odata.type": "#ComputerSystem.v1_12_0.ComputerSystem",
    "Actions": {
        "#ComputerSystem.Reset": {
            "target":
"/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2/Actions/ComputerSys
            "ResetType@Redfish.AllowableValues": [
                "ForceRestart",
                "Nmi"
            ]
        }
    },
    "Bios": {
        "@odata.id":
"/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2/Bios"
    },
    "BiosVersion": null,
    "Boot": {
        "BootOptions": {
            "@odata.id":
"/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2/BootOptions"
        },
        "BootOrder": [],
        "BootOrder@odata.count": 0,
```

```
        "BootSourceOverrideEnabled": null,
        "BootSourceOverrideMode": null,
        "BootSourceOverrideTarget": null,
        "UefiTargetBootSourceOverride": null,
        "BootSourceOverrideTarget@Redfish.AllowableValues": []
    },
    "Description": "DPU System",
    "Id": "DPU.Embedded.1_NIC.Slot.2",
    "Manufacturer": "DELL",
    "Model": "NVIDIA Bluefield-2 25GbE 2p Crypto DPU",
    "Name": "DPU System",



    "Oem": {
        "Dell": {
            "@odata.type":
"#DellComputerSystem.v1_1_0.DellComputerSystem",
            "DPUConfig": {
                "FQDD": "DPU.Embedded.1:NIC.Slot.2",
                "BootStatus": "OSBooting",
                "DPUBootSynchronization": "Enabled",
                "DPUTrust": "Enabled",
                "IdenticalSBDF": [
                    "0:23:0:0",
                    "0:23:0:1"
                ],
                "LastResetReason": null,
                "OSName": null,
                "OSReadyTimeout": 20,
                "OSInstallationTimeout": 30,
                "OSVersion": null,
                "OSVendor": null,
                "OSStatus": "Unknown",
                "Slot": "2",
                "PCIeSlotState": "Enabled",
```

```
                "PostCode": null,
                "VendorID": "0x15B3",
                "DeviceID": "0xA2D6",
                "SubVendorID": "0x15B3",
                "SubDeviceID": "0x0129"
            },
            "Name": "DPUConfig",
            "Id": "DPU.Embedded.1_NIC.Slot.2"
        }
    },
    "PartNumber": "JNDCMX01",
    "SecureBoot": {
        "@odata.id":
"/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2/SecureBoot"
    },
    "SerialNumber": "IL740311A5000A",
    "SKU": "0JNDCM",
    "Status": {
        "Health": "Ok",
        "HealthRollup": "Ok",
        "State": "Enabled"
    },
    "SystemType": "DPU",
    "UUID": "ec6dd921-882a-ec11-8000-08c0eb5180ba",
    "@Redfish.Settings": {
        "@odata.context":
"/redfish/v1/$metadata#Settings.Settings",
        "@odata.type": "#Settings.v1_3_3.Settings",
        "SettingsObject": {
            "@odata.id":
"/redfish/v1/Systems/DPU.Embedded.1_NIC.Slot.2/Settings"
        }
    }
}
```

Boot Override

This example demonstrates how to boot a BlueField Platform while overriding the existing boot options and using HTTP boot to obtain the image.

Check the current boot override settings by doing a GET on `ComputerSystem` schema. Look for the Boot property.

```
curl -vk -X GET -u "user:password"
https://<bmc_ip>/redfish/v1/Systems/SystemId/ | python3 -m
json.tool
{
...
"Boot": {
        "BootNext": "",
        "BootOrderPropertySelection": "BootOrder",
        "BootSourceOverrideEnabled": "Disabled",
        "BootSourceOverrideMode": "UEFI",
        "BootSourceOverrideTarget": "None",
        "UefiTargetBootSourceOverride": "None",
        .....
        },
  ....
  "BootSourceOverrideEnabled@Redfish.AllowableValues": [
            "Once",
            "Continuous",
            "Disabled"
        ],
    "BootSourceOverrideTarget@Redfish.AllowableValues": [
            "None",
            "Pxe",
            "UefiHttp",
            "UefiShell",
            "UefiTarget",
            "UefiBootNext"
        ],
```

```
  ....
}
```

The sample output above shows the `BootSourceOverrideEnabled` property is `Disabled` and `BootSourceOverrideTarget` is `None`. The `BootSourceOverrideMode` property should always be set to `UEFI`. Allowable values of `BootSourceOverrideEnabled` and `BootSourceOverrideTarget` are defined in the meta-data `BootSourceOverrideEnabled@Redfish.AllowableValues` and `BootSourceOverrideTarget@Redfish.AllowableValues` respectively.

To perform boot override, you must perform a PATCH to pending settings URI:

```
curl -vk -X PATCH -d '{"Boot":
{"BootSourceOverrideEnabled":"Once",
"BootSourceOverrideMode":"UEFI", "BootSourceOverrideTarget":
"UefiHttp", "HttpBootUri":"http://<HTTP-Server-Ip>/Image.iso"}}'
-u "user:password"
https://<bmc_ip>/redfish/v1/Systems/SystemId/Settings | python3 -
m json.tool
```

After performing the above PATCH successfully, reboot the BlueField Platform. Once UEFI has completed, check whether the settings are applied by performing a GET on `ComputerSystem` schema.

Note that the `HttpBootUri` property is parsed by the Redfish server and the URI is presented to BlueField as part of DHCP lease when BlueField performs the HTTP boot.

```
curl -vk -X GET -u "user:password"
https://<bmc_ip>/redfish/v1/Systems/SystemId/ | python3 -m
json.tool
{
...
"Boot": {
```

```
        "BootNext": "",
        "BootOrderPropertySelection": "BootOrder",
        "BootSourceOverrideEnabled": "Once",
        "BootSourceOverrideMode": "UEFI",
        "BootSourceOverrideTarget": "UefiHttp",
        "UefiTargetBootSourceOverride": "None",
        .....
        },
    .....
}
```

After confirming the settings are applied (see PATCH properties above), reboot BlueField for the settings to take effect. If `BootSourceOverrideEnabled` is set to `Once`, boot override is disabled and any related properties are reset to their former values to avoid repetition. If it is set to `Continuous`, then on every reboot, BlueField would keep performing boot override (`HTTPBoot`).

Boot Order

The following is an example of changing the boot order and fetching the details of a boot option.

1. Check the current boot order by doing GET on the `ComputerSystem` schema. Look for the `BootOrder` attribute under the `Boot` property.

2. Get the details of a particular entity in the `BootOrder` array by performing a GET to the respective BootOption URL. For example, to get details of `Boot0006`, run:

```
curl -vk -X GET -u "user:password"
https://<bmc_ip>/redfish/v1/Systems/SystemId/BootOptions/Boot0
| python3 -m json.tool

{
    "@odata.type": "#BootOption.v1_0_3.BootOption",
```

```
    "@odata.id":
"/redfish/v1/Systems/SystemId/BootOptions/Boot0006",
    "Id": "Boot0006",
    "BootOptionEnabled": true,
    "BootOptionReference": "Boot0006",
    "DisplayName": "UEFI HTTPv6 (MAC:B8CEF6B8A006)",
    "UefiDevicePath":
"PciRoot(0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/Pci(0x0,0
}
```

3. To change the boot order, the entire `BootOrder` array must be PATCHed to the pending settings URI. For the above example of the `BootOrder` array, if you intend to have `Boot0006` at the beginning of the array, then the PATCH operation is as follows.

```
curl -vk -X PATCH -d '{ "Boot": { "BootOrder": [ "Boot0006",
"Boot0017", "Boot0001", "Boot0002", "Boot0003", "Boot0004",
"Boot0005", "Boot0007", ] }}' -u "user:password"
https://<bmc_ip>/redfish/v1/Systems/SystemId/Settings |
python3 -m json.tool
```

> **Note**
>
> Updating the `BootOrder` array results in a permanent boot order change (persistent across reboots).

After a successful PATCH, reboot BlueField and check if the settings were applied by doing a GET on the `ComputerSystem` schema. If the `BootOrder` array is updated as intended, then the settings were applied and the BlueField Platform should boot as per the order in proceeding cycles.

Passing Arguments from BMC to DPU via Redfish

To pass arguments from BMC Redfish to UEFI and DPU OS/Kernel, there are two new attributes introduced in the BIOS Redfish schema (`redfish/v1/Systems/SYSTEM-ID/Bios/`):

- UefiArgs – This attribute can be used to pass arguments to UEFI from BMC Redfish
- OsArgs – This attribute can be used to pass arguments to OS/Kernel from BMC Redfish

UefiArgs

This attribute is write-only and non-persistent. Meaning, user can write to this attribute (via pending setting URI) and UEFI redfish client will process it accordingly, but user can't read the current value; it is always presented as an empty string in the Bios current setting URI.

Also, the arguments passed are stored in a volatile memory in UEFI so it won't be available in the next reboot after UEFI redfish client has processed it.

Currently, user needs to provide the value of this attribute in a specific format (key=value;key2=value2). The character ';' (semi-colon) is used as a separator between different key/value pairs.

The only currently supported keys are:

- `tftp_ip`
- `dhcpv6_duid`
- `hide_dpubmc_credentials`

Therefore a correct value for this attribute is of the form as mentioned below. Note that it is not required to mention all the key/value pairs.

```
"UefiArgs" :  "tftp_ip=xxx.xxx.xxx.xxx;dhcpv6_duid=XXXX;hide_dpubmc_credentials=true"
```

Where:

- `xxx.xxx.xxx.xxx` is an IPv4 address.
- `XXXX` can be either `LLT` or `UUID`

> **Note**
>
> The key `hide_dpubmc_credentials` may be set to `true` (default) or `false`.

tftp_ip

The `tftp_ip` key holds the IPv4 address which users intend the UEFI to use when performing PXEv4 boot instead of using the one provided by the DHCP server.

The change is persistent across reboots.

The following is an example use case for `tftp_ip`:

1. Perform PATCH `UefiArgs` to `Bios/Settings`.
2. Request change boot order/boot override to perform PXE boot.
3. Reboot BlueField for the UEFI Redfish client to process pending Redfish requests.
4. During PXE boot, the TFTP IP mentioned in `UefiArgs` is used (instead of the one mentioned in the DHCP server configuration file).

dhcpv6_uuid

The `dhcpv6_uuid` key can hold a Boolean value of either `LLT` or `UUID`, representing the unique identifier type used in DHCPv6 requests initiated by UEFI. This unique identifier allows for distinct system identification.

Note that the user is only setting the type of identifier, not the actual identifier value. The identifier's value is automatically generated based on the selected type, then stored and used as needed. For testing, users can use tcpdump to capture packets when triggering

an HTTPv6 or PXEv6 network boot in UEFI and examine the DHCPv6 request to verify the DUID type and value.

The default DUID type is UUID on BlueField platform.

hide_dpubmc_credentials

The `hide_dpubmc_credentials` key accepts a Boolean value (`true` or `false`, case insensitive) and defaults to `false`. When set to `true`, it hides the DPU BMC credential details from the OS.

You can verify credential visibility in the OS by checking for the following EFI variables from the DPU OS.

When this key is `true`, these files will not be present:

- `/sys/firmware/efi/efivars/DPUBMCPassword-75a6cbf6-148c-487f-9ffa-7dad26e15801`
- `/sys/firmware/efi/efivars/DPUBMCUsername-75a6cbf6-148c-487f-9ffa-7dad26e15801`

Setting the value to `true` in Redfish takes effect after BlueField reboot. However, if changed from `true` to `false`, an additional reboot is required since the unhide operation would not apply immediately due to the Redfish runtime sequence.

This setting persists across reboots.

OsArgs

This attribute is write-only and non-persistent, with a maximum length of 8192 bytes (8 KB).

Info

It can contain any arbitrary string, as there are no format restrictions.

The purpose of `OsArgs` is to pass this string to the OS/Kernel during the boot cycle. The UEFI Redfish client processes pending BIOS requests and transfers the string to the BFCF ACPI table, which the OS/Kernel can then access.

To read the BFCF ACPI table from the OS, users can use tools like `acpidump` (if `acpica-tools` is installed) or `bfcfg` (available here).

Example of using `bfcfg`:

```
bfcfg --dump-osarg
```