Document Identifier: DSP2043

Date: 2021-10-27

Version: 2021.3

Redfish Mockups Bundle Readme

Supersedes: 2021.2

Document Class: Informational

Document Status: Published

Document Language: en-US

CONTENTS

1 Foreword

The following files are part of the Redfish development effort:

- DSP0218 - Platform Level Data Model (PLDM) for Redfish Device Enablement Specification - This document specifies the Binary-Encoded JSON (BEJ) and dictionary-based mapping of Redfish schemas and properties into PLDM messages.
- DSP0266 - Redfish Specification - This file is the main Redfish Specification.
- DSP0268 - Redfish Schema Supplement - This reference guide to the Redfish Schema provides normative descriptions and additional text for every schema defined in DSP8010, as well as example payloads for every resource.
- DSP0270 - Redfish Host Interface Specification - This document specifies the "in-band" or "OS-based" Redfish host interface.
- DSP0272 - Redfish Interoperability Profiles Specification - This file specifies the structure and JSON document used to define and publish an interoperability profile used to check an implementation's conformance to a defined minimum set of functionality.
- DSP2043 - Redfish Mockups Bundle - This is a set of mockups that can be used as sample output from GETs from a Redfish service. Informative in nature, it was used to develop the schema. A person can set up an NGINX or similar server and configure it to output JSON format and then use this directory for demonstration purposes.
- DSP2044 - Redfish White Paper - This is intended to be a non-normative document helping those new to Redfish understand how to interact with the Redfish service and understand common functions and tasks.
- DSP2046 - Redfish Resource and Schema Guide - This contains informative documentation regarding common Redfish resource properties, as well as a listing of properties that can be found in each of the Redfish resources.
- DSP8010 - Redfish Schema - This contains the Redfish schema definitions. These files are normative in nature and are normatively referenced by the Redfish Specification. There are three schema formats - CSDL (OData Common Schema Definition Language format, which is in XML), JSON Schema, and OpenAPI schema. These schema definitions should be functionally equivalent, thus specifying the schema in two different languages.
- DSP8011 - Redfish Standard Registries - This contains the Redfish registry definitions. This bundle of Redfish registries includes message registries used for Redfish-defined messages, including events, and privilege maps.
- DSP8013 - Redfish Interoperability Profiles Bundle - A bundle of published Redfish interoperability profile documents as well as supporting schema and sample documents used for creating profiles.

2 Redfish Mockups Bundle contents

This archive contains a number of mockups of various Redfish service implementations. They are intended to be a guide for learning about the Redfish Specification by showing typical examples of implementations. These mockups are not prototypes and do not reflect any actual product or Redfish implementation.

Many of these mockups are also used to populate the Redfish Resource Explorer, part of the Redfish Developer Hub located at: http://redfish.dmtf.org

2.1 Simple Rack-mounted Server (public-rackmount1)

This illustration of a Redfish service implementation shows a typical rack-mount server, as commonly used in scale-out data centers. It depicts the types of information that can be expected, but does not represent an actual implementation.

2.2 Bladed System (public-bladed)

This example represents an enclosure of "blade servers" that share infrastructure components, such as power supplies and fans. Depicting an enclosure containing four blade servers (a total of five "Chassis"), this mockup demonstrates the modeling of multiple chassis and systems managed from a single Redfish service.

2.3 Simple Rack-mounted Server with Local Storage (public-localstorage)

This example shows a server with an implementation of the Redfish storage resources, showing an integrated RAID controller with four attached drives.

2.4 SAS Fabric (public-sasfabric)

This example shows a more complex storage implementation using a pair of SAS switches, storage enclosures and multiple storage devices.

2.5 Proposed OCP Redfish Profile (public-catfish)

This draft example, for ongoing development, represents a proposed minimal Redfish data model "profile" that meets the needs of the Open Compute Project's Hardware Management requirements. This draft profile is intended to help

define a list of required properties so that essential management-related tasks, as defined by OCP, can be performed on any Redfish implementation.

2.6 Composable System via Specific Compositions (public-composability)

This example shows a service with various sets of disaggregated hardware as resources. It provides an example two composed systems utilizing some of the disaggregated hardware. It also shows how Resource Zones can provide information about binding restrictions. It also shows how to express composition requests using the specific composition format.

2.7 Bladed Partitions (public-bladed-partitions)

This example shows how Redfish Composability can be used to create composed Computer System instances from smaller sets of Computer Systems. A top level enclosure called "Enclosure" contains a set of blades, which are used to create the composed Computer Systems.

2.8 Composable System via Constrained Compositions (public-constrained-composition)

This example shows a service with various sets of disaggregated hardware as resources. It provides an example two composed systems utilizing some of the disaggregated hardware. It also shows how Resource Zones can provide information about binding restrictions. It also shows how to express composition requests using the constrained composition format.

2.9 Expansion Enclosure with Resource Blocks (public-expansion-box)

This example shows a service with various sets of disaggregated hardware as resources. The service itself provides information about the types of hardware available in the enclosure, but provides no composability functionality. In these circumstances, an external Redfish service might be used to orchestrate how the equipment is provisioned for composability.

2.10 Telemetry Service (public-telemetry)

This example shows a service that supports reporting telemetry data through the Telemetry Service. It has sample metric definitions and metric reports based upon data found in other portions of the data model.

2.11 Power Distribution Unit (PDU) (public-pdu)

This example shows a set of managed power distribution units (PDU), including a rack-mount unit, a floor (row) PDU, and an automatic transfer switch.

2.12 Sample Service with OEM Extensions (public-oem-examples)

A lightweight Redfish service with OEM examples. The Service Root resource has been extended to have an OEM section. The service container also has an extension. The Account Service resource contains and OEM action. These OEM extensions are defined in the Contoso.com folder of the mockup.

2.13 Mockup for PMEM technologies (public-pmem-configuration)

A mockup of initial configurations for reporting PMEM Devices and then the configurations after a sequence of configuration requests.

2.14 NVMe-oF JBOF (public-nvmeof-jbof)

This mockup contains a sample Redfish service for an NVMe-oF JBOF. The storage resources off of service root contain the provisionable storage for external hosts. The fabric portion of the data model is used to express host connectivity to the different NVMe namespaces. It also contains a single SoC, represented as a computer system, that is used as the front-end for receiving NVMe-oF traffic before the drives are accessed.

2.15 Advanced Communication Devices (public-acd)

This example shows a server with an implementation of the Redfish advanced communication device (ACD) model using the NetworkAdapter, NetworkDeviceFunction, and Port resources.

2.16 Composable System via Compose Action with Manifest (public-compose-action)

This example shows a service with various composable elements. The composition service supports a Compose action, which allows a client to provide a manifest to perform a set of operations to allocate resources and compose systems. It also supports assignment of resource blocks into free and active pools.

2.17 Cables (public-cables)

This example shows an implementation that contains a set of cables and shows their connectivity to other components in the service.

2.18 Power Shelf (public-power-shelf)

This example shows an example power shelf with connections to an electrical bus.

2.19 SmartNICs (public-smartnic)

This example shows a server with two SmartNICs installed. Each SmartNIC is modeled as a Chassis resource. The SmartNIC in slot 1 represents an SoC-based SmartNIC with its own system representation. The SmartNIC in slot 2 represents an FPGA-based SmartNIC. There is also an Ethernet fabric to show address pool configuration for the SmartNICs.