SNIA Swordfish™ Hands-On Lab

Fall 2022 Edition
Richelle Ahlvers

This page intentionally left blank.

Abstract

In the SNIA Swordfish™ Hands-on Lab, Fall 2022 Edition, participants will get an introduction to three different storage configurations, each instrumented in SNIA Swordfish. The lab will allow the participants to see how each device is modeled, interact with an emulated instance of each storage device type, and perform typical configuration actions.

Details

Pre-work:

Prior to the lab, the participants will receive access information for the HOL session, which includes a Zoom link, as well as VPN access information for the SNIA Innovations Lab. The participants will be asked to confirm they can successfully access the lab via VPN prior to the start of the lab session.

HOL Session:

For the HOL Session, the Participants will join a virtual Zoom session. During this session:

Participants will be greeted by the HOL lead, who will describe the basic outline of the activity and its goals, and then lead the participants through the following activities.

1. (5-10 minutes) The HOL lead will provide a short overview of SNIA Swordfish objects that participants should look for, an overview of the three device configurations used as part of the lab, and the tools available on the systems for use.

2. (15 minutes) The first activity will be to navigate through the basic structure and application of Swordfish to an NVMe "EBOF" – an "ethernet-attached bunch of flash" enclosure and contained drives. The enclosure is an Ethernet front-end attach configuration (has an embedded switch) with a set of ethernet-attached drives inside. Each IP-attached drive has a complete Swordfish storage model – Storage object, controller, volume, and drive, along with its network interface modeled under the chassis. The connectivity is modeled under the Fabric/Ethernet, and the access is modeled under Fabric/NVMeoF.

These devices do not support much configuration, other than in the networking configuration and access; this activity is largely navigating to understand the configuration and model.

3. (15 minutes) The next activity will be to navigate through the structure and application of Swordfish to an external storage array which supports replication. The participant will navigate around to understand the system. After navigating around the system, the participant will find StoragePool1, check the available capacity, and then generate a request to create a new volume from this storage pool.

4. (15 minutes) The final activity will be to navigate through an external array which has an NVMe front end. This system combines elements seen in the first two activities – NVMe behaviors, as well as elements of external block storage devices. After familiarizing themselves with the configuration, the participant will perform an NVMe-specific action: configuring an additional host access to a namespace (aka, a volume).

Configuration Overview

All the HOL exercises will use one, two, or three of the Swordfish service instances running on a Windows VM instance.

Tools on Windows VM:

- PostMan – Rest client
- WSL (user “hol” password “hol”)

Each Swordfish service is an emulation of an actual Swordfish system management interface. All emulated systems are configured with no authorization required.

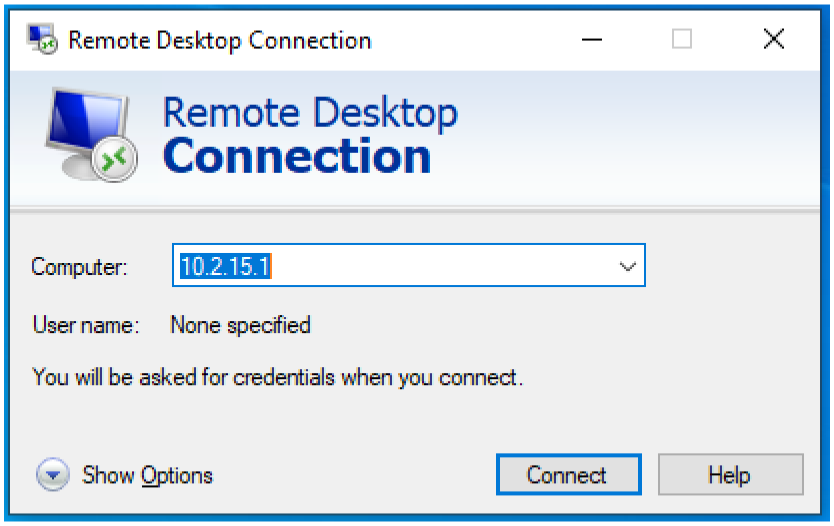
System 1: An EBOF (Ethernet Bunch of Flash)
Swordfish interface: http://localhost:4000/redfish/v1

System 2: An external storage array, configured with local replication.
Swordfish interface: http://localhost:6000/redfish/v1

System 3: External storage array, with an NVMe front end; has NVMe/TCP connectivity configured.
Swordfish interface: http://localhost:8000/redfish/v1

System Access

Connect via RDC to the system provided, using the IP address (“10.2.15.1” used in this example):

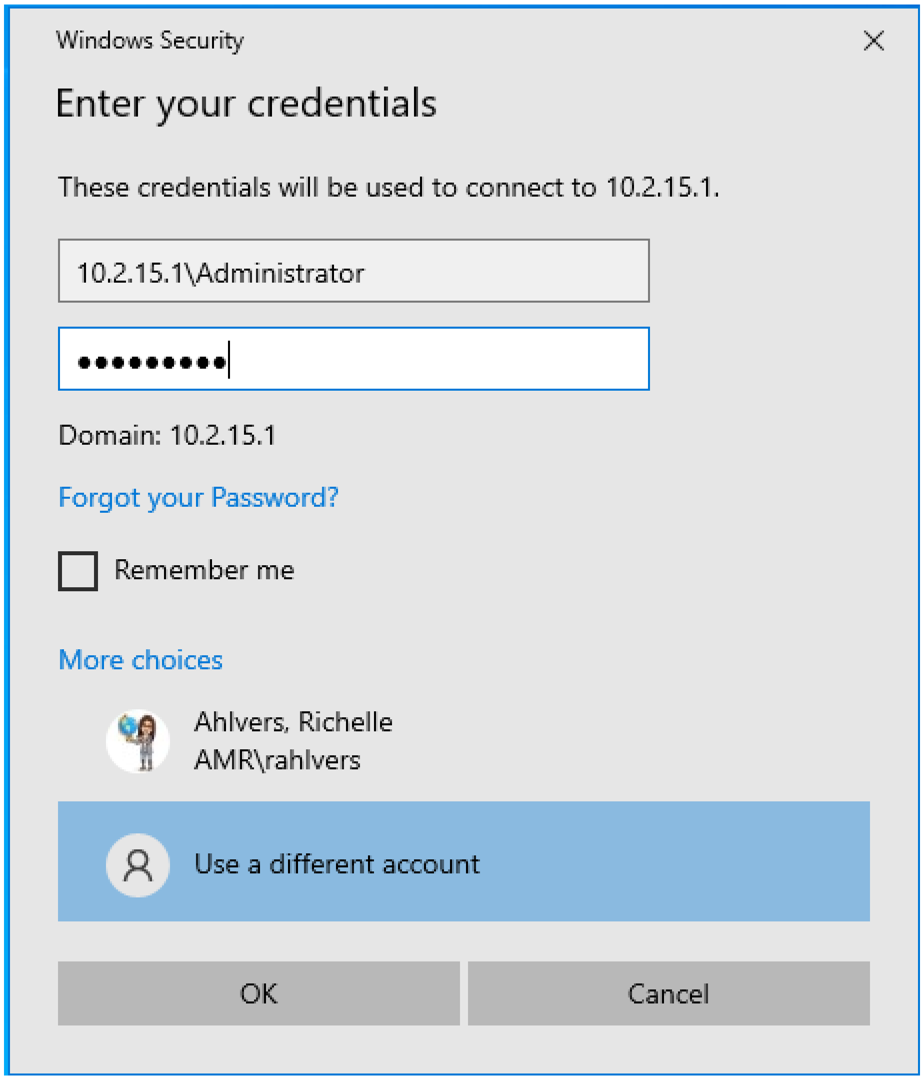

Remote Desktop Connection

Computer: 10.2.15.1

User name: None specified

You will be asked for credentials when you connect.

Select “Use a Different Account”, and login with Administrator and the provided password:


Windows Security

Enter your credentials

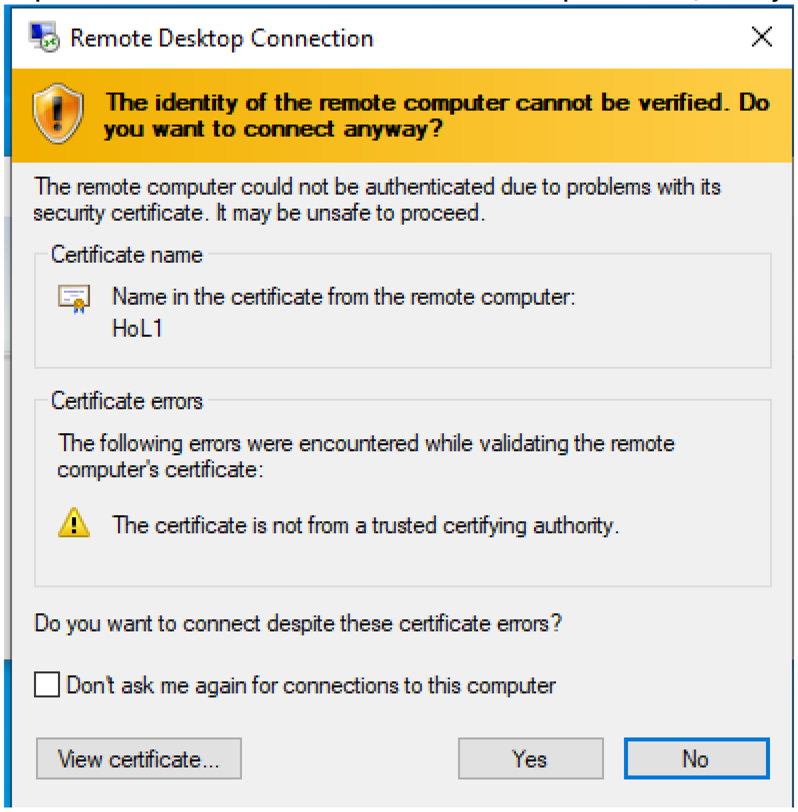
These credentials will be used to connect to 10.2.15.1.

10.2.15.1\Administrator

●●●●●●●●●

Domain: 10.2.15.1

Accept the untrusted certificate notice to proceed (click yes).



Remote Desktop Connection

The identity of the remote computer cannot be verified. Do you want to connect anyway?

The remote computer could not be authenticated due to problems with its security certificate. It may be unsafe to proceed.

Certificate name

Name in the certificate from the remote computer:
HoL1

Certificate errors

The following errors were encountered while validating the remote computer's certificate:

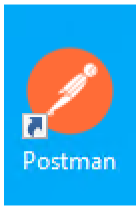
The certificate is not from a trusted certifying authority.

Do you want to connect despite these certificate errors?

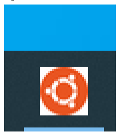
☐ Don't ask me again for connections to this computer

Tools:

On the desktop you'll see the "Postman" shortcut.



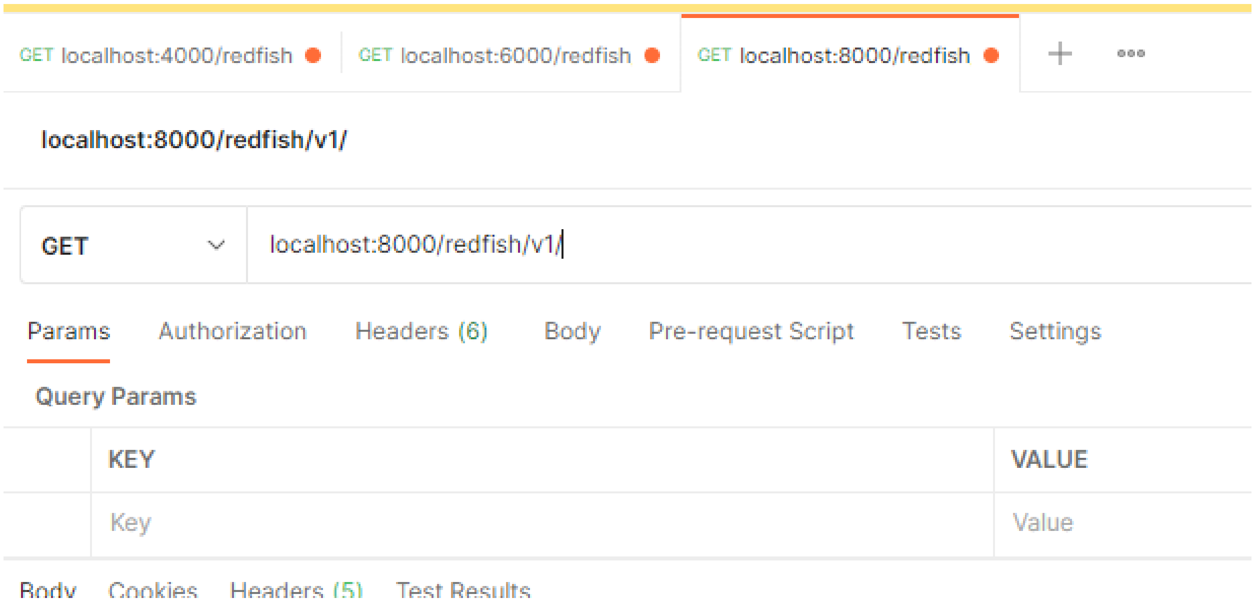
There's also a link to the Ubuntu / WSL (Windows Subsystem for Linux) on the taskbar.



Getting Started:

1. Open a WSL (Ubuntu) window. (This will start the Swordfish service emulators in the background).

2. Open a Postman window. There should be three "GET" links auto-populated for you.



GET localhost:4000/redfish
GET localhost:6000/redfish
GET localhost:8000/redfish
localhost:8000/redfish/v1/
GET
localhost:8000/redfish/v1/
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
Key
Value
Body
Cookies
Headers (5)
Test Results

Try these out. These will take you to each of the three Swordfish "services" we will use in this lab.

The services are differentiated by the port used to access them:

- Localhost:4000 == EBOF

- Localhost:6000 == external array

- Localhost:8000 == NVMe/TCP array

To access Redfish/Swordfish services, append "/redfish/v1" to the name or IP address (plus port) of the system.

Exercise 1: Understanding Swordfish in an EBOF

Purpose:

This exercise will familiarize the participant with the Swordfish model when applied to an NVMe device. This includes many elements: logical storage models, physical models, network configurations, access rights management, and connectivity management. It also highlights the application of the Swordfish model to NVMe devices.

Overview:

The following activities are covered in this exercise:
Navigate through the basic structure and application of Swordfish to an NVMe "EBOF" – an "ethernet-attached bunch of flash" enclosure and contained drives. The enclosure is an Ethernet front-end attach configuration (has an embedded switch) with a set of ethernet-attached drives inside. Each IP-attached drive has a complete Swordfish storage model – Storage object, controller, volume, and drive, along with its network interface modeled under the chassis. The connectivity is modeled under the Fabric/Ethernet, and the access is modeled under Fabric/NVMeoF.

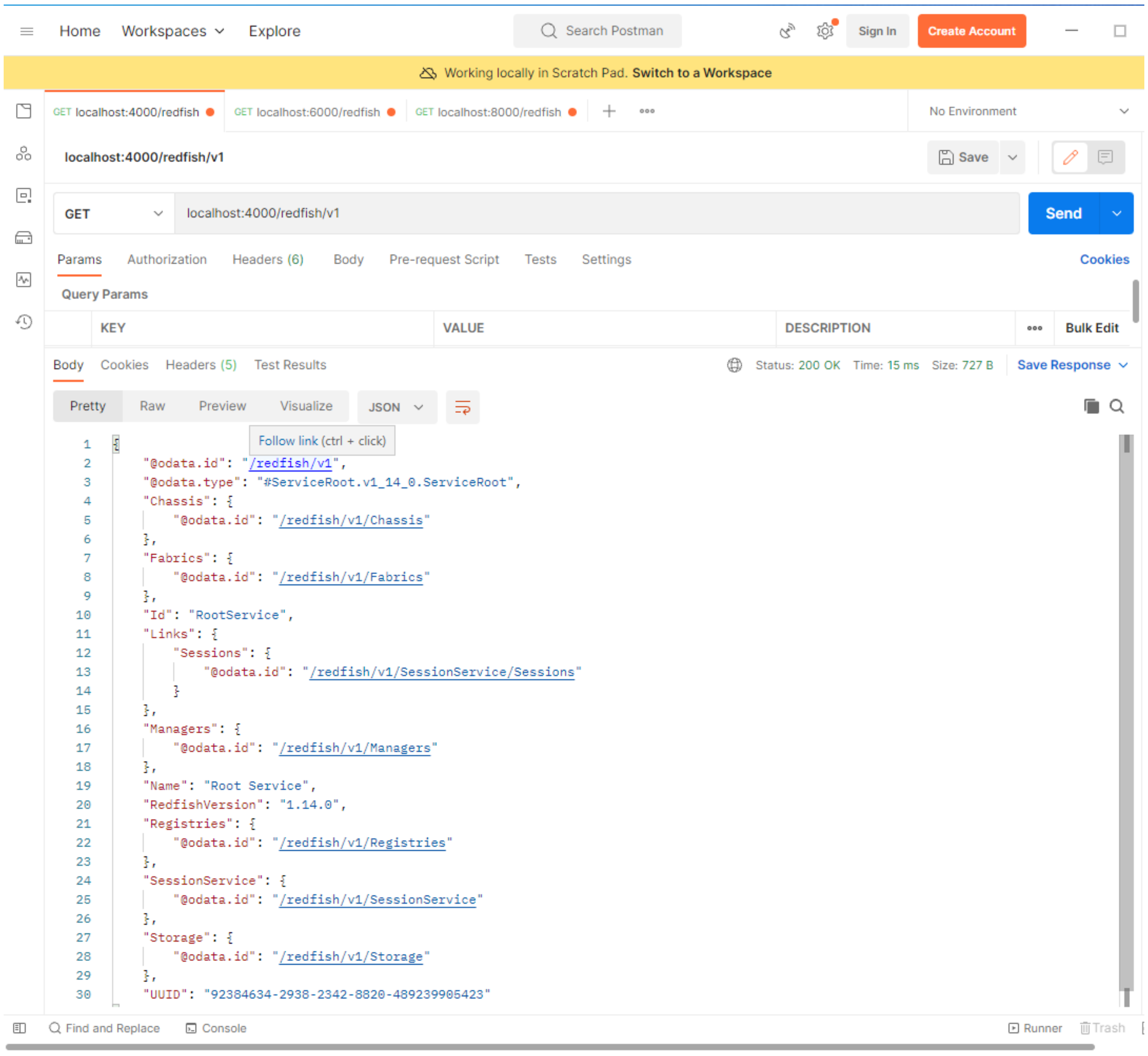
- Identify the logical Storage components:
  - Storage, Volume (namespace), controllers
- Identify networking components:
  - Chassis/NetworkAdapters
  - Fabric/Ethernet
    - Switches
- Identify NVMe-specific components using common objects
  - Logical controllers: Storage/Controllers
  - Fabric/NVMeoF – NVMe access rights management
  - NVMe Namespace – Storage/Volume

Getting Started:

Using Postman, go to the ServiceRoot of the EBOF Swordfish Service:

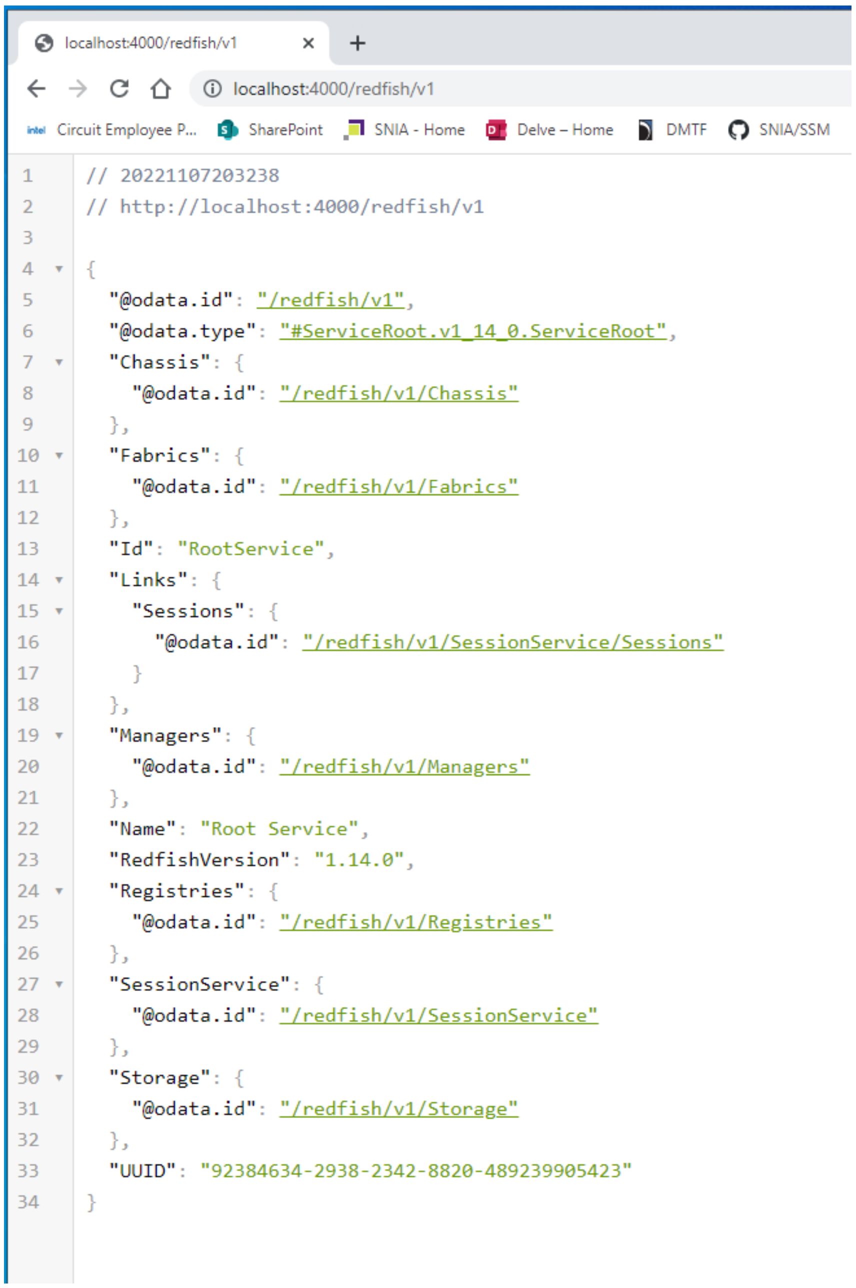
Select the REST command type in the drop-down box (GET), and type the URI in the gray box. (localhost:4000/redfish/v1). Click "Send"

The view below shows the Body view in “Pretty” mode. Take a minute to check out some of the other options. If you have questions about any of the other views, or options, ask your instructor.



```
{
    "@odata.id": "/redfish/v1",
    "@odata.type": "#ServiceRoot.v1_14_0.ServiceRoot",
    "Chassis": {
        "@odata.id": "/redfish/v1/Chassis"
    },
    "Fabrics": {
        "@odata.id": "/redfish/v1/Fabrics"
    },
    "Id": "RootService",
    "Links": {
        "Sessions": {
            "@odata.id": "/redfish/v1/SessionService/Sessions"
        }
    },
    "Managers": {
        "@odata.id": "/redfish/v1/Managers"
    },
    "Name": "Root Service",
    "RedfishVersion": "1.14.0",
    "Registries": {
        "@odata.id": "/redfish/v1/Registries"
    },
    "SessionService": {
        "@odata.id": "/redfish/v1/SessionService"
    },
    "Storage": {
        "@odata.id": "/redfish/v1/Storage"
    },
    "UUID": "92384634-2938-2342-8820-489239905423"
```

Note: you can also make the same query directly from a web browser and you will see the same information returned. This is because REST is a web semantic.



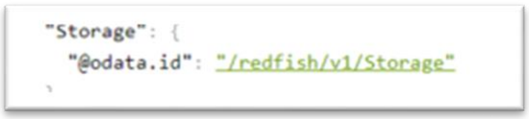
```
// 20221107203238
// http://localhost:4000/redfish/v1

{
  "@odata.id": "/redfish/v1",
  "@odata.type": "#ServiceRoot.v1_14_0.ServiceRoot",
  "Chassis": {
    "@odata.id": "/redfish/v1/Chassis"
  },
  "Fabrics": {
    "@odata.id": "/redfish/v1/Fabrics"
  },
  "Id": "RootService",
  "Links": {
    "Sessions": {
      "@odata.id": "/redfish/v1/SessionService/Sessions"
    }
  },
  "Managers": {
    "@odata.id": "/redfish/v1/Managers"
  },
  "Name": "Root Service",
  "RedfishVersion": "1.14.0",
  "Registries": {
    "@odata.id": "/redfish/v1/Registries"
  },
  "SessionService": {
    "@odata.id": "/redfish/v1/SessionService"
  },
  "Storage": {
    "@odata.id": "/redfish/v1/Storage"
  },
  "UUID": "92384634-2938-2342-8820-489239905423"
}
```

Next Step: Navigating Through the Service

There are two ways to move through the model in Postman – you can click on a link (double-click, or click and press “Send”), or type a value directly into the bar, as above.

Find the link to the Storage and navigate to it.

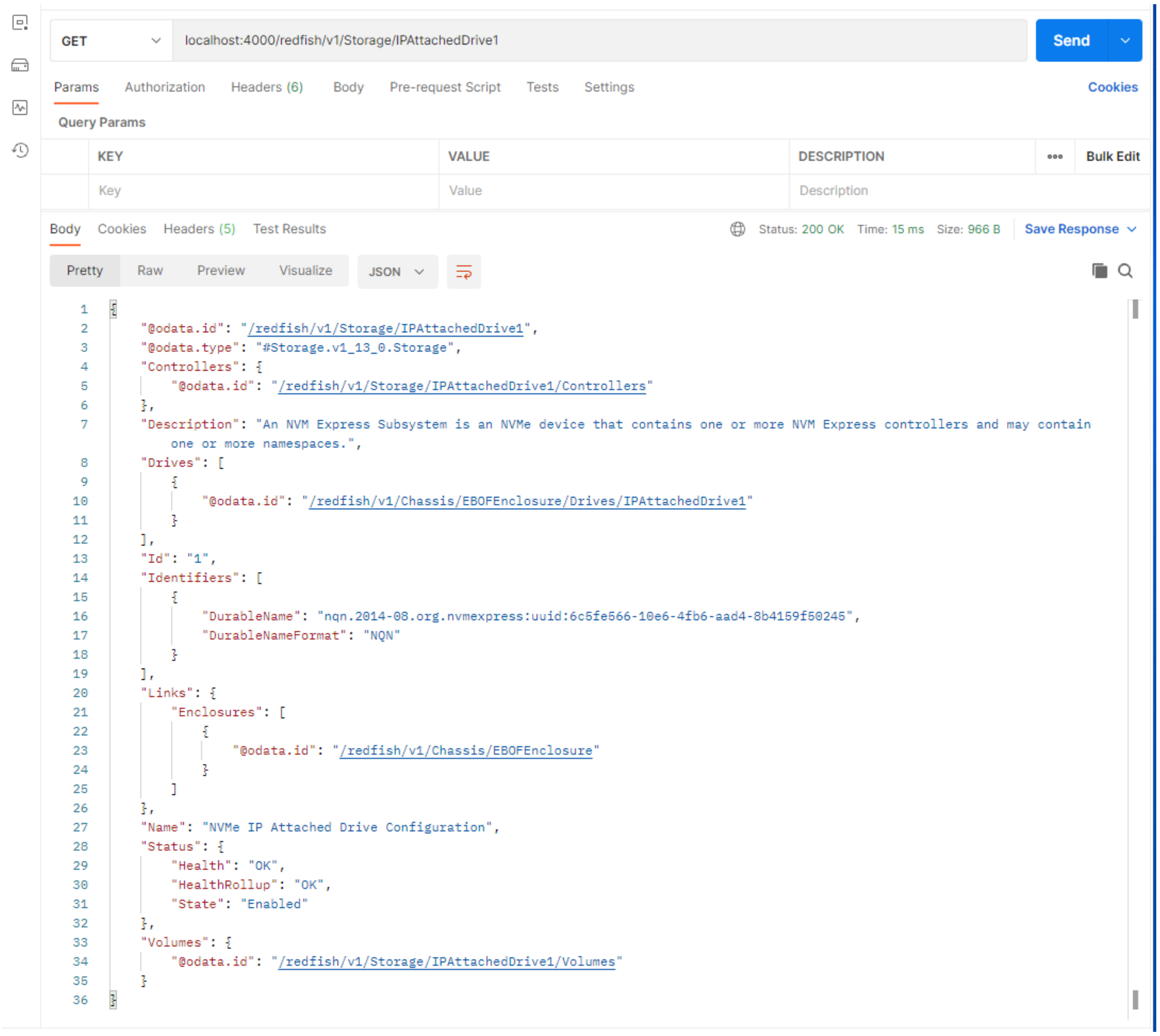


```
"Storage": {
  "@odata.id": "/redfish/v1/Storage"
```

If you get stuck, you can always to the nav bar and reset to the service root (/redfish/v1).

Identify the Components of the Storage Model

From the Storage collection, navigation into one of the Storage objects.


GET localhost:4000/redfish/v1/Storage/IPAttachedDrive1

Status: 200 OK Time: 15 ms Size: 966 B

```
{
    "@odata.id": "/redfish/v1/Storage/IPAttachedDrive1",
    "@odata.type": "#Storage.v1_13_0.Storage",
    "Controllers": {
        "@odata.id": "/redfish/v1/Storage/IPAttachedDrive1/Controllers"
    },
    "Description": "An NVM Express Subsystem is an NVMe device that contains one or more NVM Express controllers and may contain
        one or more namespaces.",
    "Drives": [
        {
            "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/Drives/IPAttachedDrive1"
        }
    ],
    "Id": "1",
    "Identifiers": [
        {
            "DurableName": "nqn.2014-08.org.nvmexpress:uuid:6c5fe566-10e6-4fb6-aad4-8b4159f50245",
            "DurableNameFormat": "NQN"
        }
    ],
    "Links": {
        "Enclosures": [
            {
                "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure"
            }
        ]
    },
    "Name": "NVMe IP Attached Drive Configuration",
    "Status": {
        "Health": "OK",
        "HealthRollup": "OK",
        "State": "Enabled"
    },
    "Volumes": {
        "@odata.id": "/redfish/v1/Storage/IPAttachedDrive1/Volumes"
    }
}
```

Once there, you can see the properties and additional sub-components of the storage object.

Navigate through the tree, looking at the controllers and volumes. Note by both their URI structures, as well as from their definitions, that these look like they are both direct components of this objects.

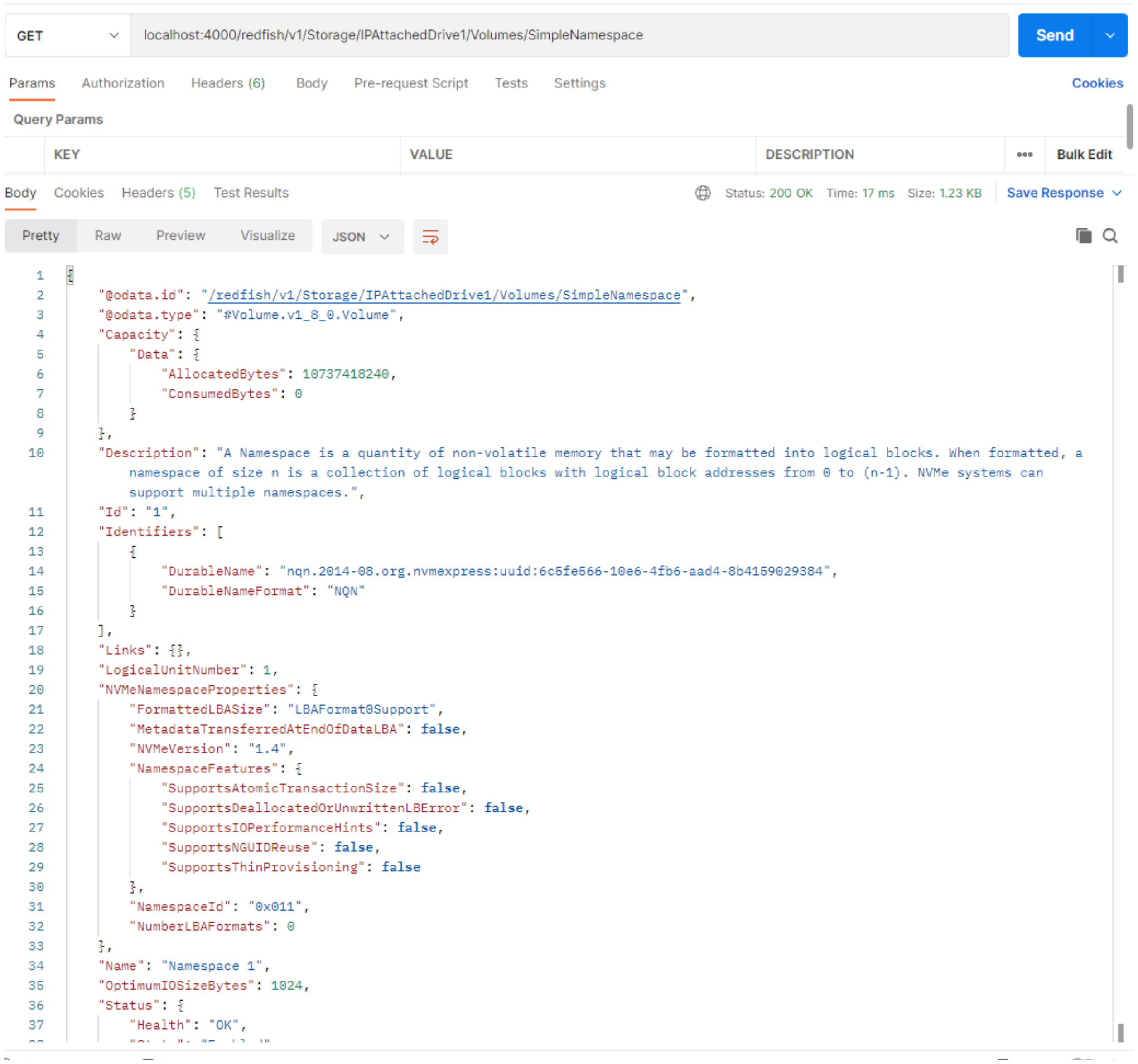
The Drives ID, however, does not. It is a link to another place in the system.

There is also a links section, which points to related items.

Look at the Namespace (Volume)

Navigate to the Volume from this Storage object. Since this is an NVMe device, this has many unique properties. These are largely contained in a unique section called "NVMeNamespaceProperties". NVMe unique behaviors are harder to see; these may manifest in specific properties implemented, relationships to required objects, etc..

The best way to determine what should be implemented is to follow required profiles. These are then advertised as Features so clients can tell what to expect. This device should advertise itself as an "NVMe EBOF".



localhost:4000/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNamespace

Status: 200 OK Time: 17 ms Size: 1.23 KB

```
{
    "@odata.id": "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNamespace",
    "@odata.type": "#Volume.v1_8_0.Volume",
    "Capacity": {
        "Data": {
            "AllocatedBytes": 10737418240,
            "ConsumedBytes": 0
        }
    },
    "Description": "A Namespace is a quantity of non-volatile memory that may be formatted into logical blocks. When formatted, a
        namespace of size n is a collection of logical blocks with logical block addresses from 0 to (n-1). NVMe systems can
        support multiple namespaces.",
    "Id": "1",
    "Identifiers": [
        {
            "DurableName": "nqn.2014-08.org.nvmexpress:uuid:6c5fe566-10e6-4fb6-aad4-8b4159029384",
            "DurableNameFormat": "NQN"
        }
    ],
    "Links": {},
    "LogicalUnitNumber": 1,
    "NVMeNamespaceProperties": {
        "FormattedLBASize": "LBAFormat0Support",
        "MetadataTransferredAtEndOfDataLBA": false,
        "NVMeVersion": "1.4",
        "NamespaceFeatures": {
            "SupportsAtomicTransactionSize": false,
            "SupportsDeallocatedOrUnwrittenLBError": false,
            "SupportsIOPerformanceHints": false,
            "SupportsNGUIDReuse": false,
            "SupportsThinProvisioning": false
        },
        "NamespaceId": "0x011",
        "NumberLBAFormats": 0
    },
    "Name": "Namespace 1",
    "OptimumIOSizeBytes": 1024,
    "Status": {
        "Health": "OK",
```

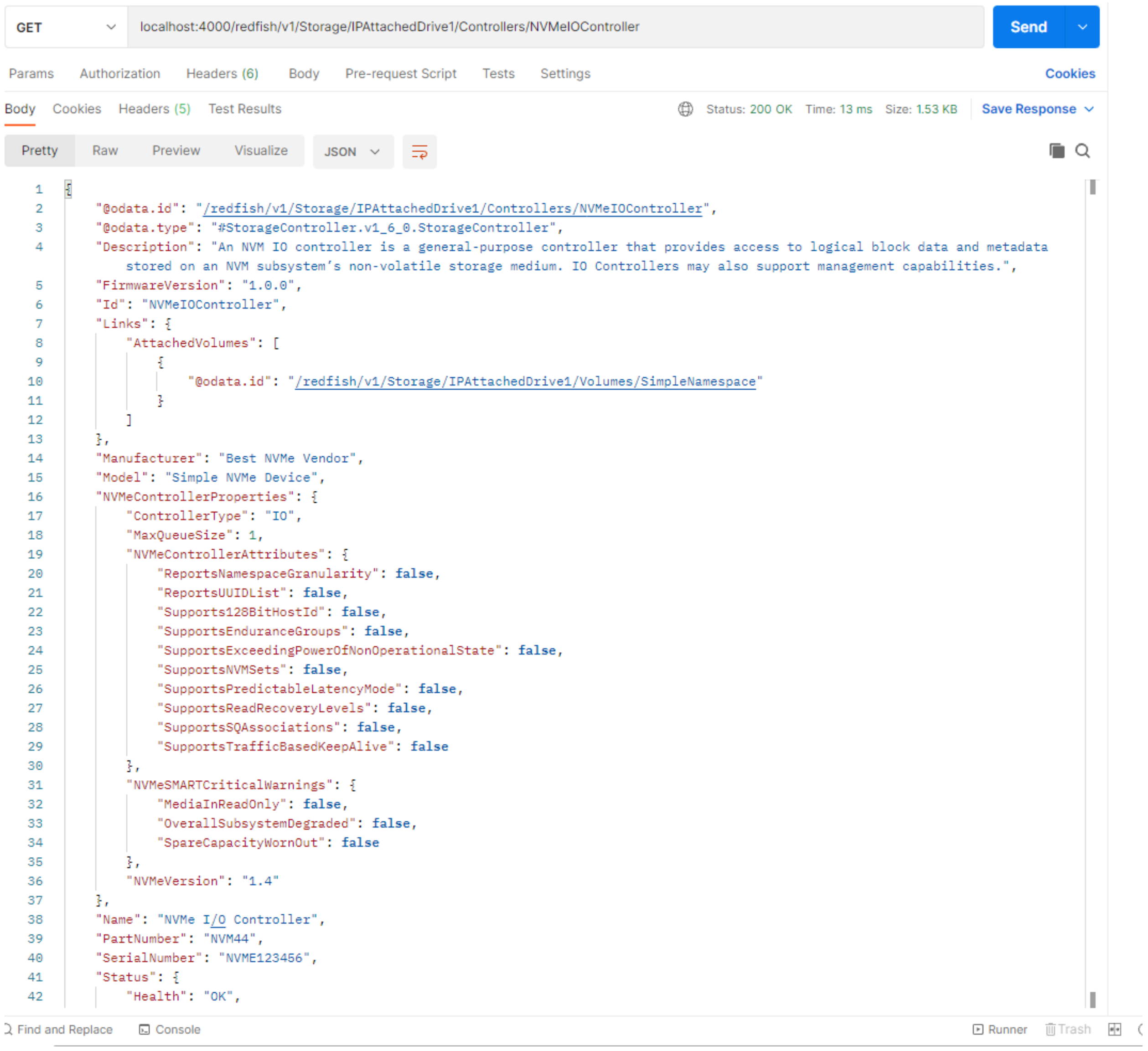
Find the related Controllers

NVMe uses the notion of “controller” differently from traditional storage. These are “logical” controllers that indicate things like connections of volumes (namespaces) to hosts.

This particular namespace doesn’t show us its related controllers; that would be a great enhancement request to this system.

Instead, to find the controller, go back up to the Storage entity, and find the controller.

(You can see the controller has the corresponding link to the namespace.)



localhost:4000/redfish/v1/Storage/IPAttachedDrive1/Controllers/NVMeIOController

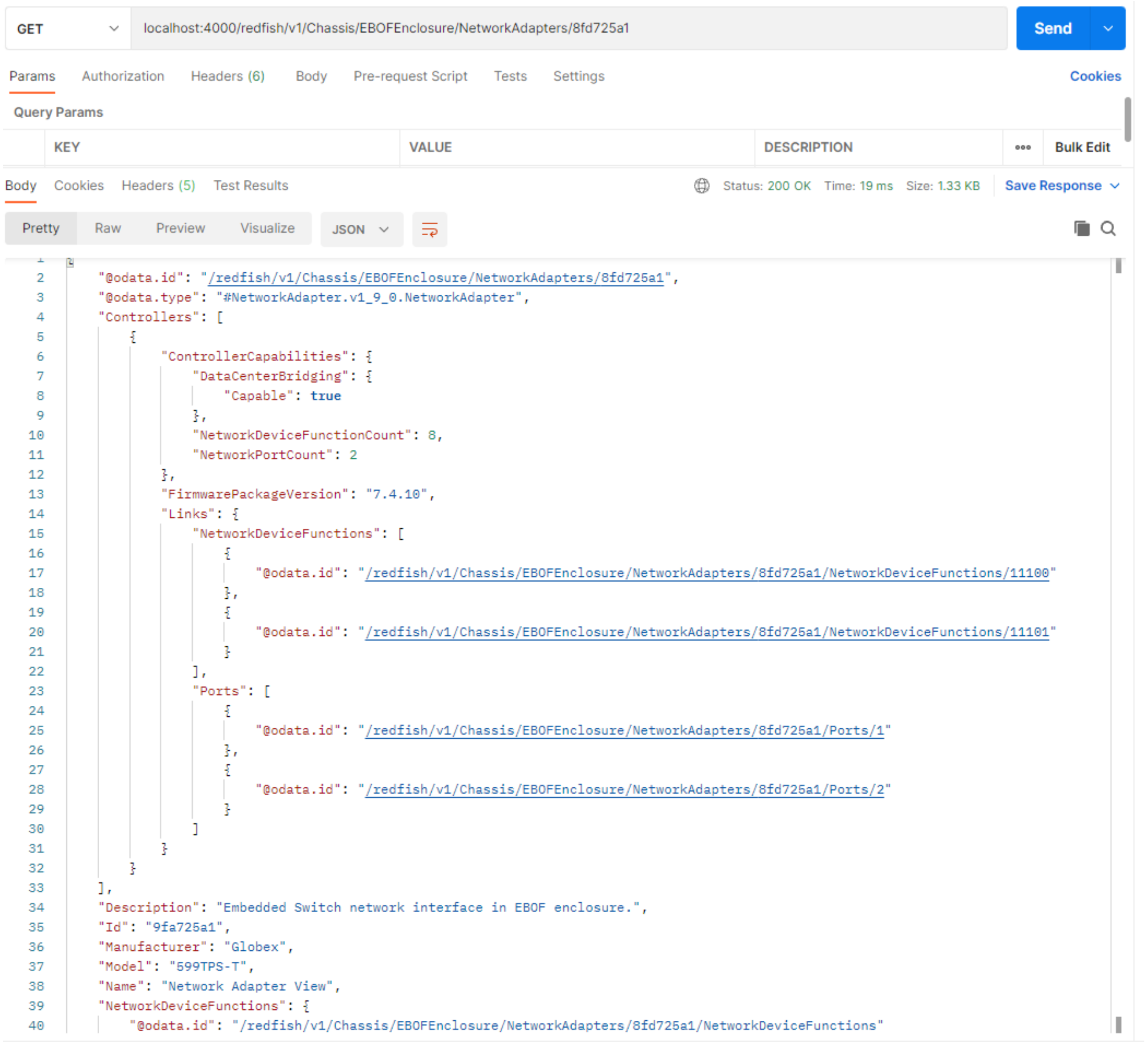
Status: 200 OK Time: 13 ms Size: 1.53 KB

```
{
    "@odata.id": "/redfish/v1/Storage/IPAttachedDrive1/Controllers/NVMeIOController",
    "@odata.type": "#StorageController.v1_6_0.StorageController",
    "Description": "An NVM IO controller is a general-purpose controller that provides access to logical block data and metadata
        stored on an NVM subsystem’s non-volatile storage medium. IO Controllers may also support management capabilities.",
    "FirmwareVersion": "1.0.0",
    "Id": "NVMeIOController",
    "Links": {
        "AttachedVolumes": [
            {
                "@odata.id": "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNamespace"
            }
        ]
    },
    "Manufacturer": "Best NVMe Vendor",
    "Model": "Simple NVMe Device",
    "NVMeControllerProperties": {
        "ControllerType": "IO",
        "MaxQueueSize": 1,
        "NVMeControllerAttributes": {
            "ReportsNamespaceGranularity": false,
            "ReportsUUIDList": false,
            "Supports128BitHostId": false,
            "SupportsEnduranceGroups": false,
            "SupportsExceedingPowerOfNonOperationalState": false,
            "SupportsNVMSets": false,
            "SupportsPredictableLatencyMode": false,
            "SupportsReadRecoveryLevels": false,
            "SupportsSQAssociations": false,
            "SupportsTrafficBasedKeepAlive": false
        },
        "NVMeSMARTCriticalWarnings": {
            "MediaInReadOnly": false,
            "OverallSubsystemDegraded": false,
            "SpareCapacityWornOut": false
        },
        "NVMeVersion": "1.4"
    },
    "Name": "NVMe I/O Controller",
    "PartNumber": "NVM44",
    "SerialNumber": "NVME123456",
    "Status": {
        "Health": "OK",
```

Identify Networking Components:

Now, navigate back to the ServiceRoot, and look for the networking components. Start in the Chassis.

In the EBOFEnclosure Chassis instance, you will see NetworkAdapters. Check these out. Each network adapter includes NetworkDeviceFunctions and Ports.

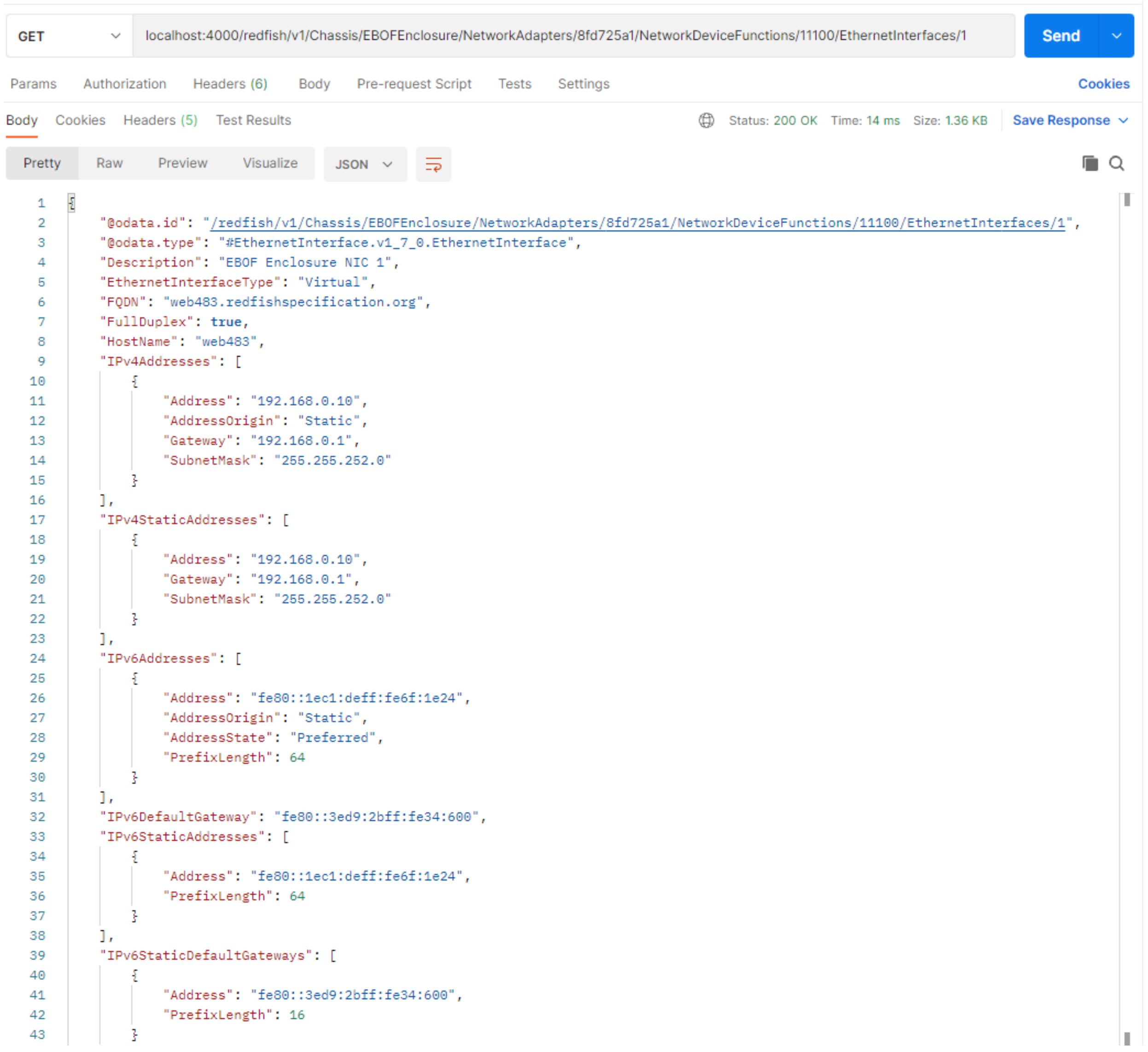


GET localhost:4000/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1

Status: 200 OK Time: 19 ms Size: 1.33 KB

```
    "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1",
    "@odata.type": "#NetworkAdapter.v1_9_0.NetworkAdapter",
    "Controllers": [
        {
            "ControllerCapabilities": {
                "DataCenterBridging": {
                    "Capable": true
                },
                "NetworkDeviceFunctionCount": 8,
                "NetworkPortCount": 2
            },
            "FirmwarePackageVersion": "7.4.10",
            "Links": {
                "NetworkDeviceFunctions": [
                    {
                        "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11100"
                    },
                    {
                        "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11101"
                    }
                ],
                "Ports": [
                    {
                        "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/Ports/1"
                    },
                    {
                        "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/Ports/2"
                    }
                ]
            }
        }
    ],
    "Description": "Embedded Switch network interface in EBOF enclosure.",
    "Id": "9fa725a1",
    "Manufacturer": "Globex",
    "Model": "599TPS-T",
    "Name": "Network Adapter View",
    "NetworkDeviceFunctions": {
        "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/NetworkDeviceFunctions"
```

Also, under each NetworkDeviceFunction, you will find additional definitions for EthernetInterfaces:



localhost:4000/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11100/EthernetInterfaces/1

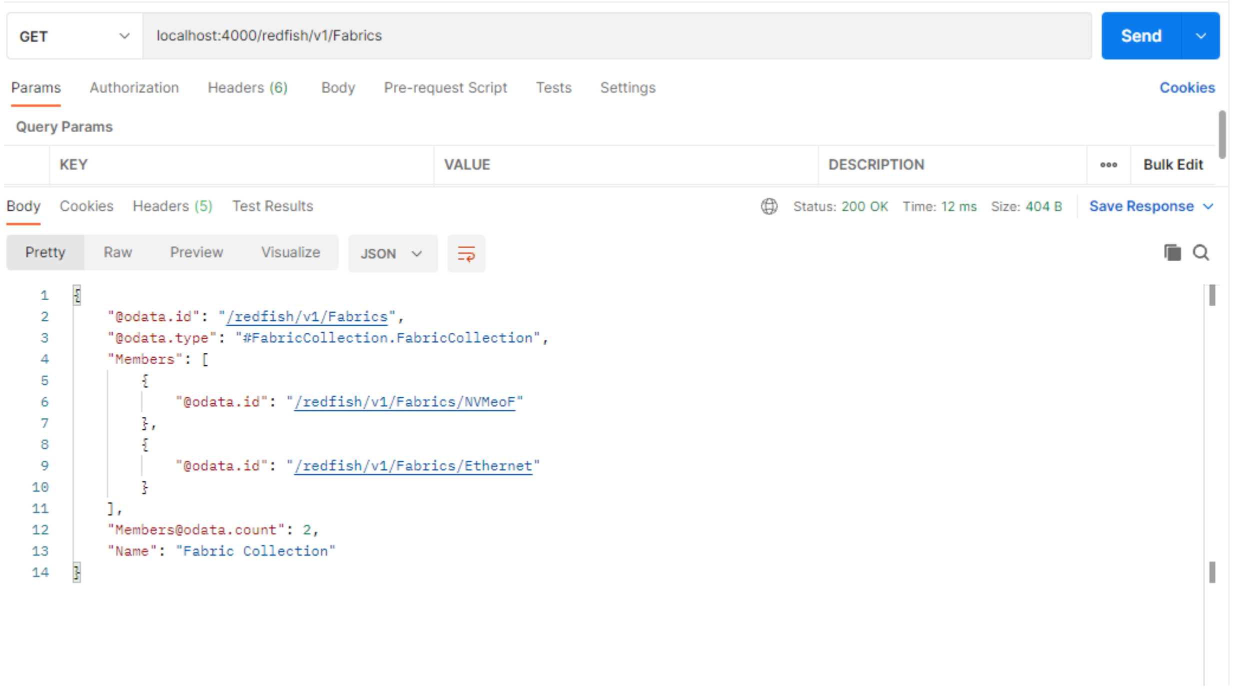
```
{
    "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11100/EthernetInterfaces/1",
    "@odata.type": "#EthernetInterface.v1_7_0.EthernetInterface",
    "Description": "EBOF Enclosure NIC 1",
    "EthernetInterfaceType": "Virtual",
    "FQDN": "web483.redfishspecification.org",
    "FullDuplex": true,
    "HostName": "web483",
    "IPv4Addresses": [
        {
            "Address": "192.168.0.10",
            "AddressOrigin": "Static",
            "Gateway": "192.168.0.1",
            "SubnetMask": "255.255.252.0"
        }
    ],
    "IPv4StaticAddresses": [
        {
            "Address": "192.168.0.10",
            "Gateway": "192.168.0.1",
            "SubnetMask": "255.255.252.0"
        }
    ],
    "IPv6Addresses": [
        {
            "Address": "fe80::1ec1:deff:fe6f:1e24",
            "AddressOrigin": "Static",
            "AddressState": "Preferred",
            "PrefixLength": 64
        }
    ],
    "IPv6DefaultGateway": "fe80::3ed9:2bff:fe34:600",
    "IPv6StaticAddresses": [
        {
            "Address": "fe80::1ec1:deff:fe6f:1e24",
            "PrefixLength": 64
        }
    ],
    "IPv6StaticDefaultGateways": [
        {
            "Address": "fe80::3ed9:2bff:fe34:600",
            "PrefixLength": 16
        }
```

That’s just the basic definition of the hardware interfaces on the chassis. Think of that as the device’s view of its network.

However, when configuring any device to connect to a network, there are actually three views: the system’s view, the target’s view, and the network’s view.

In this configuration, we have the target’s view, as well as a part of the network’s view: the EBOF itself has an embedded switch.

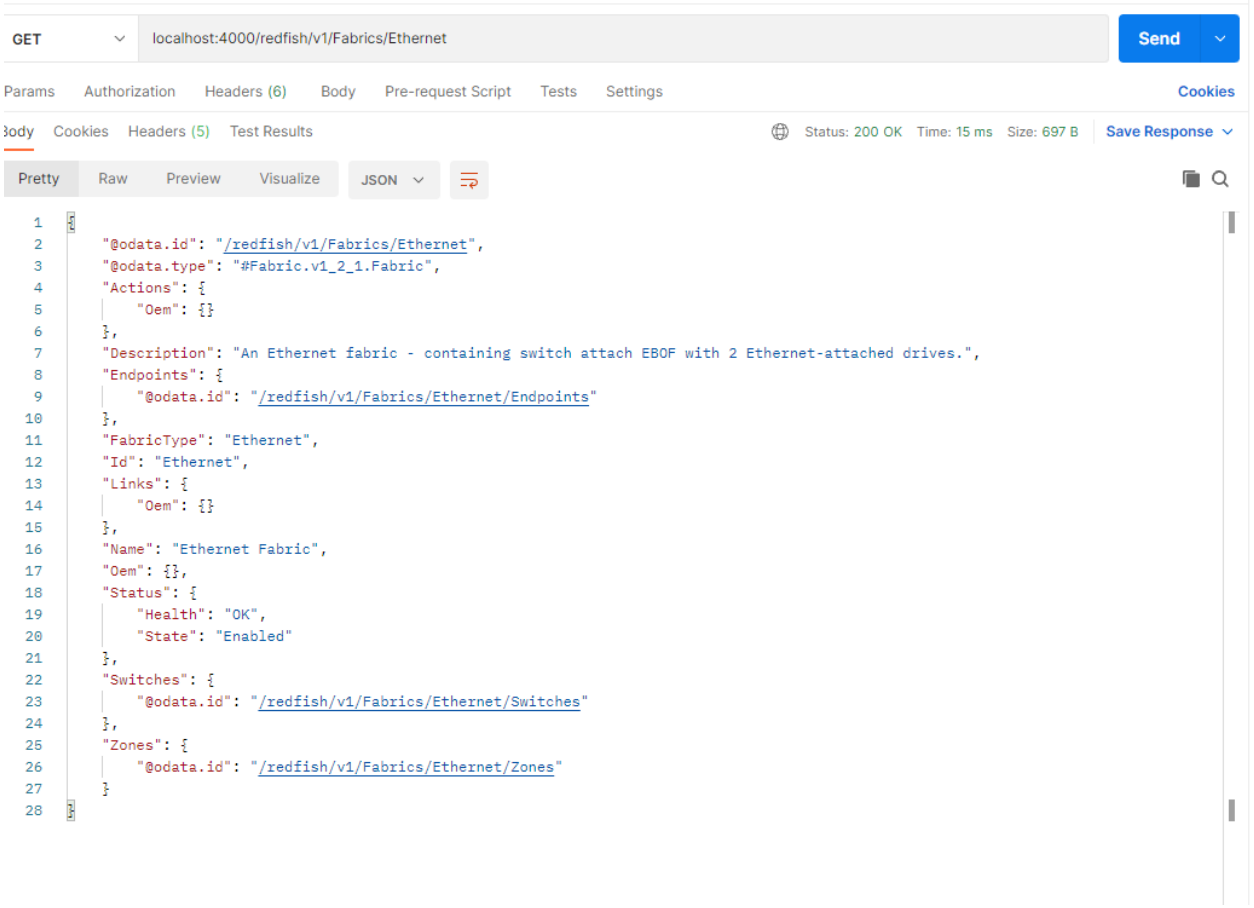
This view is modeled as part of the Fabric hierarchy. Go to the service root, and look at Fabrics:


GET localhost:4000/redfish/v1/Fabrics

Status: 200 OK Time: 12 ms Size: 404 B

```
{
    "@odata.id": "/redfish/v1/Fabrics",
    "@odata.type": "#FabricCollection.FabricCollection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Fabrics/NVMeoF"
        },
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet"
        }
    ],
    "Members@odata.count": 2,
    "Name": "Fabric Collection"
}
```

There are two Fabrics instances listed here. This is a best practice in Swordfish; we have separated out two different uses of the fabric model for clarity. The "Ethernet" instance is used to manage connectivity, and that is where we will find the switch object.
The NVMeoF instance is used to manage access rights.

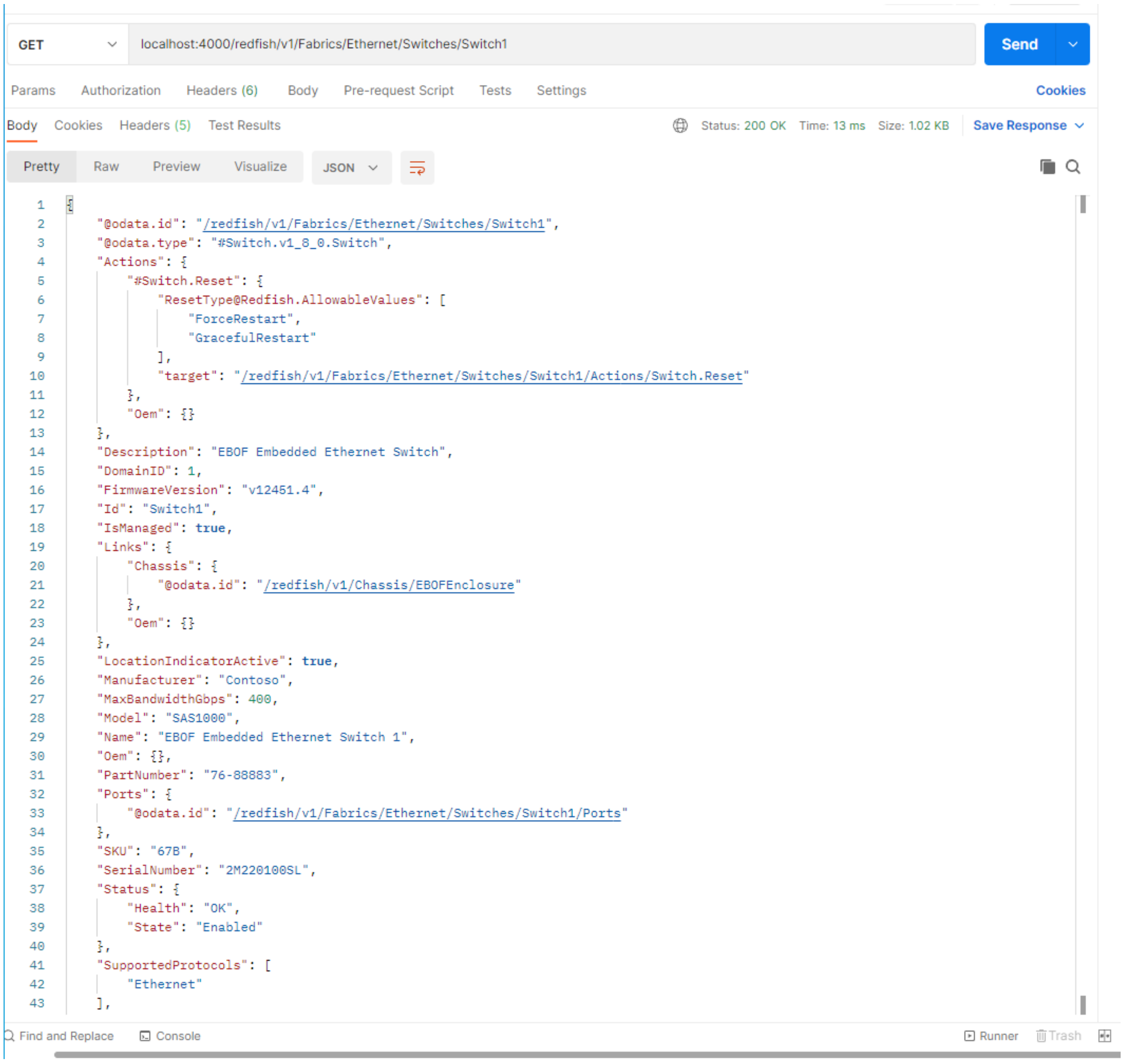


localhost:4000/redfish/v1/Fabrics/Ethernet

Status: 200 OK Time: 15 ms Size: 697 B

```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet",
    "@odata.type": "#Fabric.v1_2_1.Fabric",
    "Actions": {
        "Oem": {}
    },
    "Description": "An Ethernet fabric - containing switch attach EBOF with 2 Ethernet-attached drives.",
    "Endpoints": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints"
    },
    "FabricType": "Ethernet",
    "Id": "Ethernet",
    "Links": {
        "Oem": {}
    },
    "Name": "Ethernet Fabric",
    "Oem": {},
    "Status": {
        "Health": "OK",
        "State": "Enabled"
    },
    "Switches": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches"
    },
    "Zones": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Zones"
    }
}
```

The switch modeled here is the remaining piece of the physical representation of the network components.

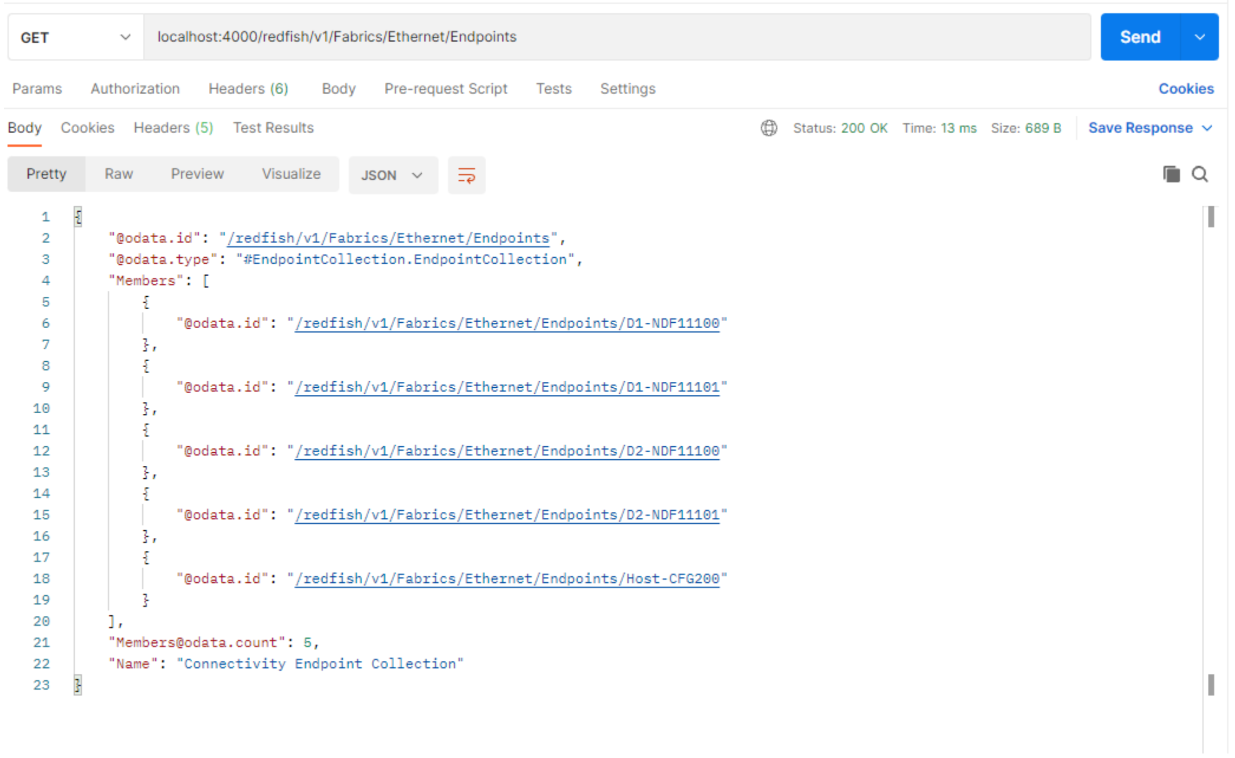


localhost:4000/redfish/v1/Fabrics/Ethernet/Switches/Switch1

```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches/Switch1",
    "@odata.type": "#Switch.v1_8_0.Switch",
    "Actions": {
        "#Switch.Reset": {
            "ResetType@Redfish.AllowableValues": [
                "ForceRestart",
                "GracefulRestart"
            ],
            "target": "/redfish/v1/Fabrics/Ethernet/Switches/Switch1/Actions/Switch.Reset"
        },
        "Oem": {}
    },
    "Description": "EBOF Embedded Ethernet Switch",
    "DomainID": 1,
    "FirmwareVersion": "v12451.4",
    "Id": "Switch1",
    "IsManaged": true,
    "Links": {
        "Chassis": {
            "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure"
        },
        "Oem": {}
    },
    "LocationIndicatorActive": true,
    "Manufacturer": "Contoso",
    "MaxBandwidthGbps": 400,
    "Model": "SAS1000",
    "Name": "EBOF Embedded Ethernet Switch 1",
    "Oem": {},
    "PartNumber": "76-88883",
    "Ports": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches/Switch1/Ports"
    },
    "SKU": "67B",
    "SerialNumber": "2M220100SL",
    "Status": {
        "Health": "OK",
        "State": "Enabled"
    },
    "SupportedProtocols": [
        "Ethernet"
    ],
```

Connectivity is modeled in the rest of the fabric, using Endpoints and Zones.

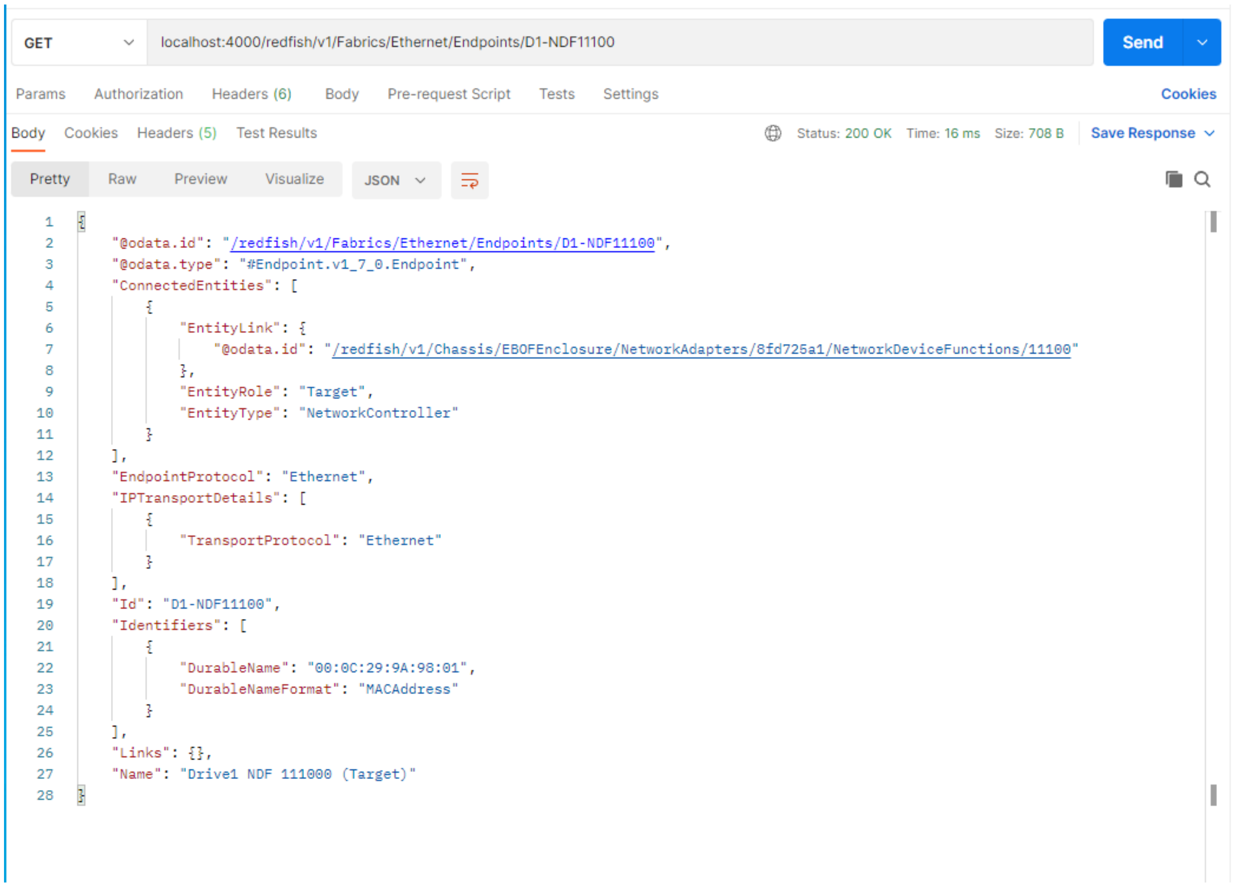
Endpoints are logical abstractions of connectivity:


GET localhost:4000/redfish/v1/Fabrics/Ethernet/Endpoints

Status: 200 OK Time: 13 ms Size: 689 B

```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints",
    "@odata.type": "#EndpointCollection.EndpointCollection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/D1-NDF11100"
        },
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/D1-NDF11101"
        },
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/D2-NDF11100"
        },
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/D2-NDF11101"
        },
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/Host-CFG200"
        }
    ],
    "Members@odata.count": 5,
    "Name": "Connectivity Endpoint Collection"
}
```

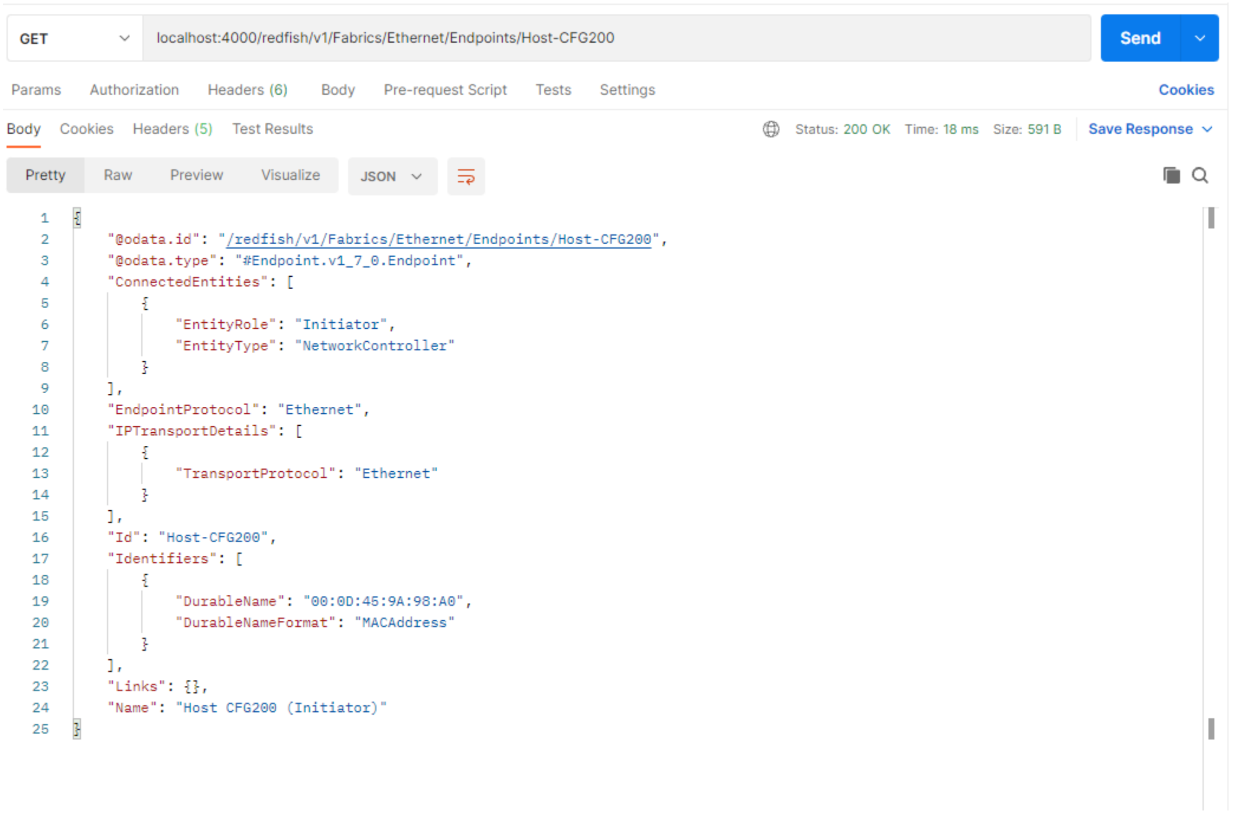
For each network device function, there is an endpoint.


GET localhost:4000/redfish/v1/Fabrics/Ethernet/Endpoints/D1-NDF11100

Status: 200 OK Time: 16 ms Size: 708 B

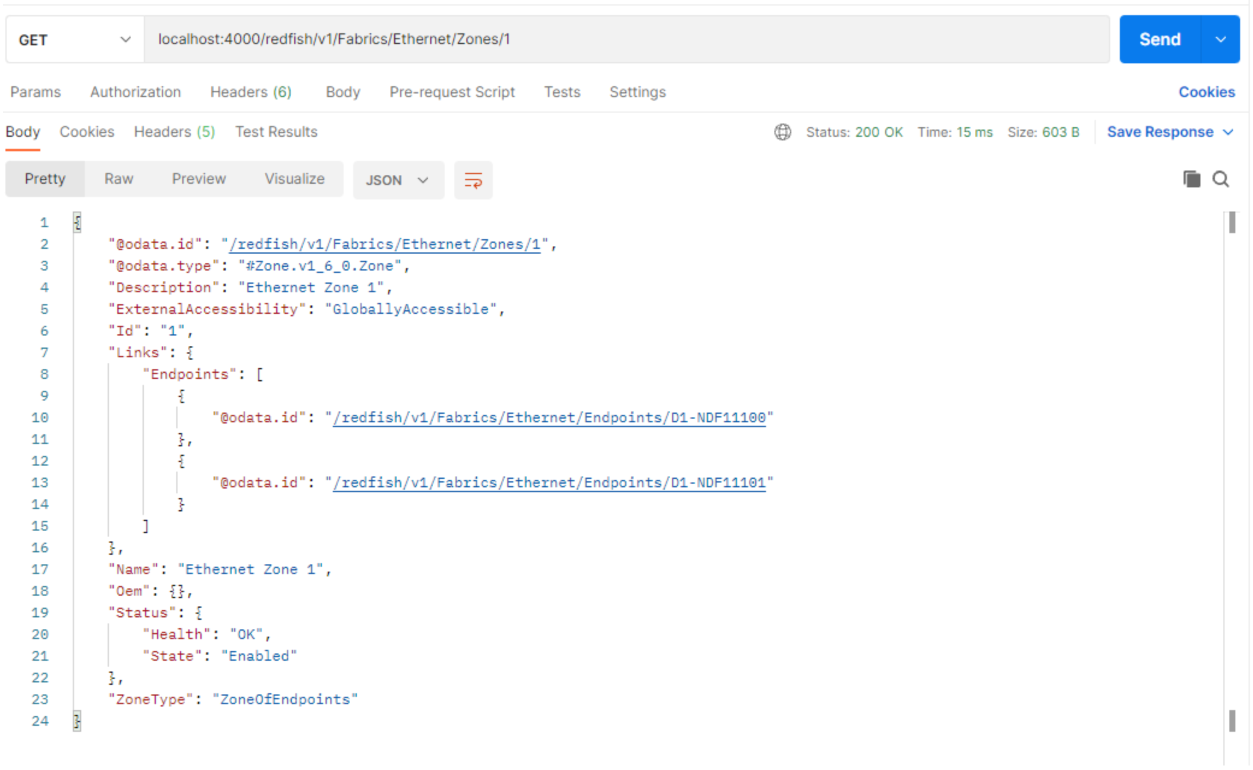
```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/D1-NDF11100",
    "@odata.type": "#Endpoint.v1_7_0.Endpoint",
    "ConnectedEntities": [
        {
            "EntityLink": {
                "@odata.id": "/redfish/v1/Chassis/EBOFEnclosure/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11100"
            },
            "EntityRole": "Target",
            "EntityType": "NetworkController"
        }
    ],
    "EndpointProtocol": "Ethernet",
    "IPTransportDetails": [
        {
            "TransportProtocol": "Ethernet"
        }
    ],
    "Id": "D1-NDF11100",
    "Identifiers": [
        {
            "DurableName": "00:0C:29:9A:98:01",
            "DurableNameFormat": "MACAddress"
        }
    ],
    "Links": {},
    "Name": "Drive1 NDF 111000 (Target)"
}
```

And on the host, there is an endpoint to represent the network controller:


GET
localhost:4000/redfish/v1/Fabrics/Ethernet/Endpoints/Host-CFG200
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies
Body Cookies Headers (5) Test Results Status: 200 OK Time: 18 ms Size: 591 B Save Response
Pretty Raw Preview Visualize JSON

```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/Host-CFG200",
    "@odata.type": "#Endpoint.v1_7_0.Endpoint",
    "ConnectedEntities": [
        {
            "EntityRole": "Initiator",
            "EntityType": "NetworkController"
        }
    ],
    "EndpointProtocol": "Ethernet",
    "IPTransportDetails": [
        {
            "TransportProtocol": "Ethernet"
        }
    ],
    "Id": "Host-CFG200",
    "Identifiers": [
        {
            "DurableName": "00:0D:45:9A:98:A0",
            "DurableNameFormat": "MACAddress"
        }
    ],
    "Links": {},
    "Name": "Host CFG200 (Initiator)"
}
```

Connectivity between them, then, is represented in Zones:


GET
localhost:4000/redfish/v1/Fabrics/Ethernet/Zones/1
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies
Body Cookies Headers (5) Test Results Status: 200 OK Time: 15 ms Size: 603 B Save Response
Pretty Raw Preview Visualize JSON

```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Zones/1",
    "@odata.type": "#Zone.v1_6_0.Zone",
    "Description": "Ethernet Zone 1",
    "ExternalAccessibility": "GloballyAccessible",
    "Id": "1",
    "Links": {
        "Endpoints": [
            {
                "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/D1-NDF11100"
            },
            {
                "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints/D1-NDF11101"
            }
        ]
    },
    "Name": "Ethernet Zone 1",
    "Oem": {},
    "Status": {
        "Health": "OK",
        "State": "Enabled"
    },
    "ZoneType": "ZoneOfEndpoints"
}
```

Exercise 2: Create a new Volume on a Disk Array

Purpose:

This exercise will familiarize the participant with the Swordfish model when applied to an external block storage array. It will also introduce the participant to techniques required for active configuration via the Swordfish interface.

Overview:

The following activities are covered in this exercise:

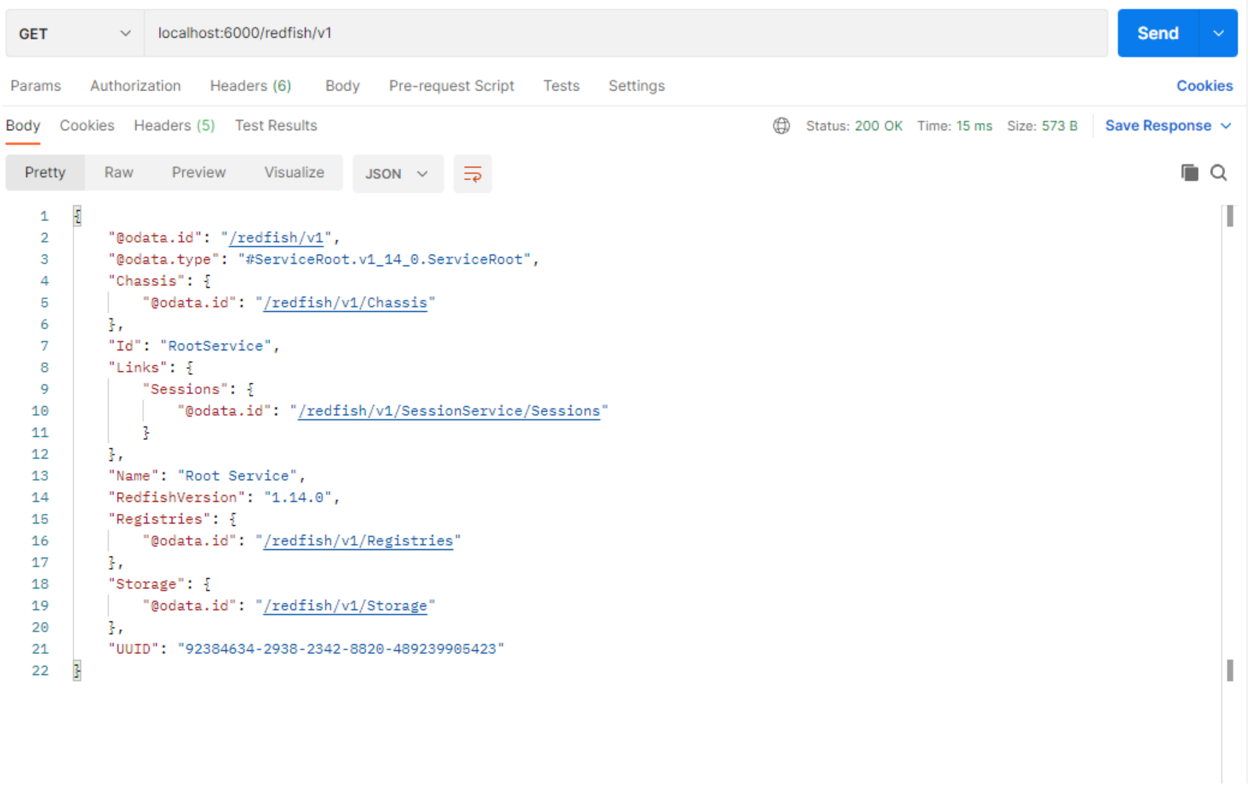
The participant will navigate around to understand the system. After navigating around the system, the participant will find the StoragePools collection, identify a Storage Pool, and POST a new Volume to its AllocatedVolumes collection.

(Disclaimer: the participant is working with the Swordfish emulator; no checking of appropriate parameters for a storage pool is done at POST.)

Getting Started:

Using Postman, go to the ServiceRoot of the External Array Swordfish Service:

Select the REST command type in the drop-down box (GET), and type the URI in the gray box. (localhost:6000/redfish/v1). Click "Send"


GET localhost:6000/redfish/v1

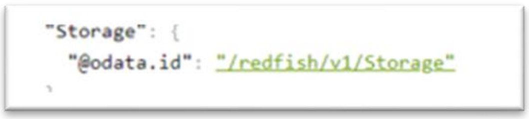
Status: 200 OK Time: 15 ms Size: 573 B

```
{
    "@odata.id": "/redfish/v1",
    "@odata.type": "#ServiceRoot.v1_14_0.ServiceRoot",
    "Chassis": {
        "@odata.id": "/redfish/v1/Chassis"
    },
    "Id": "RootService",
    "Links": {
        "Sessions": {
            "@odata.id": "/redfish/v1/SessionService/Sessions"
        }
    },
    "Name": "Root Service",
    "RedfishVersion": "1.14.0",
    "Registries": {
        "@odata.id": "/redfish/v1/Registries"
    },
    "Storage": {
        "@odata.id": "/redfish/v1/Storage"
    },
    "UUID": "92384634-2938-2342-8820-489239905423"
}
```

Note that this system does not include comprehensive Fabric / connectivity information. For this exercise, we will focus on the logical / storage configuration.

Navigate to the Storage Model:

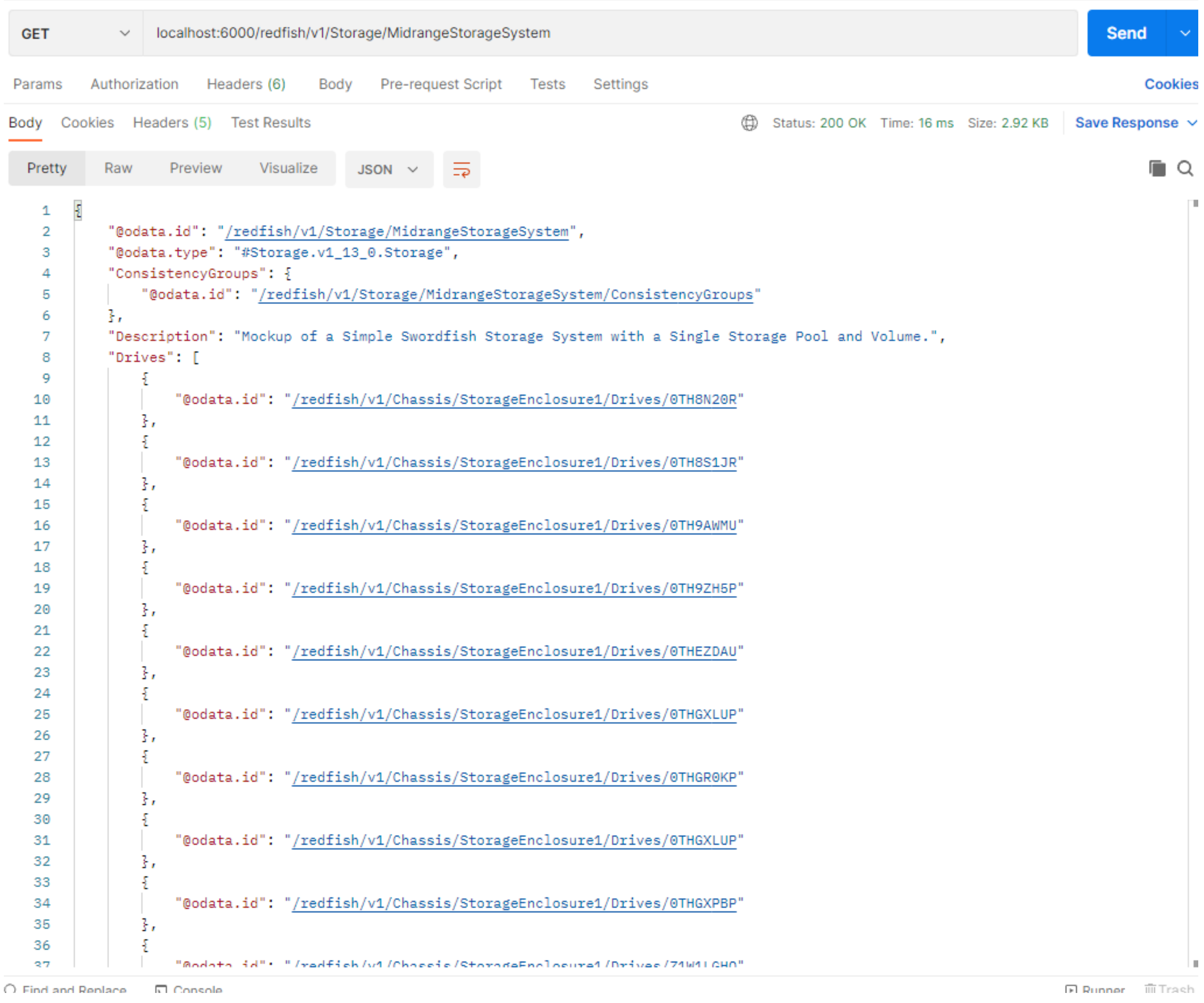
Find the link to the Storage and navigate to it.


```
"Storage": {
  "@odata.id": "/redfish/v1/Storage"
```

If you get stuck, you can always to the nav bar and reset to the service root (/redfish/v1).

Identify the Components of the Storage Model

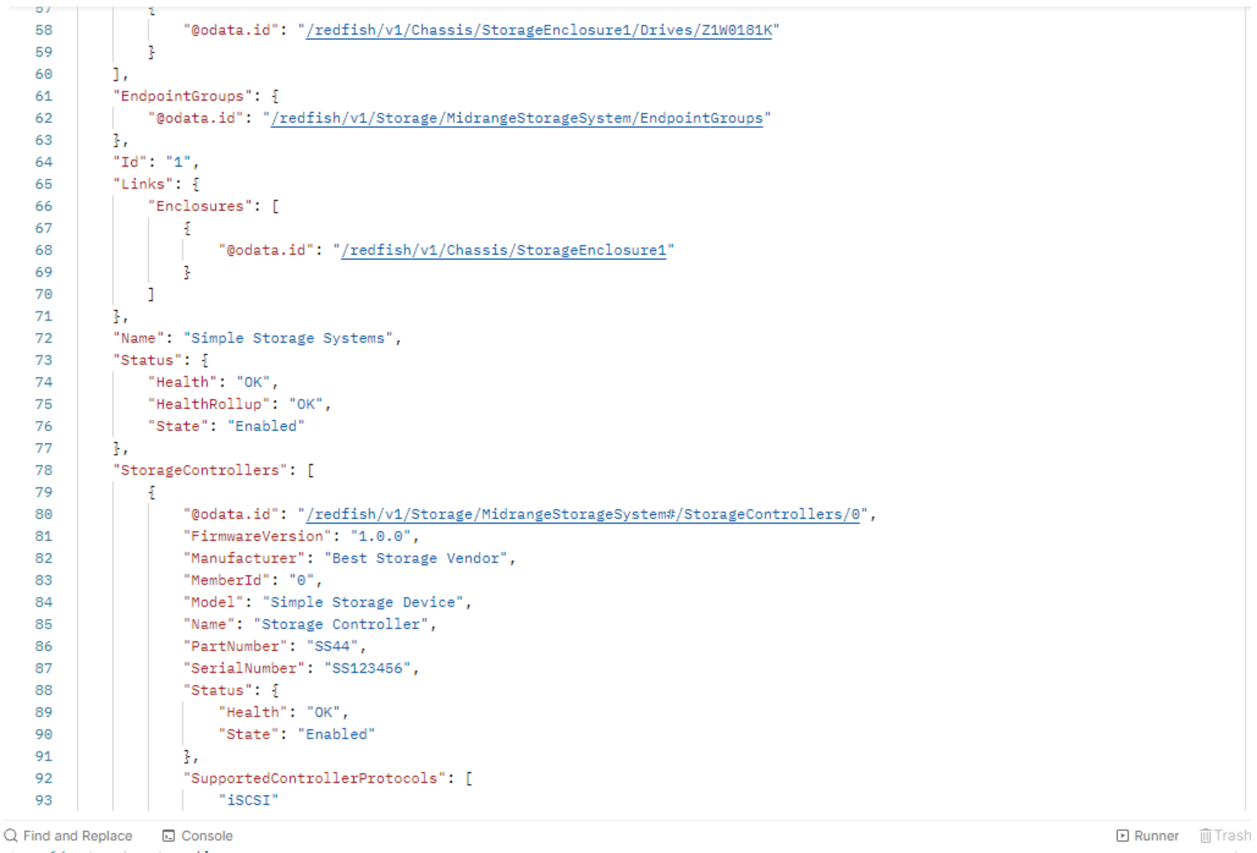
From the Storage collection, navigation into the Storage device. You'll notice this device looks quite different from the EBOF configuration.



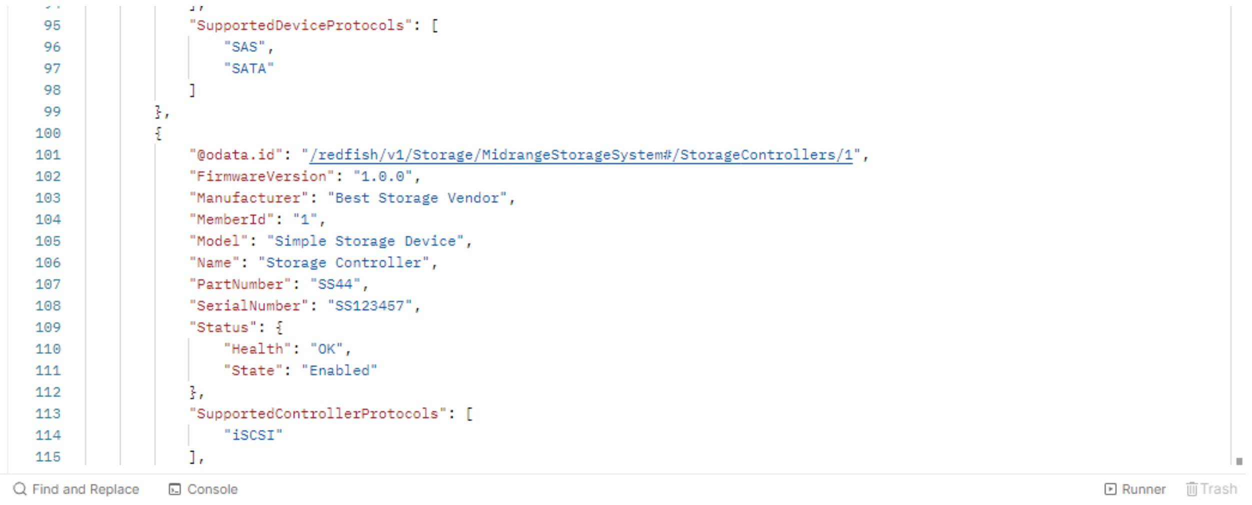
localhost:6000/redfish/v1/Storage/MidrangeStorageSystem

Status: 200 OK Time: 16 ms Size: 2.92 KB

```
{
    "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem",
    "@odata.type": "#Storage.v1_13_0.Storage",
    "ConsistencyGroups": {
        "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/ConsistencyGroups"
    },
    "Description": "Mockup of a Simple Swordfish Storage System with a Single Storage Pool and Volume.",
    "Drives": [
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0TH8N20R"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0TH8S1JR"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0TH9AWMU"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0TH9ZH5P"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0THEZDAU"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0THGXLUP"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0THGR0KP"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0THGXLUP"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0THGXPBP"
        },
        {
            "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/Z1W1LGH0"
```



```
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/Z1W0181K"
            }
        ],
        "EndpointGroups": {
            "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/EndpointGroups"
        },
        "Id": "1",
        "Links": {
            "Enclosures": [
                {
                    "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1"
                }
            ]
        },
        "Name": "Simple Storage Systems",
        "Status": {
            "Health": "OK",
            "HealthRollup": "OK",
            "State": "Enabled"
        },
        "StorageControllers": [
            {
                "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem#/StorageControllers/0",
                "FirmwareVersion": "1.0.0",
                "Manufacturer": "Best Storage Vendor",
                "MemberId": "0",
                "Model": "Simple Storage Device",
                "Name": "Storage Controller",
                "PartNumber": "SS44",
                "SerialNumber": "SS123456",
                "Status": {
                    "Health": "OK",
                    "State": "Enabled"
                },
                "SupportedControllerProtocols": [
                    "iSCSI"
```

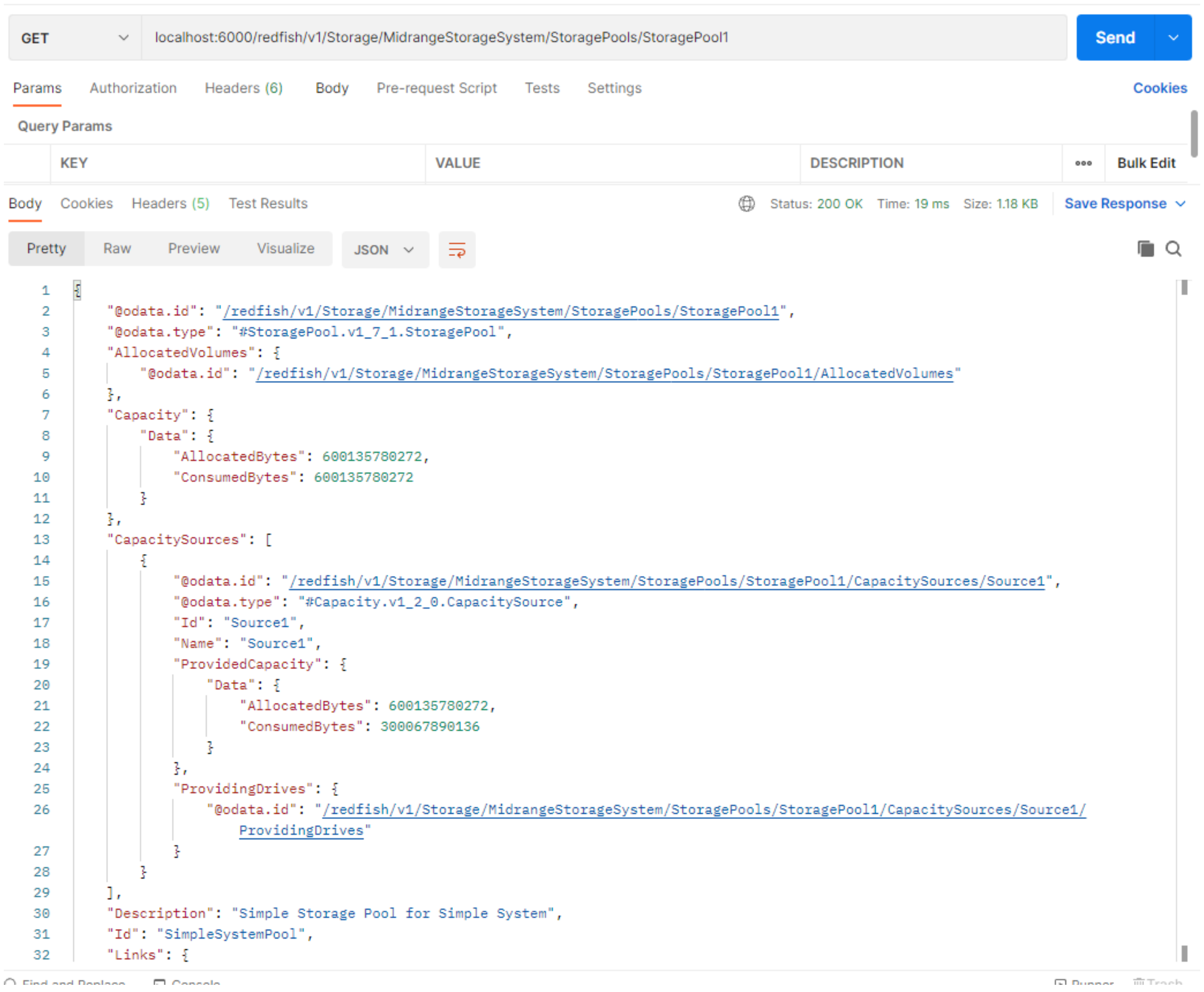


```
                ],
                "SupportedDeviceProtocols": [
                    "SAS",
                    "SATA"
                ]
            },
            {
                "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem#/StorageControllers/1",
                "FirmwareVersion": "1.0.0",
                "Manufacturer": "Best Storage Vendor",
                "MemberId": "1",
                "Model": "Simple Storage Device",
                "Name": "Storage Controller",
                "PartNumber": "SS44",
                "SerialNumber": "SS123457",
                "Status": {
                    "Health": "OK",
                    "State": "Enabled"
                },
                "SupportedControllerProtocols": [
                    "iSCSI"
                ],
```

There are quite a few of the same properties, but some notable differences. The system represents controllers differently (representing physical instead of logical controllers).

Navigating around the system, you will see the traditional relationship between StoragePools and Volumes.

There is one additional layer in the hierarchy, called CapacitySources, that supports the use of multiple types of capacity to create a single pool. Pools can also be hierarchical; you can have pools of pools; or pools of volumes; or, pools of drives.


GET localhost:6000/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1

Status: 200 OK Time: 19 ms Size: 1.18 KB

```
{
    "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1",
    "@odata.type": "#StoragePool.v1_7_1.StoragePool",
    "AllocatedVolumes": {
        "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1/AllocatedVolumes"
    },
    "Capacity": {
        "Data": {
            "AllocatedBytes": 600135780272,
            "ConsumedBytes": 600135780272
        }
    },
    "CapacitySources": [
        {
            "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1/CapacitySources/Source1",
            "@odata.type": "#Capacity.v1_2_0.CapacitySource",
            "Id": "Source1",
            "Name": "Source1",
            "ProvidedCapacity": {
                "Data": {
                    "AllocatedBytes": 600135780272,
                    "ConsumedBytes": 300067890136
                }
            },
            "ProvidingDrives": {
                "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1/CapacitySources/Source1/
                    ProvidingDrives"
            }
        }
    ],
    "Description": "Simple Storage Pool for Simple System",
    "Id": "SimpleSystemPool",
    "Links": {
```

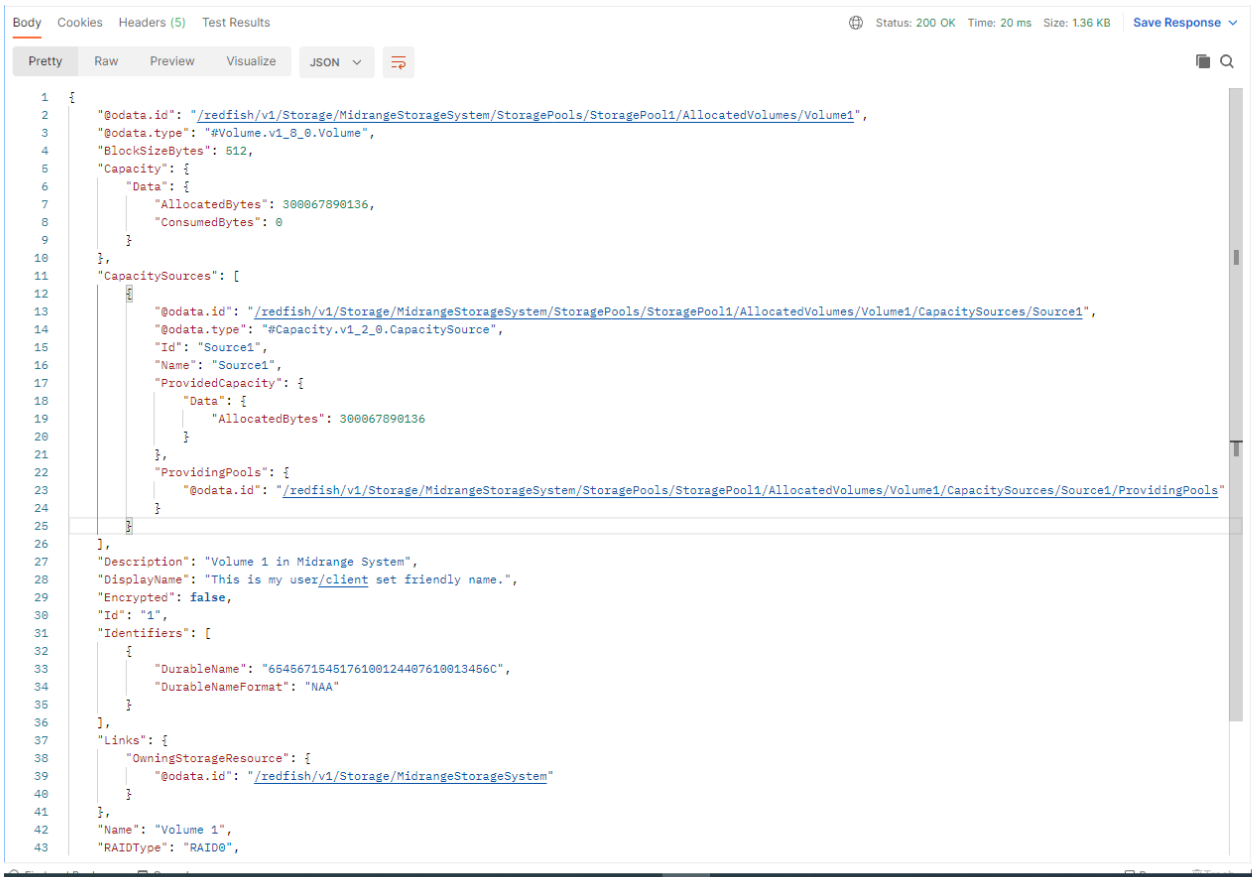
Volumes are created from StoragePools. The “AllocatedVolumes” within a StoragePool is the origin, or actual location, where the volume lives. The volume can be referenced from many other places within the Swordfish hierarchy for convenience.

This includes a collection at the root of the Storage instance (./Storage/{StorageId}/Volumes} where you can find links to all Volumes within a storage instance.

Create a New Volume:

In order to identify the information we need to provide to create a new Volume, let’s look at an existing one:

Here is the body for “Volume1” on StoragePool1. We’ll can use this text to reference when we create our POST command.


Body Cookies Headers (5) Test Results
Status: 200 OK Time: 20 ms Size: 1.36 KB
Save Response
Pretty Raw Preview Visualize JSON

```
{
    "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1/AllocatedVolumes/Volume1",
    "@odata.type": "#Volume.v1_8_0.Volume",
    "BlockSizeBytes": 512,
    "Capacity": {
        "Data": {
            "AllocatedBytes": 300067890136,
            "ConsumedBytes": 0
        }
    },
    "CapacitySources": [
        {
            "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1/AllocatedVolumes/Volume1/CapacitySources/Source1",
            "@odata.type": "#Capacity.v1_2_0.CapacitySource",
            "Id": "Source1",
            "Name": "Source1",
            "ProvidedCapacity": {
                "Data": {
                    "AllocatedBytes": 300067890136
                }
            },
            "ProvidingPools": {
                "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1/AllocatedVolumes/Volume1/CapacitySources/Source1/ProvidingPools"
            }
        }
    ],
    "Description": "Volume 1 in Midrange System",
    "DisplayName": "This is my user/client set friendly name.",
    "Encrypted": false,
    "Id": "1",
    "Identifiers": [
        {
            "DurableName": "6545671545176100124407610013456C",
            "DurableNameFormat": "NAA"
        }
    ],
    "Links": {
        "OwningStorageResource": {
            "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem"
        }
    },
    "Name": "Volume 1",
    "RAIDType": "RAID0",
```

Create POST command:

Let's determine what properties we need in order to create our new volume.

Volume ID: What's the ID for the new volume? We will use this for both the odata.id and ID properties.

(NOTE – on a real system you may or may not be able to set this yourself. For the emulator this doesn't yet create it automatically, so we include it in the POST request body.)

Capacity – set the AllocatedBytes in Capacity to a value at or below the capacity remaining in your StoragePool. (note that the emulator won't check this for you.)

Set "RAIDType" something supported by the StoragePool.

Do you want encryption enabled? Set "Encrypted" if supported by the StoragePool

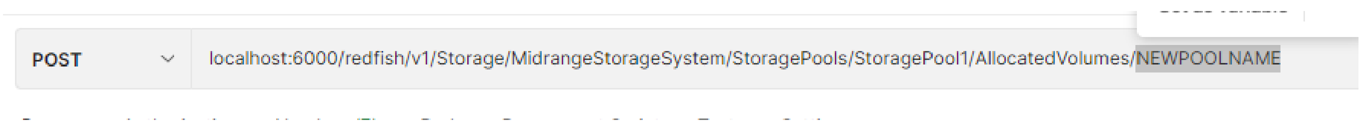
There are more properties that can be set – dependent on the device you are working with. But we'll stop here for today.

```
{
  "@odata.id":
"/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/Stora
gePool1/AllocatedVolumes/NEWVOLUME",
  "ID": "NEWVOLUME",

  "BlockSizeBytes": 512,
  "Capacity": {
    "Data": {
      "AllocatedBytes": 300067890136,
      "ConsumedBytes": 0
    }
  },
  "RAIDType": "RAID0"
}
```

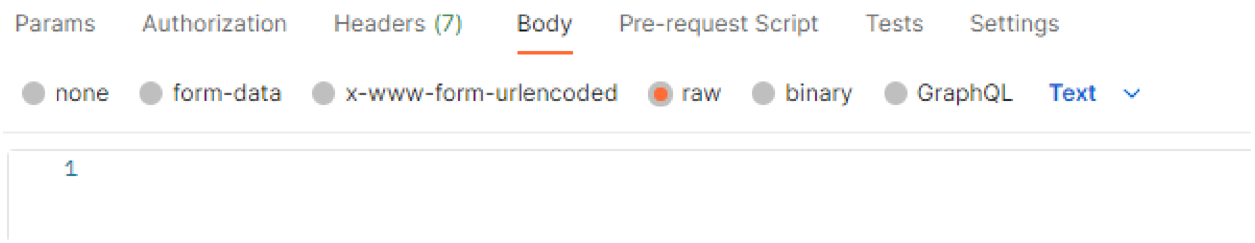
POST Commands are submitted on the Collection.

What that means: Select POST as the REST command type, and then type the new pool name (StoragePool3?) after the collection name on the nav bar. DON'T CLICK SEND YET.



POST localhost:6000/redfish/v1/Storage/MidrangeStorageSystem/StoragePools/StoragePool1/AllocatedVolumes/NEWPOOLNAME

Under the nav bar area, select "body", and "raw"



In this text area, paste in the new Volume object (see above) that you plan to use as a template.

Now. Click SEND.

LARGE DISCLAIMER: Real systems will do all sorts of data checking, object creation, etc for you.

Exercise 3: Remove a Redundant Connection to a Namespace

Purpose:

This exercise will familiarize the participant with the Swordfish model when applied to an external block storage array which has an NVMe/TCP interface, combining elements from the two previous exercises. After familiarizing themselves with the configuration, the participant will perform an NVMe-specific action: removing a redundant host access configuration to a namespace (aka, a volume).

Overview:

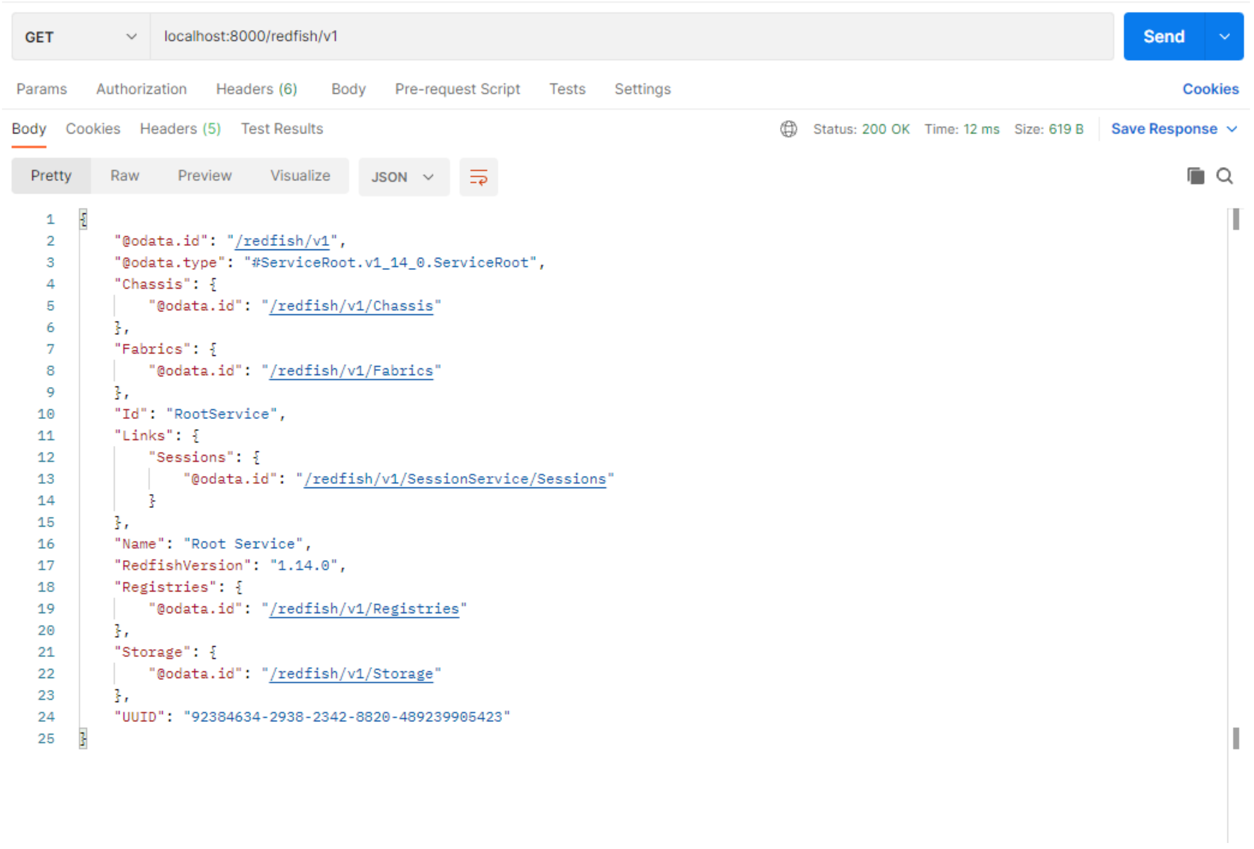
The following activities are covered in this exercise:

The participant will navigate around to understand the system. After navigating around the system, the participant will find the StorageControllers collection and delete a redundant IO Controller.

Getting Started:

Using Postman, go to the ServiceRoot of the NVMe/TCP array service:

Select the REST command type in the drop-down box (GET), and type the URI in the gray box. (localhost:8000/redfish/v1). Click “Send”

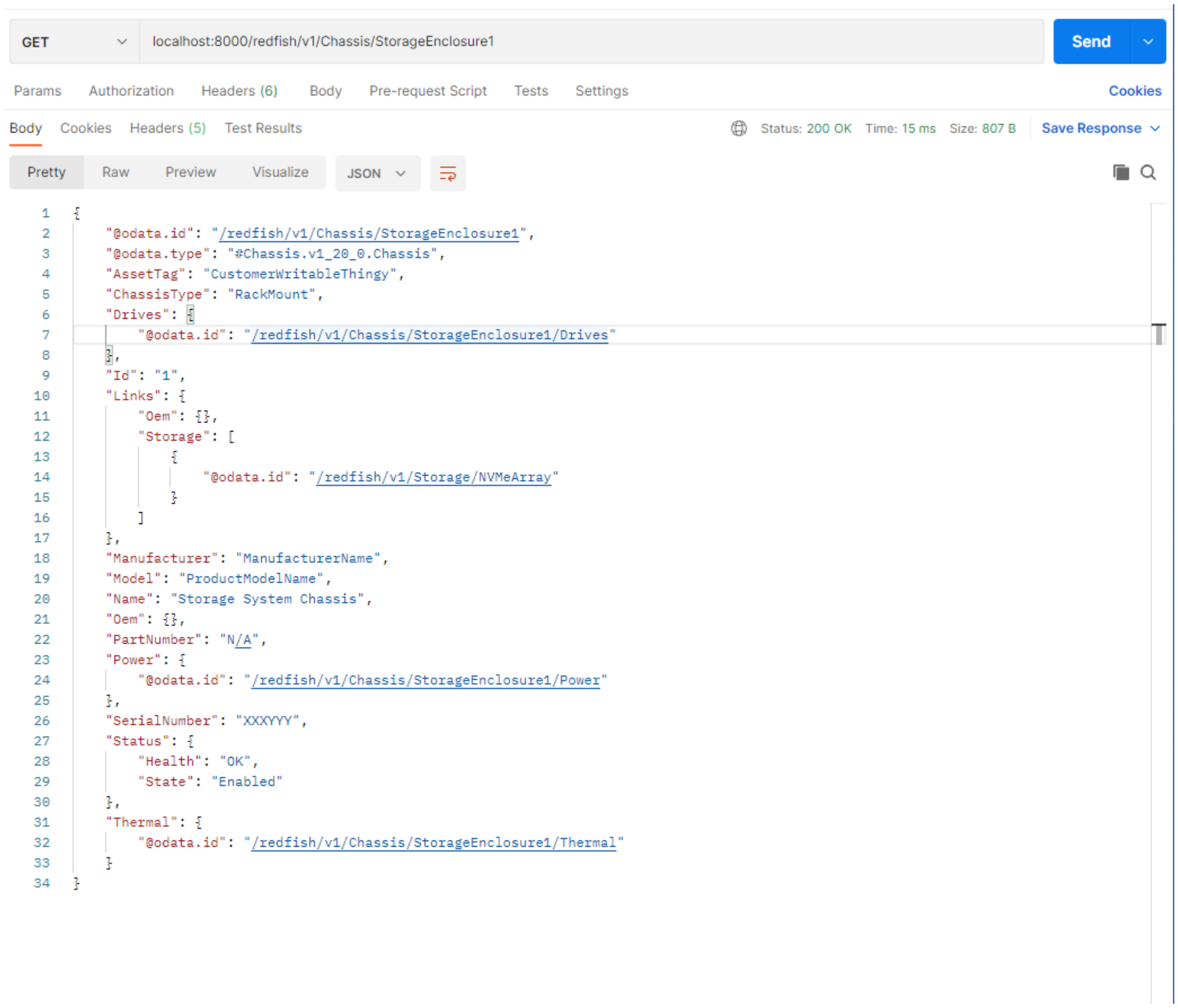


```
{
    "@odata.id": "/redfish/v1",
    "@odata.type": "#ServiceRoot.v1_14_0.ServiceRoot",
    "Chassis": {
        "@odata.id": "/redfish/v1/Chassis"
    },
    "Fabrics": {
        "@odata.id": "/redfish/v1/Fabrics"
    },
    "Id": "RootService",
    "Links": {
        "Sessions": {
            "@odata.id": "/redfish/v1/SessionService/Sessions"
        }
    },
    "Name": "Root Service",
    "RedfishVersion": "1.14.0",
    "Registries": {
        "@odata.id": "/redfish/v1/Registries"
    },
    "Storage": {
        "@odata.id": "/redfish/v1/Storage"
    },
    "UUID": "92384634-2938-2342-8820-489239905423"
}
```

Let’s navigate around this configuration and see what it looks like.

Learning the system configuration:

Let’s look at the physical configuration. Go to the Chassis and look around.

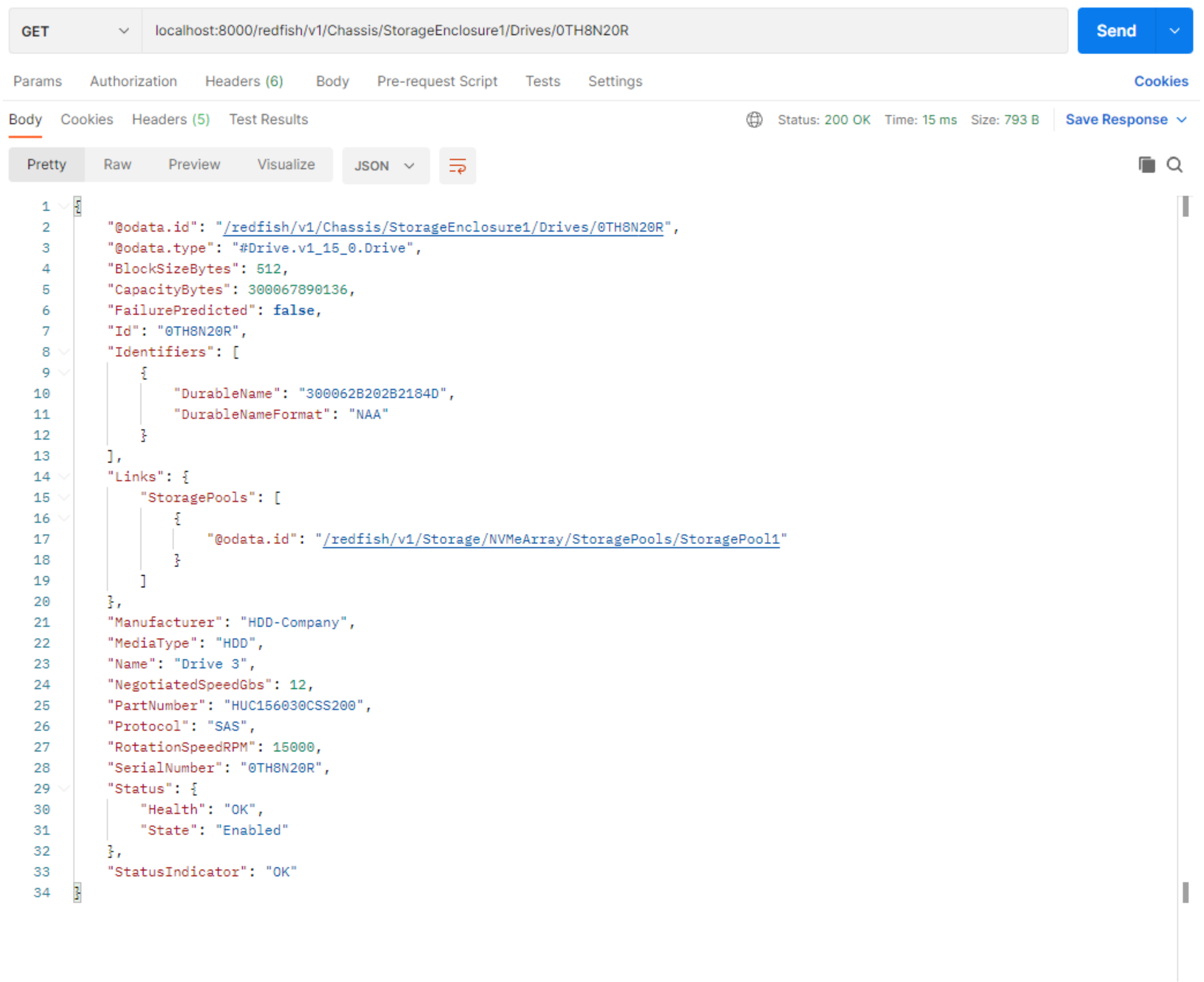


localhost:8000/redfish/v1/Chassis/StorageEnclosure1

Status: 200 OK Time: 15 ms Size: 807 B

```
{
    "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1",
    "@odata.type": "#Chassis.v1_20_0.Chassis",
    "AssetTag": "CustomerWritableThingy",
    "ChassisType": "RackMount",
    "Drives": {
        "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives"
    },
    "Id": "1",
    "Links": {
        "Oem": {},
        "Storage": [
            {
                "@odata.id": "/redfish/v1/Storage/NVMeArray"
            }
        ]
    },
    "Manufacturer": "ManufacturerName",
    "Model": "ProductModelName",
    "Name": "Storage System Chassis",
    "Oem": {},
    "PartNumber": "N/A",
    "Power": {
        "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Power"
    },
    "SerialNumber": "XXXYYY",
    "Status": {
        "Health": "OK",
        "State": "Enabled"
    },
    "Thermal": {
        "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Thermal"
    }
}
```

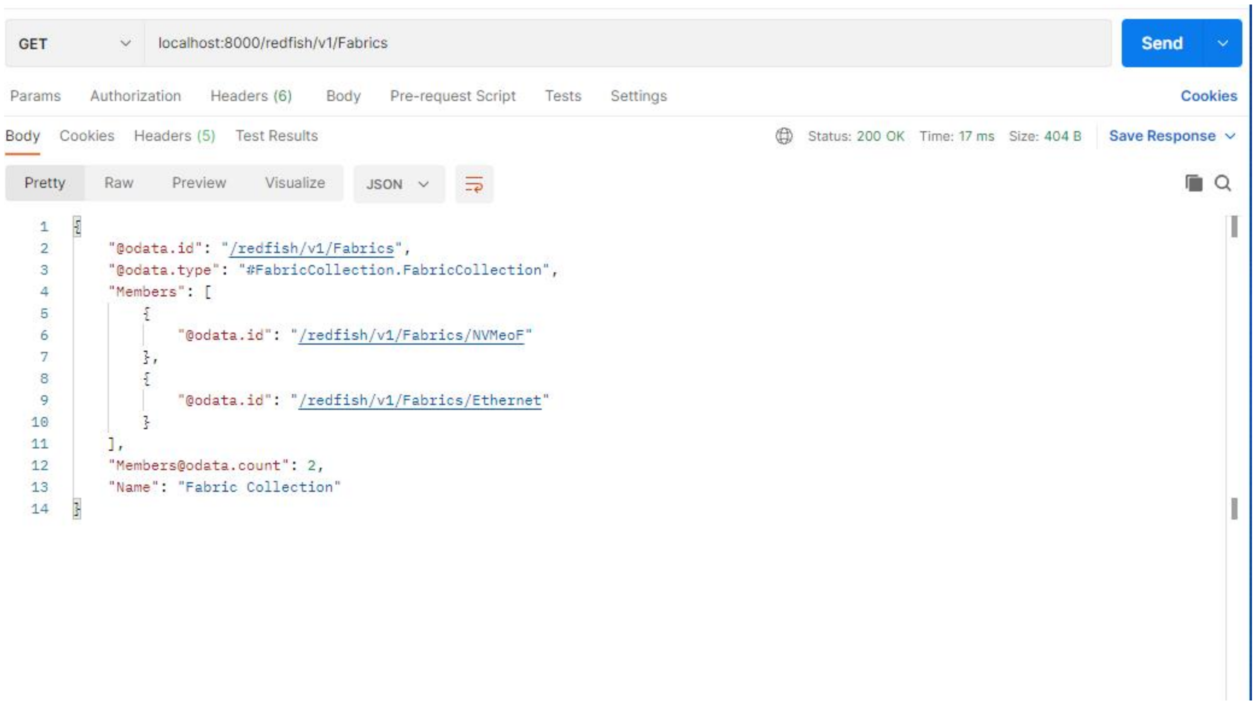
In the Chassis, we see a single enclosure, with a set of drives. As we look at these drives, we notice something interesting…


GET
localhost:8000/redfish/v1/Chassis/StorageEnclosure1/Drives/0TH8N20R
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies
Body Cookies Headers (5) Test Results
Status: 200 OK Time: 15 ms Size: 793 B
Save Response
Pretty Raw Preview Visualize JSON

```
{
    "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives/0TH8N20R",
    "@odata.type": "#Drive.v1_15_0.Drive",
    "BlockSizeBytes": 512,
    "CapacityBytes": 300067890136,
    "FailurePredicted": false,
    "Id": "0TH8N20R",
    "Identifiers": [
        {
            "DurableName": "300062B202B2184D",
            "DurableNameFormat": "NAA"
        }
    ],
    "Links": {
        "StoragePools": [
            {
                "@odata.id": "/redfish/v1/Storage/NVMeArray/StoragePools/StoragePool1"
            }
        ]
    },
    "Manufacturer": "HDD-Company",
    "MediaType": "HDD",
    "Name": "Drive 3",
    "NegotiatedSpeedGbs": 12,
    "PartNumber": "HUC156030CSS200",
    "Protocol": "SAS",
    "RotationSpeedRPM": 15000,
    "SerialNumber": "0TH8N20R",
    "Status": {
        "Health": "OK",
        "State": "Enabled"
    },
    "StatusIndicator": "OK"
}
```

These are SAS drives. As we look at more of this configuration, we'll see that this array has an NVMe front end, but a SAS back end. Swordfish supports this hybrid configuration very easily, blending the traditional model with the NVMe model seamlessly.

Continuing to check out the configuration, let's go back to the ServiceRoot, and look at the Fabrics configuration:



localhost:8000/redfish/v1/Fabrics

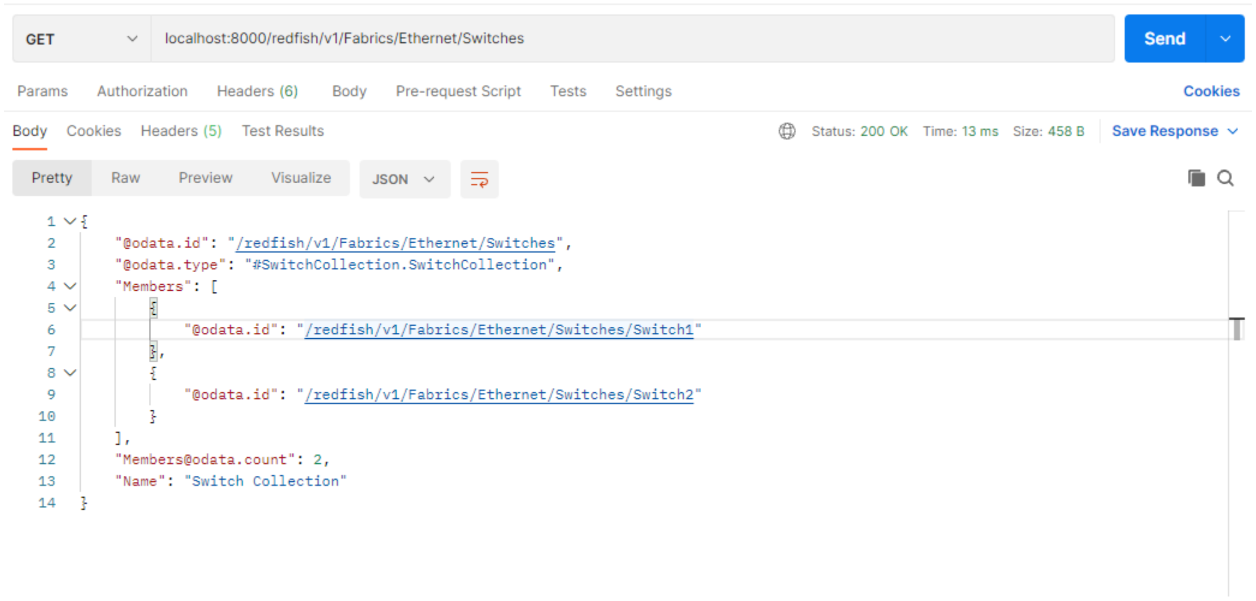
Status: 200 OK Time: 17 ms Size: 404 B

```
{
    "@odata.id": "/redfish/v1/Fabrics",
    "@odata.type": "#FabricCollection.FabricCollection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Fabrics/NVMeoF"
        },
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet"
        }
    ],
    "Members@odata.count": 2,
    "Name": "Fabric Collection"
}
```

This looks familiar – we saw this same type of configuration with the EBOF system. This must be, again, both a connectivity and access rights configuration. Let's look and confirm.

Looking at the Ethernet Fabric:

This system again has a similar configuration – the chassis has two switches built in for connectivity.



localhost:8000/redfish/v1/Fabrics/Ethernet/Switches

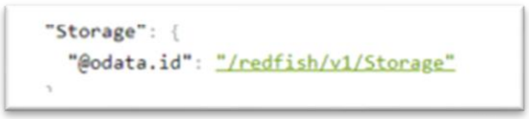
Status: 200 OK Time: 13 ms Size: 458 B

```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches",
    "@odata.type": "#SwitchCollection.SwitchCollection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches/Switch1"
        },
        {
            "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches/Switch2"
        }
    ],
    "Members@odata.count": 2,
    "Name": "Switch Collection"
}
```

Continuing to navigate around the Fabric model, you will see many similarities to the modeling for the EBOF configuration.

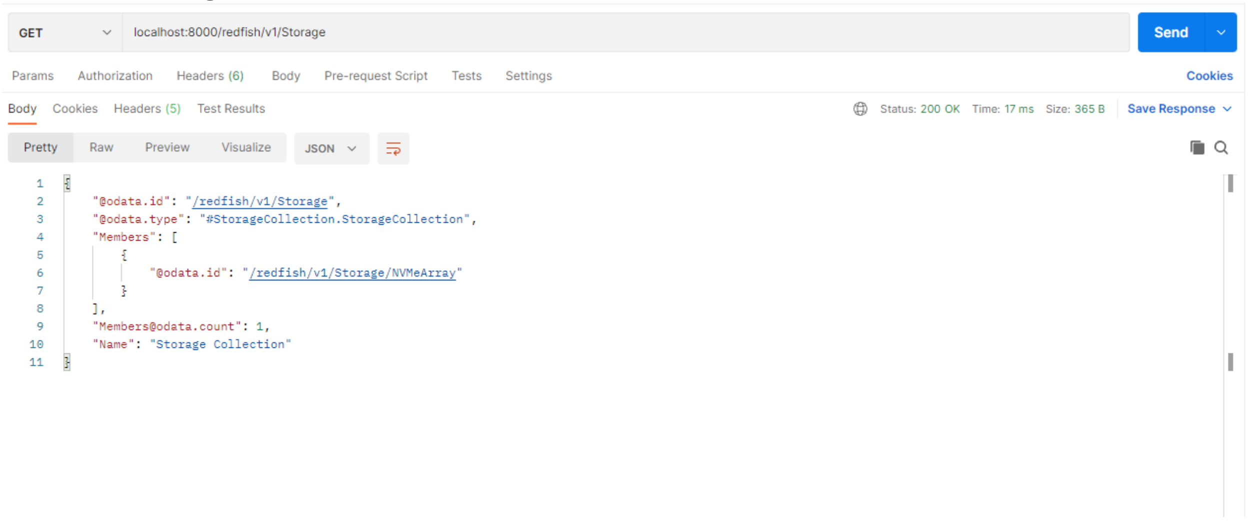
Navigate through the Storage Hierarchy:

Find the link to the Storage in the ServiceRoot and navigate to it.



```
"Storage": {
  "@odata.id": "/redfish/v1/Storage"
},
```

Find the storage instance.

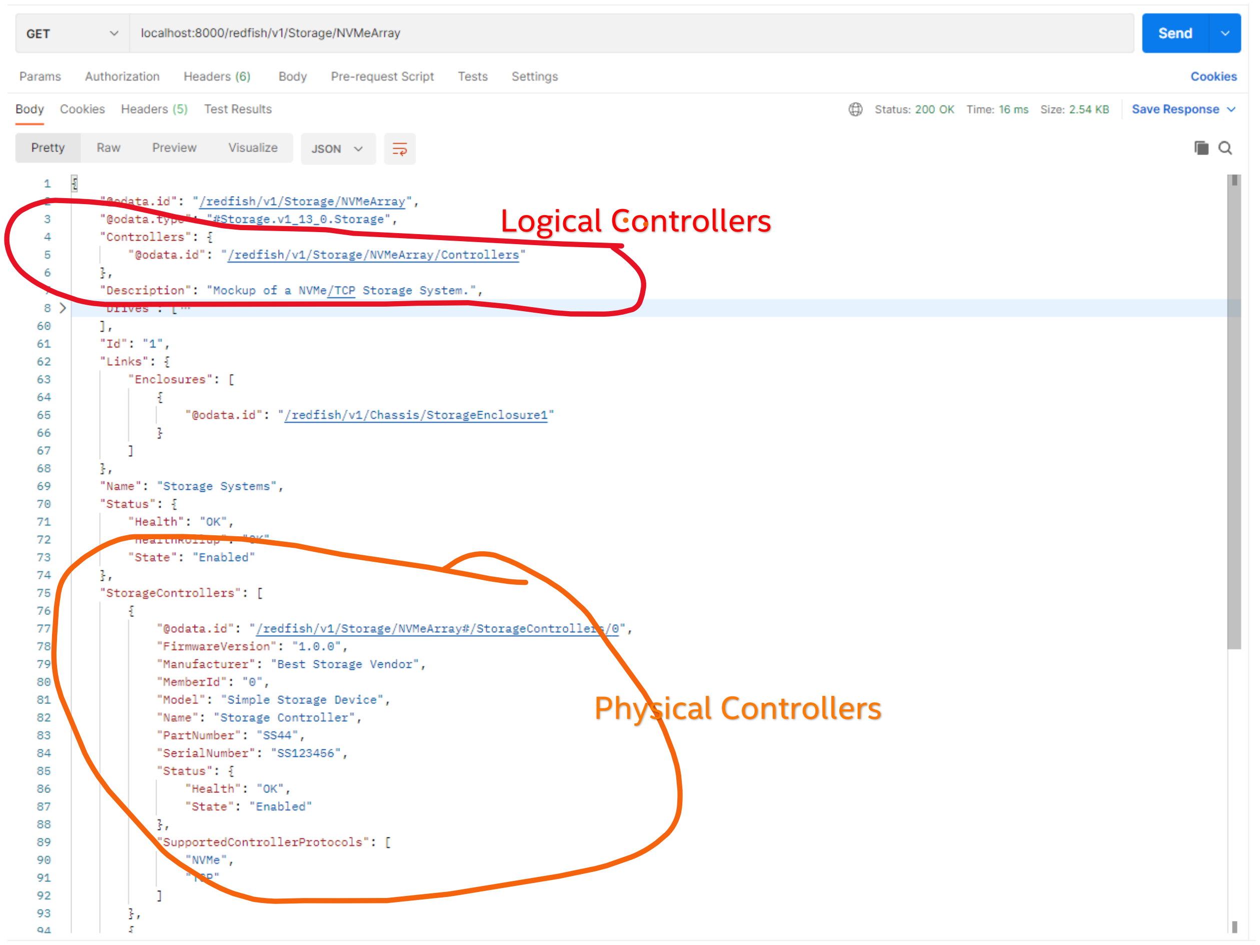


localhost:8000/redfish/v1/Storage

Status: 200 OK Time: 17 ms Size: 365 B

```
{
    "@odata.id": "/redfish/v1/Storage",
    "@odata.type": "#StorageCollection.StorageCollection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Storage/NVMeArray"
        }
    ],
    "Members@odata.count": 1,
    "Name": "Storage Collection"
}
```

Note that this device has both physical and logical controllers.



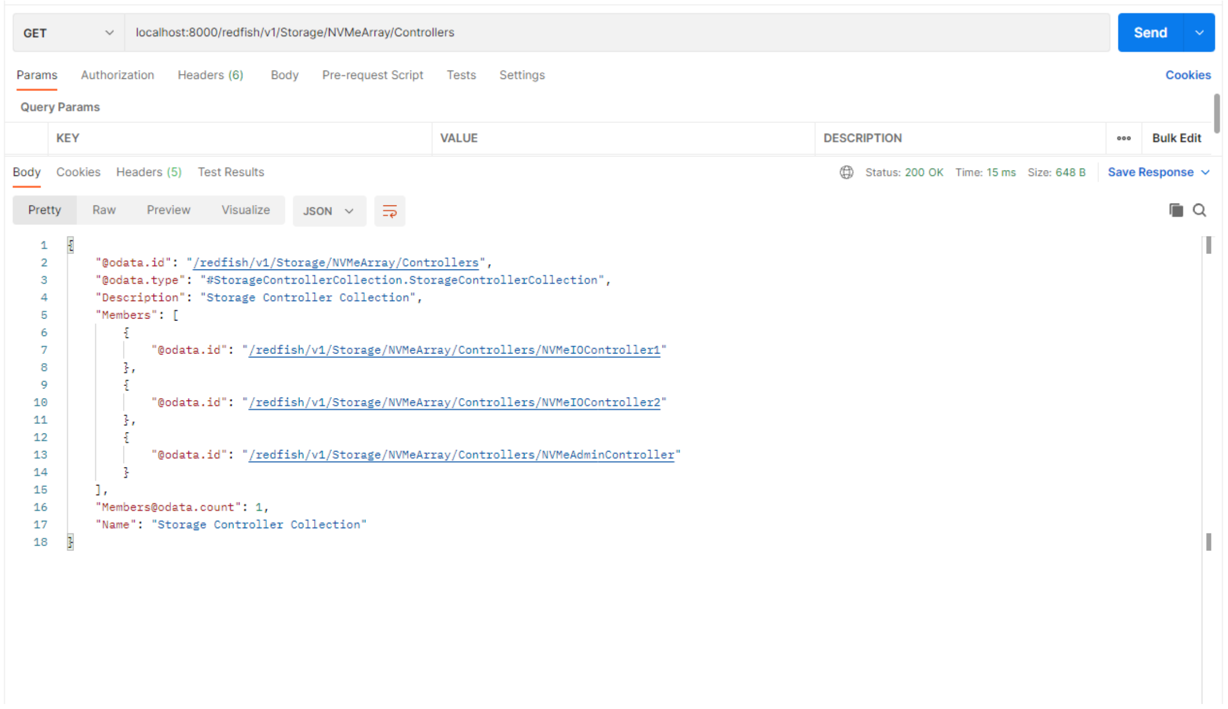
GET localhost:8000/redfish/v1/Storage/NVMeArray

Status: 200 OK Time: 16 ms Size: 2.54 KB

```
{
    "@odata.id": "/redfish/v1/Storage/NVMeArray",
    "@odata.type": "#Storage.v1_13_0.Storage",
    "Controllers": {
        "@odata.id": "/redfish/v1/Storage/NVMeArray/Controllers"
    },
    "Description": "Mockup of a NVMe/TCP Storage System.",
    "Drives": [⋯
    ],
    "Id": "1",
    "Links": {
        "Enclosures": [
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1"
            }
        ]
    },
    "Name": "Storage Systems",
    "Status": {
        "Health": "OK",
        "HealthRollup": "OK",
        "State": "Enabled"
    },
    "StorageControllers": [
        {
            "@odata.id": "/redfish/v1/Storage/NVMeArray#/StorageControllers/0",
            "FirmwareVersion": "1.0.0",
            "Manufacturer": "Best Storage Vendor",
            "MemberId": "0",
            "Model": "Simple Storage Device",
            "Name": "Storage Controller",
            "PartNumber": "SS44",
            "SerialNumber": "SS123456",
            "Status": {
                "Health": "OK",
                "State": "Enabled"
            },
            "SupportedControllerProtocols": [
                "NVMe",
                "TCP"
            ]
        },
        {
```

Navigate to Logical Controllers:

Go to the NVMe logical controllers collection.

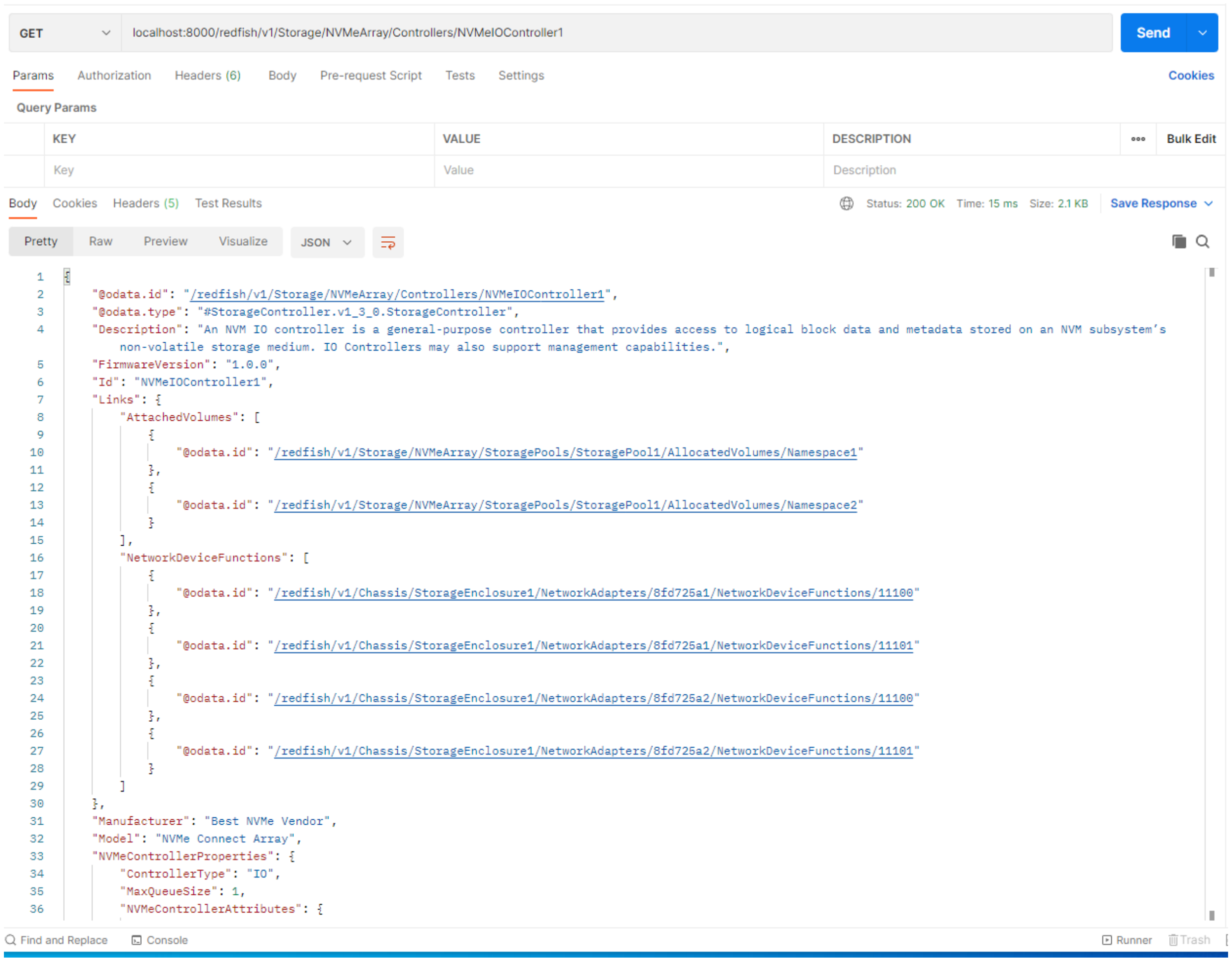


GET localhost:8000/redfish/v1/Storage/NVMeArray/Controllers

Status: 200 OK Time: 15 ms Size: 648 B

```
{
    "@odata.id": "/redfish/v1/Storage/NVMeArray/Controllers",
    "@odata.type": "#StorageControllerCollection.StorageControllerCollection",
    "Description": "Storage Controller Collection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Storage/NVMeArray/Controllers/NVMeIOController1"
        },
        {
            "@odata.id": "/redfish/v1/Storage/NVMeArray/Controllers/NVMeIOController2"
        },
        {
            "@odata.id": "/redfish/v1/Storage/NVMeArray/Controllers/NVMeAdminController"
        }
    ],
    "Members@odata.count": 1,
    "Name": "Storage Controller Collection"
}
```

Look at the details of the two NVMe IO Controllers.

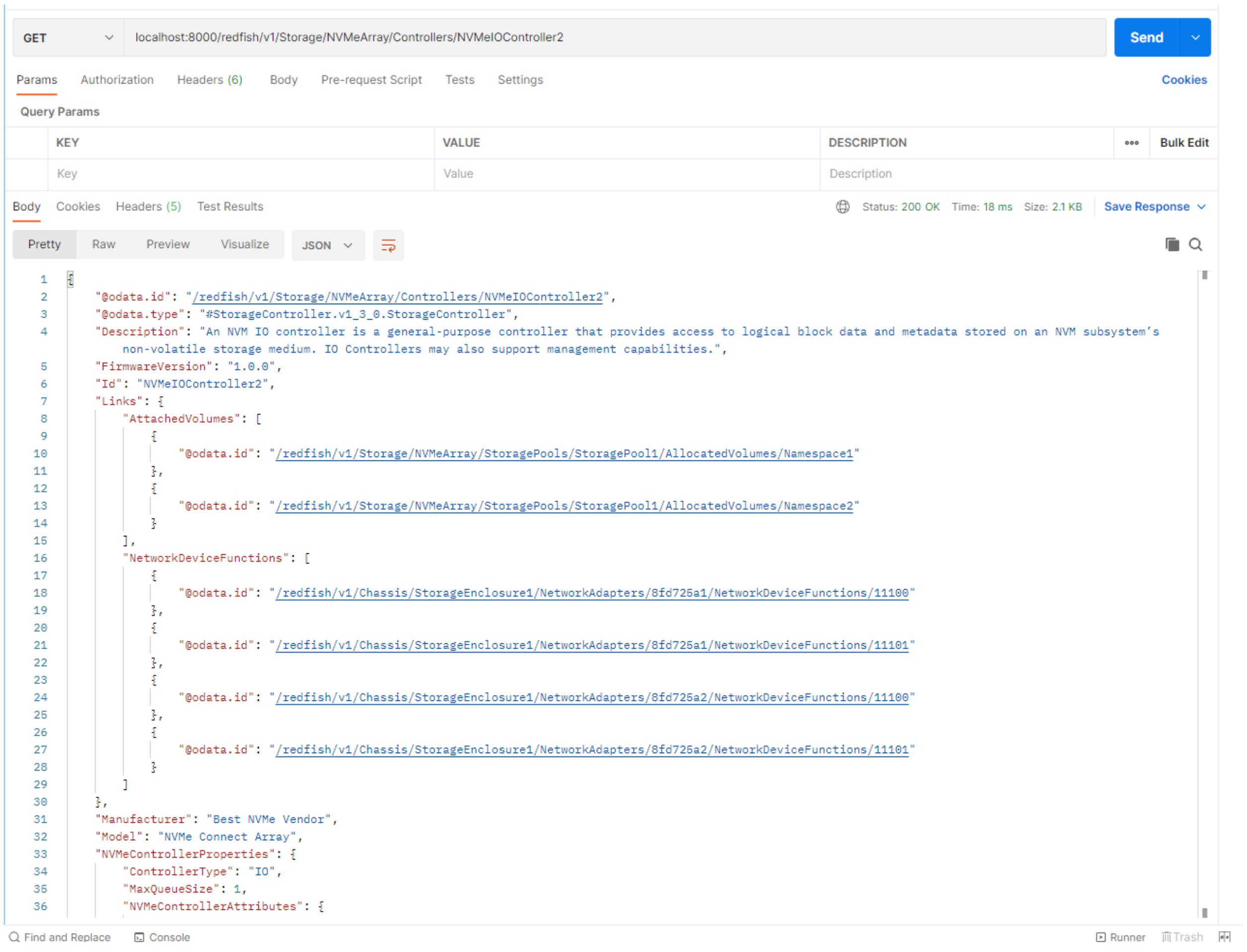


GET localhost:8000/redfish/v1/Storage/NVMeArray/Controllers/NVMeIOController1

Status: 200 OK Time: 15 ms Size: 2.1 KB

```
{
    "@odata.id": "/redfish/v1/Storage/NVMeArray/Controllers/NVMeIOController1",
    "@odata.type": "#StorageController.v1_3_0.StorageController",
    "Description": "An NVM IO controller is a general-purpose controller that provides access to logical block data and metadata stored on an NVM subsystem's
        non-volatile storage medium. IO Controllers may also support management capabilities.",
    "FirmwareVersion": "1.0.0",
    "Id": "NVMeIOController1",
    "Links": {
        "AttachedVolumes": [
            {
                "@odata.id": "/redfish/v1/Storage/NVMeArray/StoragePools/StoragePool1/AllocatedVolumes/Namespace1"
            },
            {
                "@odata.id": "/redfish/v1/Storage/NVMeArray/StoragePools/StoragePool1/AllocatedVolumes/Namespace2"
            }
        ],
        "NetworkDeviceFunctions": [
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11100"
            },
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11101"
            },
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a2/NetworkDeviceFunctions/11100"
            },
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a2/NetworkDeviceFunctions/11101"
            }
        ]
    },
    "Manufacturer": "Best NVMe Vendor",
    "Model": "NVMe Connect Array",
    "NVMeControllerProperties": {
        "ControllerType": "IO",
        "MaxQueueSize": 1,
        "NVMeControllerAttributes": {
```

and …



```
GET    localhost:8000/redfish/v1/Storage/NVMeArray/Controllers/NVMeIOController2
```

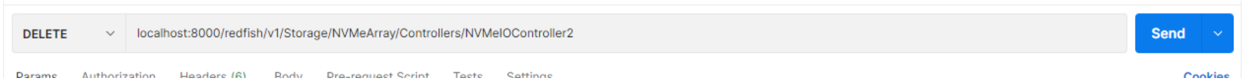
Status: 200 OK Time: 18 ms Size: 2.1 KB

```
{
    "@odata.id": "/redfish/v1/Storage/NVMeArray/Controllers/NVMeIOController2",
    "@odata.type": "#StorageController.v1_3_0.StorageController",
    "Description": "An NVM IO controller is a general-purpose controller that provides access to logical block data and metadata stored on an NVM subsystem's
        non-volatile storage medium. IO Controllers may also support management capabilities.",
    "FirmwareVersion": "1.0.0",
    "Id": "NVMeIOController2",
    "Links": {
        "AttachedVolumes": [
            {
                "@odata.id": "/redfish/v1/Storage/NVMeArray/StoragePools/StoragePool1/AllocatedVolumes/Namespace1"
            },
            {
                "@odata.id": "/redfish/v1/Storage/NVMeArray/StoragePools/StoragePool1/AllocatedVolumes/Namespace2"
            }
        ],
        "NetworkDeviceFunctions": [
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11100"
            },
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a1/NetworkDeviceFunctions/11101"
            },
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a2/NetworkDeviceFunctions/11100"
            },
            {
                "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/NetworkAdapters/8fd725a2/NetworkDeviceFunctions/11101"
            }
        ]
    },
    "Manufacturer": "Best NVMe Vendor",
    "Model": "NVMe Connect Array",
    "NVMeControllerProperties": {
        "ControllerType": "IO",
        "MaxQueueSize": 1,
        "NVMeControllerAttributes": {
```

Since these are configured to provide the same paths, they are redundant. One can be deleted.

Delete NVMeIOController2

Select "DELETE" as the REST command and put the URI to the object on the nav bar.



```
DELETE    localhost:8000/redfish/v1/Storage/NVMeArray/Controllers/NVMeIOController2
```

Click SEND.

Check the Controllers Collection to confirm.