Document Identifier: DSP2066

Date: 2022-04-07

Version: 1.0.0

Redfish Fabrics White Paper

Supersedes: None

Document Class: Informational

Document Status: Published

Document Language: en-US

CONTENTS

1 Foreword

The Redfish Fabrics White Paper was prepared by the Redfish Forum of the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

1.1 Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

- Martin Halstead - Hewlett Packard Enterprise
- John Mayfield - Hewlett Packard Enterprise
- Michael Raineri - Dell Technologies

2 Introduction

Modern datacenters consist of hundreds to thousands of different server and storage resources used for different purposes. Often a portion of these resources are grouped into clusters that are used to work together on specific workloads. High-speed fabrics are typically used to transfer data among clustered resources to optimally satisfy the needs of the workload. Clustered resources within a datacenter can be used for a single workload or can be utilized for multiple workloads. In order to configure and manage clustered resources, Redfish has a common data model that describes fabrics to management clients. The fabric data model contains methods to configure, manage, and support the lifecycle of intra-fabric connectivity of clusters of resources.

The fabric data model describes the physical topology of the cluster, connectivity constraints, and isolation as well as zones to define sub-domains within a larger fabric. This paper describes the common fabric data model and provides examples for common types of fabrics.

3 Fabric representation

The `Fabrics` property found in the service root contains a set collection of `Fabric` resources. Each `Fabric` resource represents a fabric that is managed by the Redfish service and can contain the following information:

- `FabricType` : Describes the type of protocol sent over the fabric.
- `Switches` : Contains the switches and their connectivity information for the fabric. See Switches and Ports for more information.
- `Endpoints` : Conains the logical representation for devices on a fabric. See Endpoints for more information.
- `Zones` : Contains communication contraints for endpoints within a fabric. See Zones for more information.
- `Connections` : Contains rules for the types of resources an initiator endpoint is allowed to access when connecting to a target endpoing. See Connections for more information.
- `AddressPools` : Contains addressing rules for the fabric. See Address pools for more information.

Example `Fabric` resource:


```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet",
    "@odata.type": "#Fabric.v1_2_2.Fabric",
    "Id": "Ethernet",
    "Name": "Ethernet Fabric",
    "FabricType": "Ethernet",
    "Description": "An Ethernet Based Fabric",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "Zones": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Zones"
    },
    "Endpoints": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Endpoints"
    },
    "Switches": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches"
    },
    "AddressPools": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/AddressPools"
    }
}
```

4 Endpoints

The `Endpoint` resource represents an addressable entity, a single device, or a set of devices where traffic enters or exits a fabric. Traffic on a fabric flows from one endpoint to another endpoint, and the configuration of the fabric dictates routing and addressability rules for the endpoints. Systems, switches, adapters, and other components connected to a fabric might contain multiple endpoints to represent the different signifying an ingress and egress points on a fabric.

Endpoints can represent varying types of entities on a fabric, some physical and some logical. The `ConnectedEntities` property describes the type of entity on the fabric and how it links to other areas of the Redfish model. `ConnectedEntities` is an array to allow for multiple Redfish resources to be referenced if they work together to form an endpoint on a fabric. Each member of the `ConnectedEntities` array can contain the following properties:

Property	Description
`EntityType`	The type of the entity on the fabric. See the next table for more information on the values.
`EntityRole`	Describes whether the endpoint acts as an initiator, target, or both on the fabric. Initiator endpoints produce traffic to access resources exposed by target endpoints.
`EntityLink`	A link to another resource in the Redfish data model that provides more information about the endpoint. This property might not be present if the `Endpoint` resource represents an entity that is not managed by the service.
`Identifiers`	The globally unique identifier for the entity.

The value of the `EntityType` property gives guidance for the type of resource that can be found with the `EntityLink` property. The `EntityType` property can contain the following values.

Value	Description	`EntityLink` resource
`StorageInitiator`	A storage initiator.	A member of `StorageControllers` in `Storage` or a `StorageController` resource
`RootComplex`	A PCI(e) root complex.	`ComputerSystem`
`NetworkController`	A network controller.	`NetworkDeviceFunction` or `EthernetInterface`
`Drive`	A drive.	`Drive`

Value	Description	`EntityLink` resource
`StorageExpander`	A storage expander.	`Chassis`
`DisplayController`	A display controller.	N/A
`Bridge`	A PCI(e) bridge.	N/A
`Processor`	A processor.	`Processor`
`Volume`	A volume.	`Volume`
`AccelerationFunction`	An acceleration function realized through a device, such as an FPGA.	`AccelerationFunction`
`MemoryChunk`	A memory chunk.	`MemoryChunk`
`Switch`	A switch.	`Switch`
`FabricBridge`	A fabric bridge.	`FabricAdapter`
`Manager`	A manager.	`Manager`
`StorageSubsystem`	A storage subsystem.	`Storage`

Endpoints with IP connectivity can report their IP addresses with the `IPTransportDetails` property.

Endpoints that represent a logical entity on a fabric can report an additional `Identifiers` property at the root of the resource to show the globally unique identifier for the endpoint.

The following example shows an `Endpoint` resource for a SAS drive. The `EndpointProtocol` property contains `SAS` to reflect the protocol accepted by the endpoint. `ConnectedEntities` references the `Drive` resource that the endpoint represents. `ConnectedPorts` within `Links` shows the switch ports to which the endpoint is connected.

```
{
    "@odata.id": "/redfish/v1/Fabrics/SAS/Endpoints/Drive1",
    "@odata.type": "#Endpoint.v1_6_1.Endpoint",
    "Id": "Drive1",
    "Name": "SAS Drive",
    "Description": "The SAS Drive in Enclosure 2 Bay 0",
    "EndpointProtocol": "SAS",
    "ConnectedEntities": [
        {
            "EntityType": "Drive",
            "EntityRole": "Target",
            "EntityLink": {
                "@odata.id": "/redfish/v1/Chassis/2/Drives/0"
            },
            "Identifiers": [
                {
```

```
                    "DurableNameFormat": "NAA",
                    "DurableName": "32ADF365C6C1B7C3"
                }
            ]
        }
    ],
    "Links": {
        "ConnectedPorts": [
            {
                "@odata.id": "/redfish/v1/Fabrics/SAS/Switches/Switch1/Ports/8"
            },
            {
                "@odata.id": "/redfish/v1/Fabrics/SAS/Switches/Switch2/Ports/8"
            }
        ]
    }
}
```

5 Connectivity

Redfish describes the physical topology of a fabric using the `Switch` and `Port` resources. The `Port` resources subordinate to the `Switch` resource show how a fabric switch provides connectivity to other devices in the infrastructure. Other resources that represent devices on a fabric, such as `FabricAdapter` and `NetworkAdapter`, contain their own `Port` resources to show connectivity to the fabric. These resources can be used to configure routing information, virtual channel information, virtual LAN information, and congestion information based on the customer's desired topology. A client can verify the topology and port map by following the links contained in the `ConnectedPorts` and `AssociatedEndpoints` properties from the various `Port` resources. Clients can discover these resources from the `Fabrics`, `Chassis`, and `System` properties off the service root. Figure 1 shows an example connectivity diagram with two fabrics named `Ethernet` and `Gen-Z`.

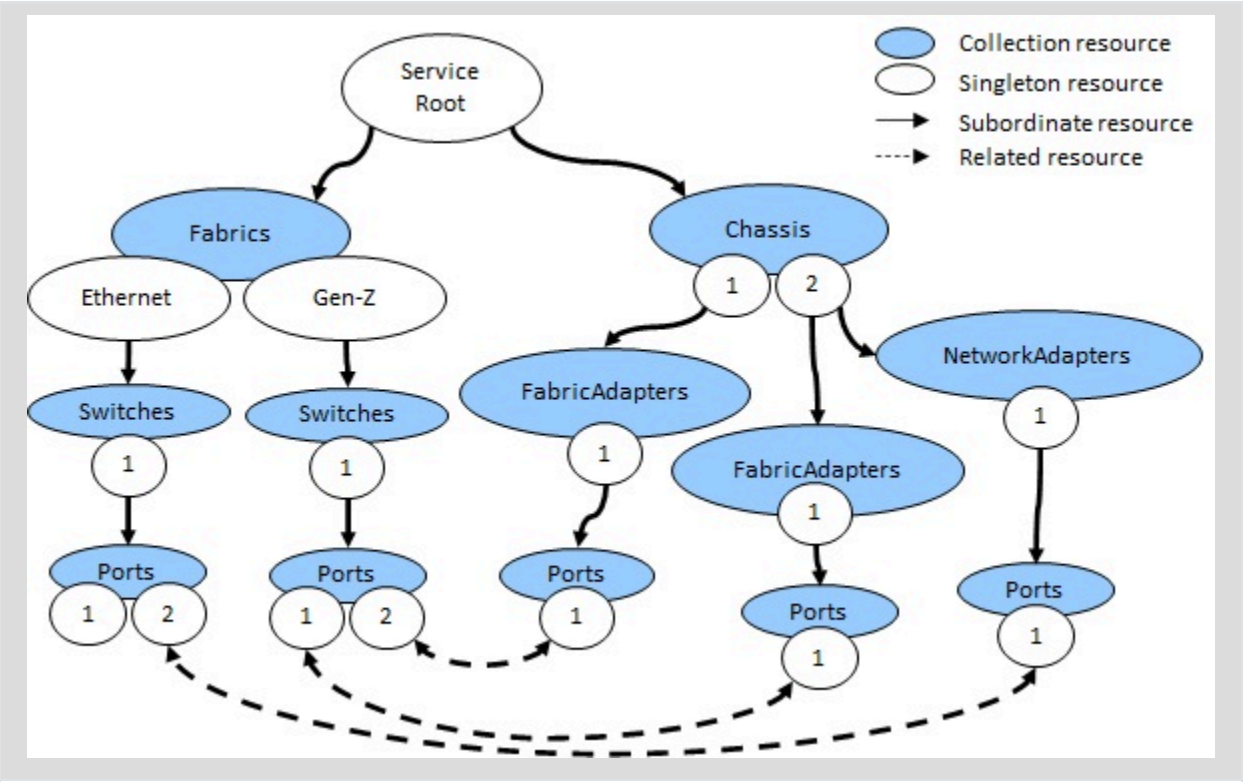

Figure 1

5.1 Switches

The `Switch` resource represents a single, generic switch. It contains information about the switch, such as its manufacturer, model, and part number. It also provides a link to the collection of ports on the switch.

Example `Switch` resource:

```
{
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches/Switch1",
    "@odata.type": "#Switch.v1_3_0.Switch",
    "Id": "Switch1",
    "Name": "Ethernet Switch",
    "SwitchType": "Ethernet",
    "Manufacturer": "Contoso",
    "Model": "8320",
    "SKU": "67B",
    "SerialNumber": "2M220100SL",
    "PartNumber": "76-88883",
    "Ports": {
        "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches/Switch1/Ports"
    }
}
```

5.2 Ports

The `Port` resource represent the physical interface of an adapter or switch to a fabric. Resources such as `Switch`, `FabricAdapter`, `NetworkAdapter`, and `Processor` contain their own collection of `Port` resources to represent their respective interfaces for the device. The `Port` resource describes the port's attributes, such as the interface speed and link status.

The `Port` resource contains several properties within `Links` to describe how the port is connected to other ports or devices in a fabric. Clients can follow the following properties within `Links` for building the topology of a fabric.

Property	Description
`ConnectedSwitches`	Switches connected to the port.
`ConnectedSwitchPorts`	Switch ports connected to the port. The members should be subordinate to the switches found in `ConnectedSwitches`. If the connected port is a device port, use `ConnectedPorts` instead.
`ConnectedPorts`	Device ports connected to the port. The members should not reference ports on switches. If the connected port is a switch port, use `ConnectedSwitchPorts` instead.
`AssociatedEndpoints`	Endpoints connected to the port. This property can be used for fabrics where `Endpoint` resources represent physical devices and `ConnectedPorts` is not applicable.

Example `Port` resource:

```
{
```

```
    "@odata.id": "/redfish/v1/Fabrics/Ethernet/Switches/Switch1/Ports/1",
    "@odata.type": "#Port.v1_2_0.Port",
    "Id": "1",
    "Name": "Ethernet Port 1",
    "Description": "Ethernet Port 1",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "PortId": "1",
    "PortProtocol": "Ethernet",
    "PortType": "BidirectionalPort",
    "CurrentSpeedGbps": 25,
    "Width": 4,
    "MaxSpeedGbps": 25,
    "Links": {
        "ConnectedPorts": [
            {
                "@odata.id": "/redfish/v1/Chassis/1/NetworkAdapters/3/Ports/1"
            }
        ]
    }
}
```


5.3 Port metrics

Many devices capture the metrics associated with their ports. For example, Gen-Z devices can capture metrics such as packet CRC errors, end-to-end CRC errors, and non-CRC transient errors. This provides clients an indication of the health of the port at a particular point in time. The `PortMetrics` resource contains these metrics.

Example `PortMetrics` resource:


```
{
    "@odata.id": "/redfish/v1/Fabrics/GenZ/Switches/Switch1/Ports/1/Metrics",
    "@odata.type": "#PortMetrics.v1_0_0.PortMetrics",
    "Id": "Metrics",
    "Name": "Gen-Z Port 1 Metrics",
    "Description": "Gen-Z Port 1 Metrics",
    "GenZ": {
        "PacketCRCErrors": 24,
        "EndToEndCRCErrors": 3,
        "RXStompedECRC": 1,
        "TXStompedECRC": 2,
        "NonCRCTransientErrors": 2,
        "LLRRecovery": 1,
        "MarkedECN": 1,
```

```
        "PacketDeadlineDiscards": 1,
        "AccessKeyViolations": 1,
        "LinkNTE": 1,
        "ReceivedECN": 1
    }
}
```

5.4 Network and fabric adapters

The `NetworkAdapter` and `FabricAdapter` resources contain a collection of `Port` resources to express physical connectivity to a fabric or other adapters. These adapters support specific types of fabrics. For example, a device on an Ethernet fabric would use a `NetworkAdapter` resource and a device on a Gen-Z fabric would use a `FabricAdapter` resource. These adapters describe the physical endpoint of the node on the fabric along with the adapter capabilities.

The `FabricAdapter` resource provides additional fabric-related settings, such as routing tables, congestion management, embedded switch configuration, and Virtual Channel/Traffic Class management.

Chassis-level and system-level connectivity is determined using the `Port` resources found on the chassis's or system's related adapters. The `ComputerSystem` resource and `Chassis` resource contain a `FabricAdapters` property to represent the set of available fabric adapters. The `Chassis` resource contains a `NetworkAdapters` property to represent the set of available network adapters. The `ComputerSystem` resource contains `NetworkInterfaces` property to represent the set of available partitions of network adapters.

6 Fabric configuration and routing

The `Zone`, `Connection`, and `AddressPool` resources model communication intent across fabrics. The following sections describe the relationships between these resources and the `Endpoint` resource for managing routing and other networking configurations for a fabric.

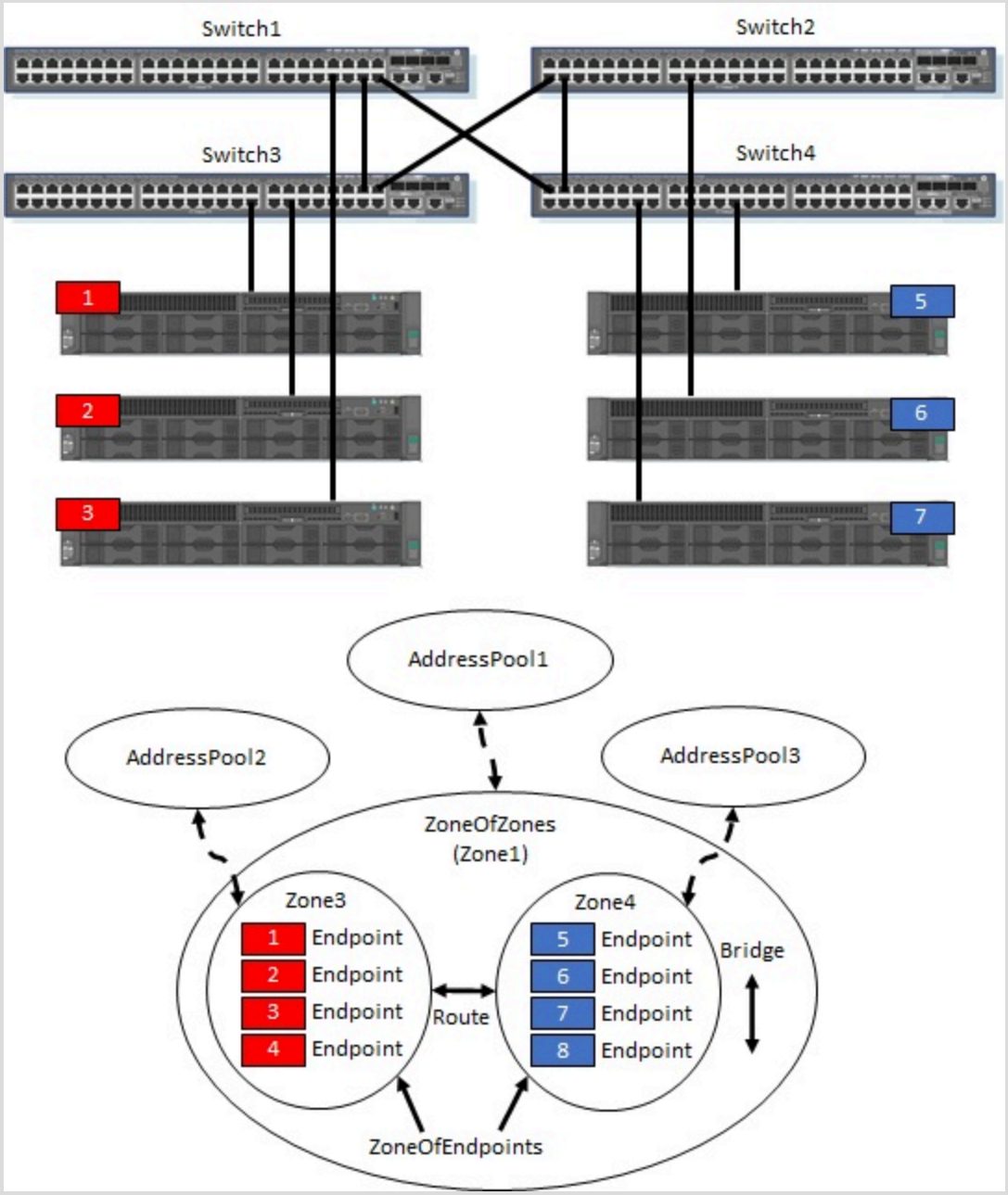

Figure 2

6.1 Zones

The `Zone` resource constrains communications by subdividing a fabric into a series of regions or subnets, either for

the entire data center fabric or for multi-tenant access to it. The `ZoneType` property describes the usage of the zone in the fabric and can contain the following values.

Value	Description
`Default`	The zone is associated with the entire fabric. Newly created endpoints appear in this zone until additional configurations are made by a client to assign the endpoint to another zone.
`ZoneOfEndpoints`	The zone contains a set of endpoints where routing is enabled for traffic to flow between the endpoints.
`ZoneOfZones`	The zone contains other zones for scalability. In Ethernet fabrics, this signifies one or more virtual routing domain or virtual routing (VRF) instances. Zones referenced by this zone will contain `ZoneOfEndpoints` for their `ZoneType` property. Endpoints that are within one zone of type `ZoneOfZones` can overlap with endpoints that are in a different zone of type `ZoneOfZones`. This construct emulates multi-tenancy across fabrics.
`ZoneOfResourceBlocks`	The zone represents a set of resource blocks that can be composed together. This value is specific to composability and does not apply to fabrics.

Each `Fabric` resource should contain a `Zone` resource whose `ZoneType` property contains the value `Default`. This is a well-known location for clients to understand default routing policies for endpoints that have not been assigned to a zone. The `DefaultRoutingEnabled` property controls whether traffic is allowed between the endpoints in this zone.

Zones of type `ZoneOfZones` are prohibited from containing other zones of type `ZoneOfZones`. This is to prevent complexities with nesting zones within zones within zones and also prevents creating circular zoning situations.

The `Links` property can contain the following properties to show additional configuration of the zone and components that are in the zone.

- `Endpoints`: The endpoints that belong in this zone. This only applies when `ZoneType` contains `Default` or `ZoneOfEndpoints`.
- `ContainsZones`: The zones that belong in this zone. This only applies when `ZoneType` contains `ZoneOfZones`.
- `ContainedByZones`: The zone that contains this zone. This only applies when `ZoneType` contains `ZoneOfEndpoints`.
- `AddressPools`: Additional networking configuration for this zone.
- `InvolvedSwitches`: The switches that are used by this zone.
- `ResourceBlocks`: The resource blocks that belong in this zone. This only applies when `ZoneType` contains `ZoneOfResourceBlocks` and does not apply to fabrics.

Figure 3 shows a sample set of zones and endpoints. There are 16 endpoints grouped into four zones of type `ZoneOfEndpoints`. There are also two zones of type `ZoneOfZones` that provide further groupings. The endpoints that belong in the same zone of type `ZoneOfEndpoints`, such as `Zone3`, are hosts on the same network, therefore traffic is bridged across the fabric between these endpoints. Endpoints that belong to different zones of type `ZoneOfEndpoints`, but are contained in the same zone of type `ZoneOfZones`, such as zones `Zone3` and `Zone4`, are on different networks where the fabric is configured to route traffic between the endpoints. Endpoints that do not

belong in any common zones, such as the endpoints in zones `Zone3` and `Zone6`, are not configured to have any routing between the endpoints. If routing is required between `Zone3` and `Zone6`, a new zone of type `ZoneOfZones` would need to be created with `Zone3` and `Zone6` as part of the new zone.

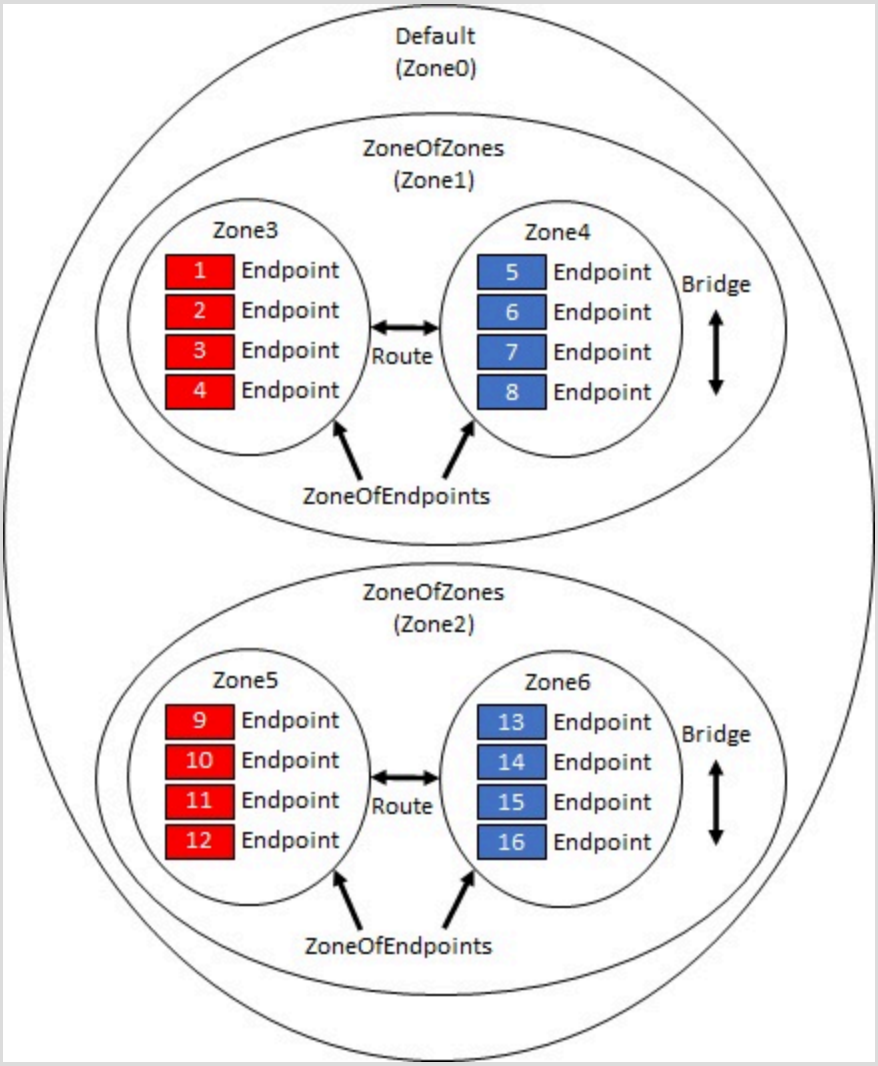

Figure 3

By automating these interrelationships, it should be possible to configure any fabric to accommodate those addressing and connectivity requirements. This allows for a multi-vendor and fully standards-based method to uniformly configure distributed fabrics in tandem with compute and storage infrastructure inside the data center.

6.2 Address pools

The `AddressPool` resource constrains control plane specific pools of addressing for setting up fabric-wide communications as well as host network address pools. Address pools can be applied to particular zones in a fabric in order to configure zone-specific networking.

In an Ethernet fabric, an address pool can contain subnet, default gateway, VLAN, BGP underlay, EVPN control plane, and other networking configurations.

In a Gen-Z fabric, an address pool can specify Global Component Identifier information.

In Figure 2, there are three example address pools that belong to an Ethernet fabric. `AddressPool1` contains cross Ethernet fabric addressing settings such as EBGP underlay addressing, EVPN address pools, and BGP timers. `AddressPool2` and `AddressPool3` contain subnet settings such as IP network ranges, gateways, and VLANs for that network.

6.3 Connections

The `Connection` resource contains access permissions for resources accessible via a target endpoint once two endpoints establish a communication channel. This differs from `Zone` resources in that zones describe the routing for the communication channel itself.

`Connection` resources contain a `ConnectionType` property to describe the type of resources target endpoints can expose to the connecting initiator endpoints and can contain the following values.

Value	Description
`Storage`	The target endpoints are able to provide storage-related resources, such as volumes.
`Memory`	The target endpoints are able to provide memory-related resources, such as memory chunks.

Based on the value of `ConnectionType`, one of the following properties will be present to provide the information about the specific resources made available to initiator endpoints.

- `VolumeInfo`: An array containing references to one or more `Volume` resources along with access capabilities, such as whether the volumes are read-only or read-write for the connecting initiators.
- `MemoryInfo`: An array containing references to one or more `MemoryChunk` resources along with access capabilities, such as whether the volumes are read-only or read-write for the connecting initiators.

The `ConnectionKeys` property may also be present for fabrics that require specifying access keys, such as with Gen-Z fabrics.

The `Links` property contains references to the endpoints affected by the connection. Initiators can be referenced with the `InitiatorEndpoints` or `InitiatorEndpointGroups` properties, and targets can be referenced by the `TargetEndpoints` or `TargetEndpointGroups` properties. When a referenced initiator establishes a connection over the fabric with one of the referenced targets, it's able to access the resources specified by other properties, such as `VolumeInfo`.

Figure 4 shows a sample NVMe-oF fabric. The endpoints `Host1` and `Host2` act as initiator endpoints on the fabric, and endpoint `Target1` is the NVMe-oF target. The connection `Conn1` specifies that both `Host1` and `Host2` are given access to the volume `NS1` when they establish a connection with `Target1`.

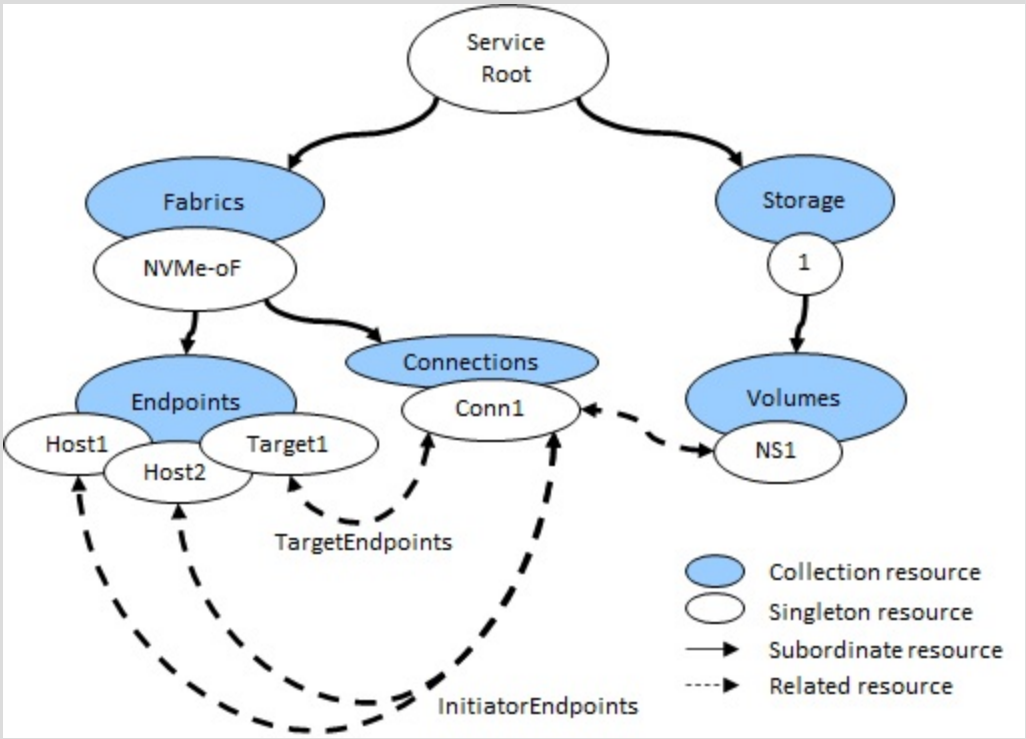

Figure 4

7 Representing different types of fabrics

The following sections contain diagrams for representing different types of fabrics with Redfish.

7.1 Gen-Z

The following figures show a sample Gen-Z fabric and how the `ComputerSystem`, `Chassis`, and `Fabric` resources are related. The sample fabric contains the following components.

- System `1`: A system with a fabric adapter to access resources on the Gen-Z fabric.
- Chassis `Gen-Z`: An enclosure containing memory that is partitioned into chunks that are accessible over a Gen-Z fabric.
- Fabric `Gen-Z`: The representation of the Gen-Z fabric with its switches and configurations.

Figure 5 shows the physical connectivity between the devices on the fabric. System `1` and chassis `Gen-Z` both contain a fabric adapter with one port. Fabric `Gen-Z` contains a single switch with two ports. The ports on system `1` and chassis `Gen-Z` connect to the ports on the switch found in the fabric `Gen-Z`.

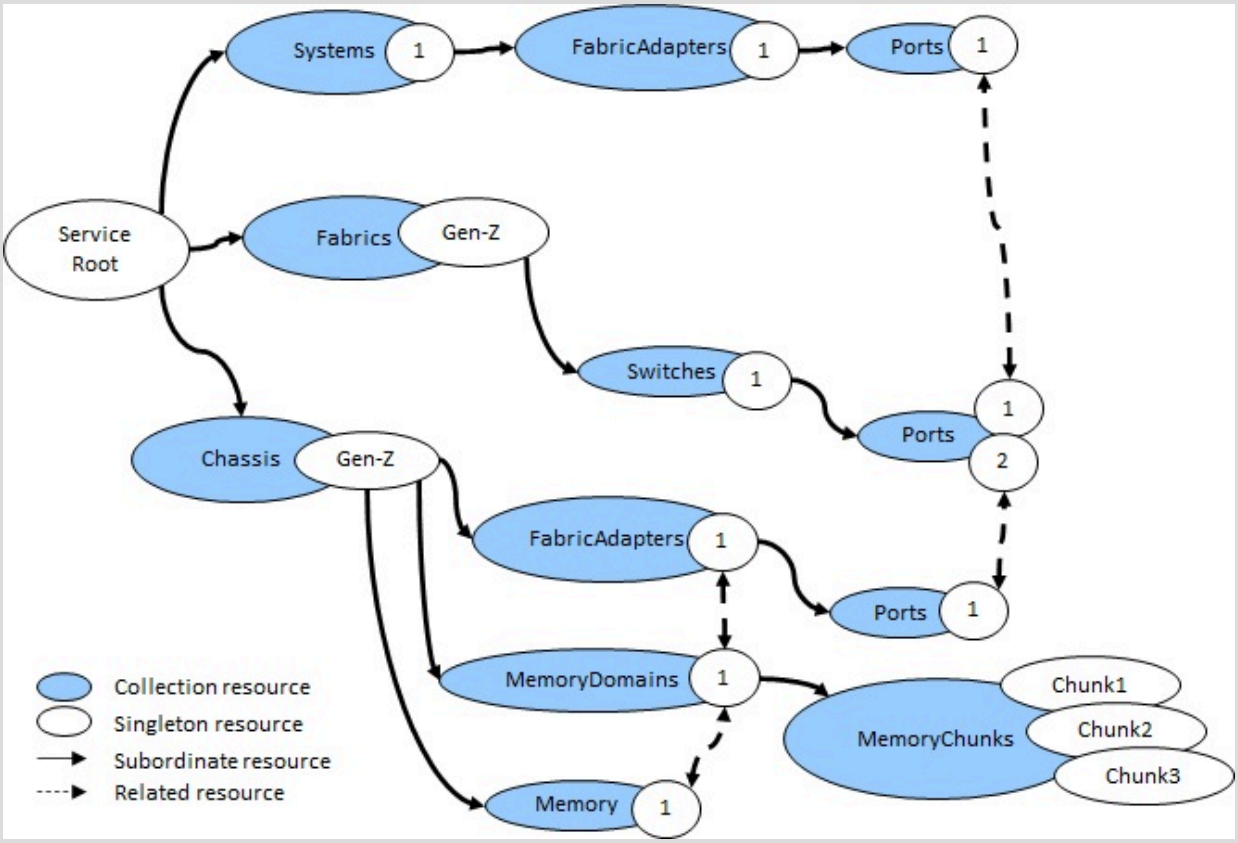

Figure 5

Figure 6 shows routing and addressing configurations for the fabric `Gen-Z`. Fabric `Gen-Z` contains two endpoints: one named `Initiator` to represent the connectivity of system `1` to the fabric, and one named `Target` to represent the connectivity of chassis `Gen-Z` to the fabric. Fabric `Gen-Z` contains a single zone where both endpoints belong to the zone to show that traffic is routable between endpoints `Initiator` and `Target`. Fabric `Gen-Z` also contains a single address pool to control Gen-Z Component Identifier and Subnet Identifier assignments to the endpoints `Initiator` and `Target`.

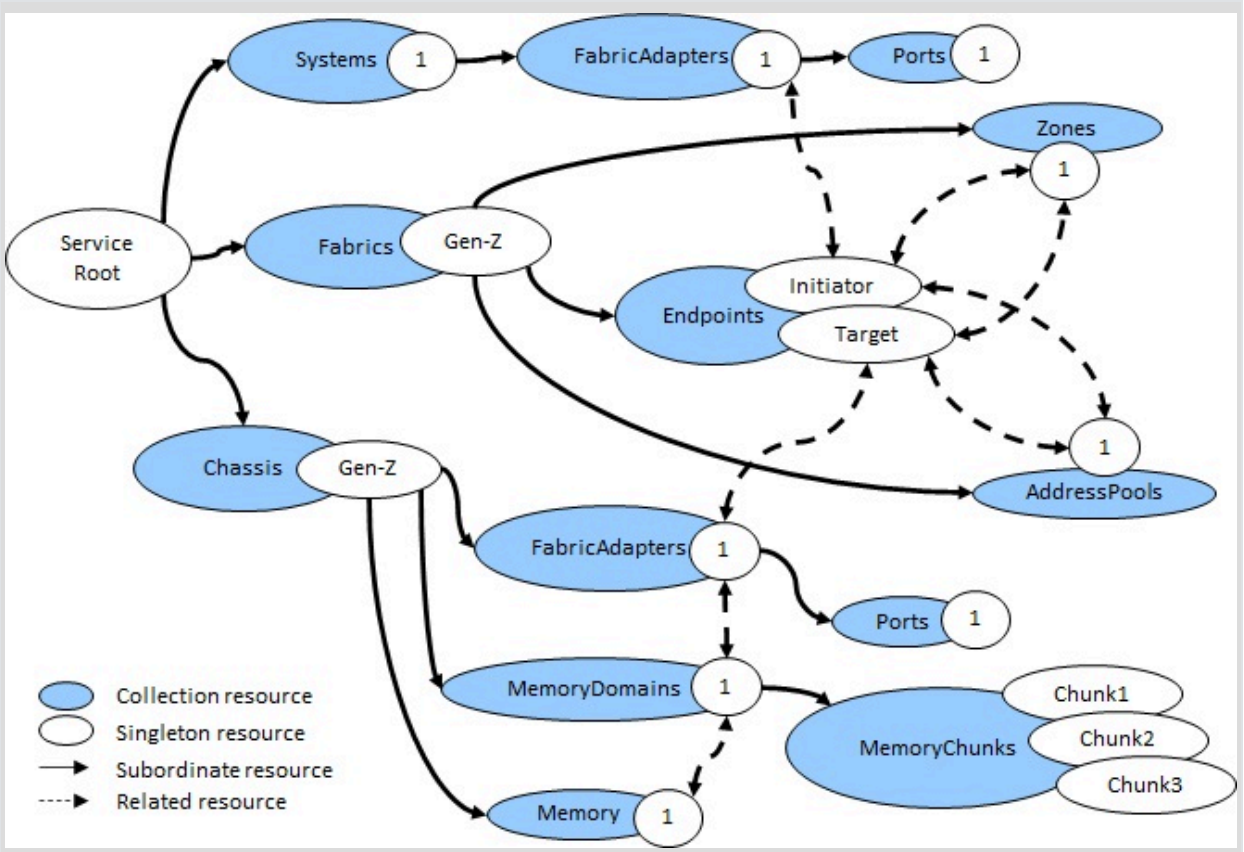

Figure 6

Figure 7 shows access control to resources for the fabric `Gen-Z`. Fabric `Gen-Z` contains one connection named `Conn1`. `Conn1` is configured to allow the endpoint `Initiator` to access memory chunks `Chunk1`, `Chunk2`, and `Chunk3` when connecting to endpoint `Target` over the fabric.

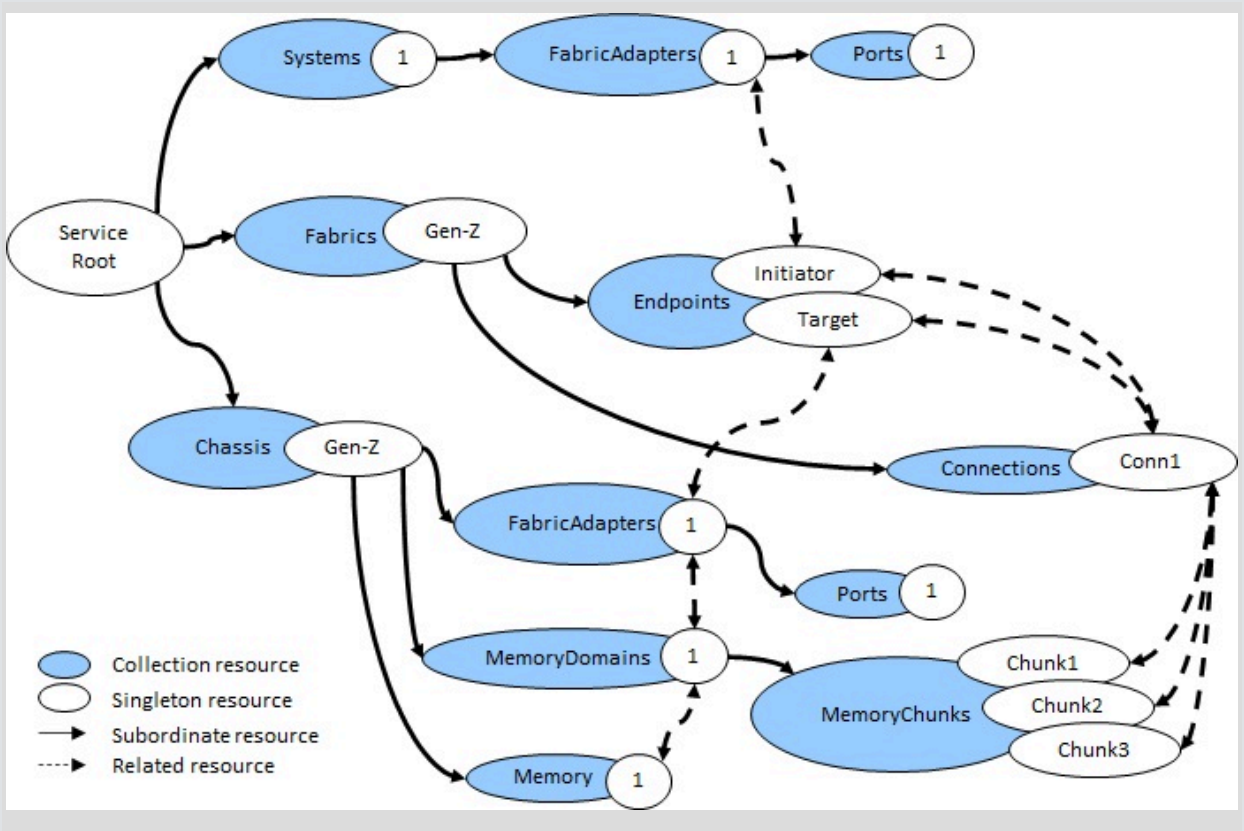

Figure 7

7.2 Ethernet

Figure 8 shows a sample Ethernet fabric and how the `Chassis` and `Fabric` resources are related. The sample fabric contains the following components.

- Chassis `1`: An enclosure with a network adapter containing a single function and port.
- Chassis `2`: A second enclosure with a network adapter containing a single function and port.
- Fabric `Ethernet`: The representation of the Ethernet fabric with its switches and configurations.

The ports for the network adapters in chassis `1` and `2` are connected to the ports on the switch found in fabric `Ethernet`. The ports for each network adapter also show a relationship to a network device function to show the port usage of the functions.

Endpoints `EP1` and `EP2` are within fabric `Ethernet` to represent the fabric-view of the network device functions found in chassis `1` and `2`. Both of these endpoints belong to zone `1`, signifying routing is enabled between them.

Zone `1` is also associated with address pool `1`, which contains networking configurations applied to the endpoints in the zone.

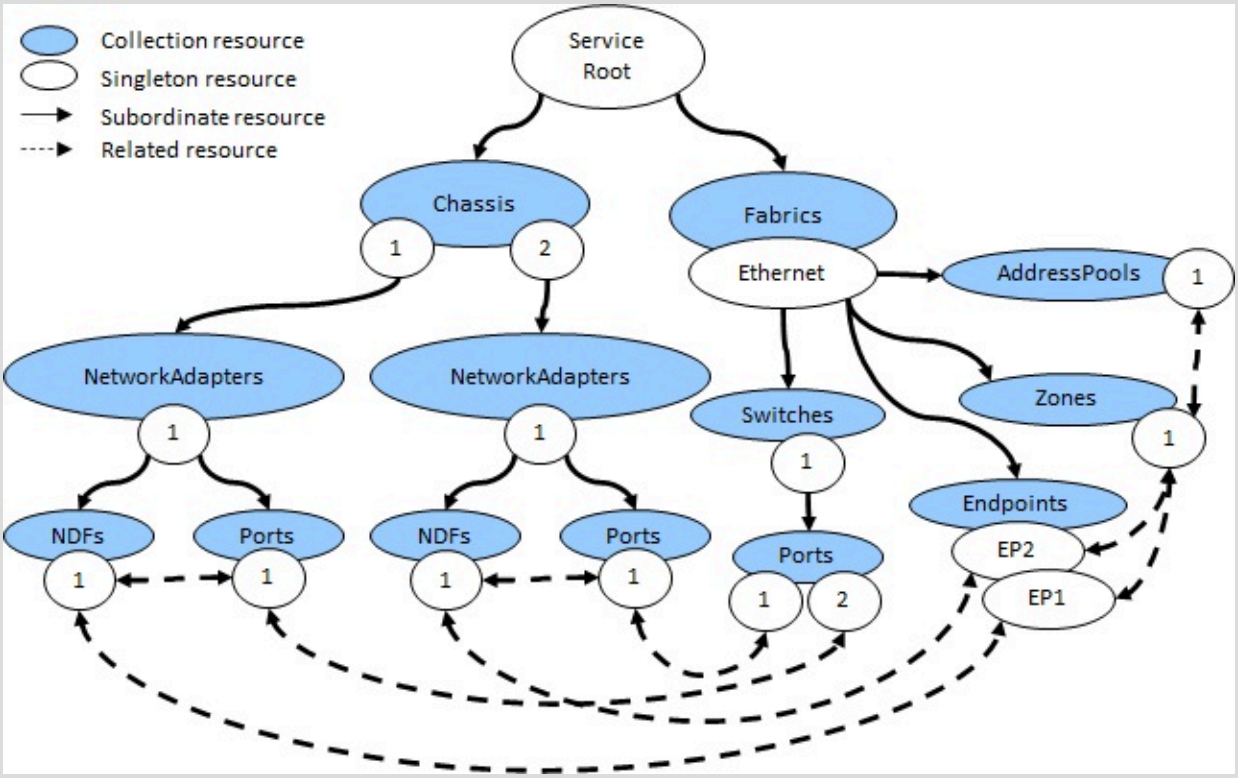

Figure 8

7.3 SAS

Figure 9 shows a sample SAS fabric and how the `ComputerSystem`, `Chassis`, and `Fabric` resources are related. The sample fabric contains the following components.

- System `1`: A system with a storage controller to access the SAS fabric.
- Chassis `1`: An enclosure containing drives that are accessible over a SAS fabric.
- Fabric `SAS`: The representation of the SAS fabric with its switches and configurations.

System `1` contains a single storage subsystem with one storage controller. The storage controller is represented in the fabric as endpoint `Initiator`. The storage controller shows its port is connected to port `3` on the switch found in fabric `SAS`.

Chassis `1` contains two drives, `D1` and `D2`. Each drive is represented in the fabric as endpoints `D1` and `D2`. These endpoints show connectivity to switch ports `1` and `2`.

Fabric `SAS` contains a single zone with all endpoints belonging to the zone. This signifies routing is enabled between all three endpoints.

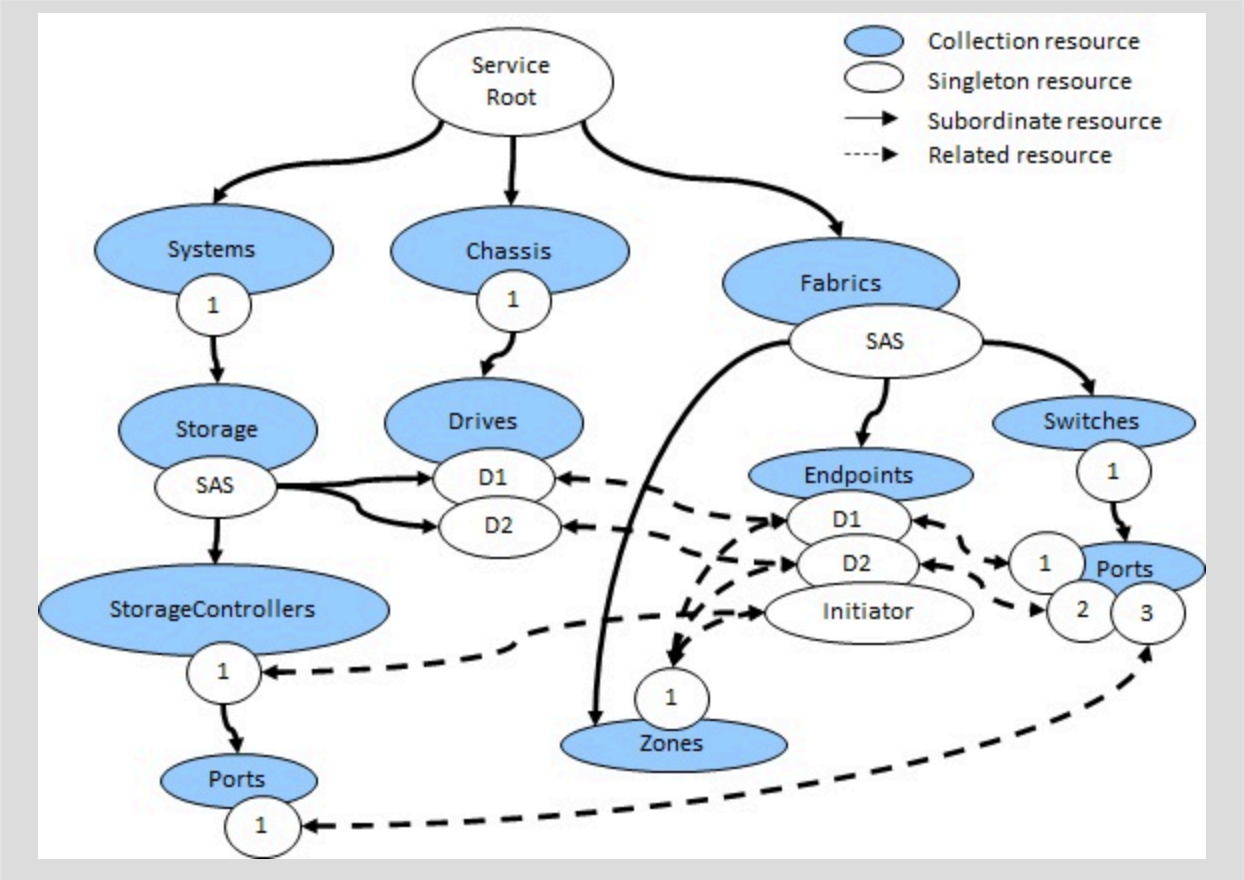

Figure 9

7.4 PCIe

Figure 10 shows a sample PCIe fabric and how the `Fabric` resources are related to other types of resources. The sample fabric contains the following components.

- System `1` : A system with a storage controller and NIC connected to a PCIe switch.
- Fabric `PCIe` : The representation of the PCIe fabric with its switches and configurations.

System `1` contains an Ethernet interface and is represented in the fabric as endpoint `NIC1`. The endpoint shows connectivity to switch port `Down1`.

System `1` also contains a single storage subsystem with one storage controller. The storage controller is represented in the fabric as endpoint `SAS-HBA`. The endpoint shows connectivity to switch port `Down2`.

Fabric `PCIe` contains an additional endpoint named `Root` and represents the PCIe root port for system `1`. The endpoint also shows connectivity to switch port `Up`. The fabric also contains a single zone with all endpoints belonging to the zone. This signifies routing is enabled between all three endpoints.

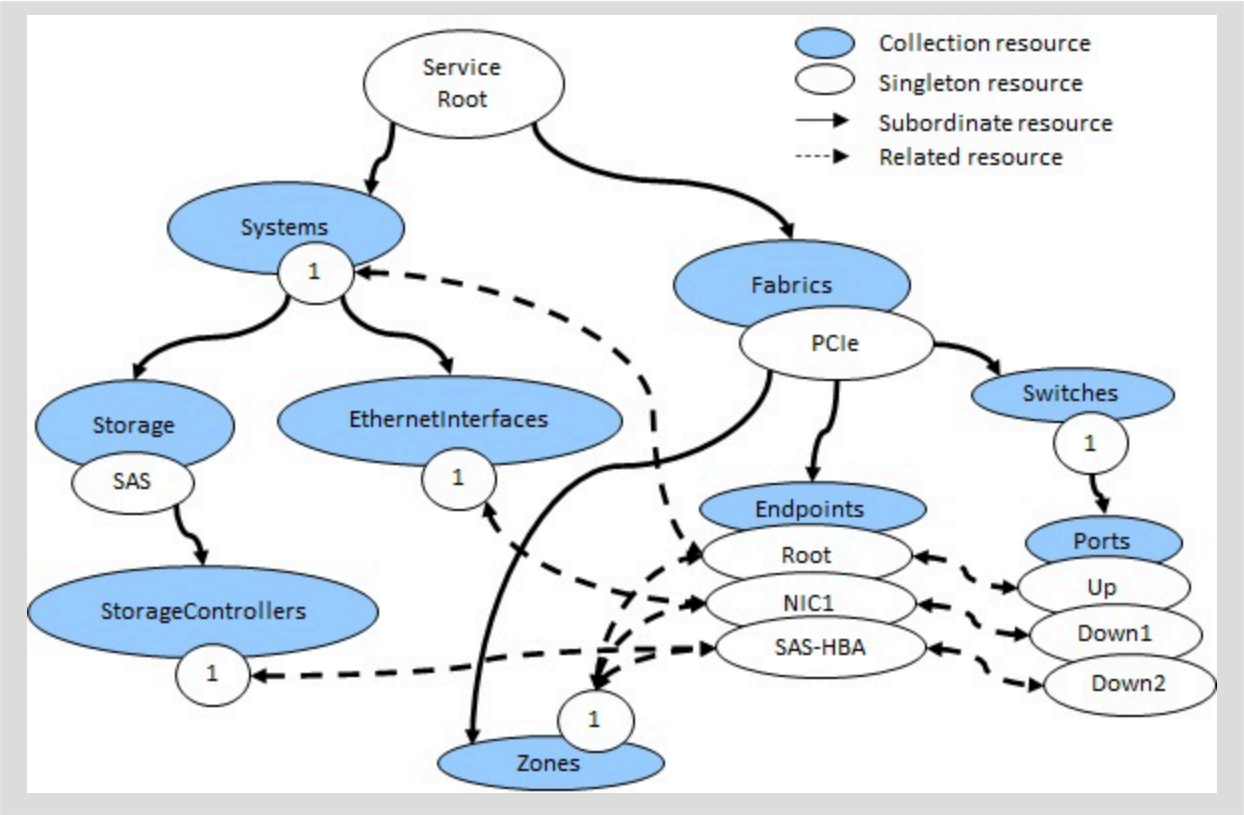

Figure 10

8 Fabric model and composability

Redfish has a data model for composability where clients are able to specify components in a service and build systems on demand. There are aspects of the fabric model that overlap concepts in composability, such as establishing routing paths between components. While there is overlap in some of the usage, both models can coexist on the same service. In cases where both models are present on the same service, performing requests in one area of the model might have impacts on the other. For example, if a client performs a composition request to build a new system, the service may perform fabric configurations and routing in order to satisfy the client's request, which are then reflected in the fabric model.

9 Appendix A: References

- "Simple SAS Fabric" and "NVMe-oF JBOF" Mockups: http://redfish.dmtf.org/redfish/v1
- DMTF DSP2050, *Redfish Composability White Paper*, https://www.dmtf.org/dsp/DSP2050

10 Appendix B: Change log

Version	Date	Description
1.0.0	2022-04-07	Initial release.