OCP SUMMIT

March 20-21
2018
San Jose, CA

OPEN. FOR BUSINESS.

Redfish Interoperability Profiles

Jeff Autor

Distinguished Technologist, Hewlett Packard Enterprise

Co-Chair, Scalable Platforms Management Forum, DMTF

Redfish Interoperability Profile Goals

- The Redfish specification has a minimal set of “required properties”
  - These are mostly structural properties necessary to support the data model
  - This was done to allow implementation on a wide variety of products
- An Interoperability Profile provides “common ground” for implementation
  - Service implementers know what needs to be included
  - Client software developers know what to expect
    - Profile document can be used by a client to pre-determine available resources and property-level support
  - Users receive consistent behavior among products and vendors
- DMTF will host profile documents for download

Redfish Interoperability Profile Usage

- A profile would apply to a particular category or class of product (e.g. “Front-end web server”, “NAS”, “Enterprise-class database server”)
- It specifies Redfish implementation requirements, but **is not** intended to mandate underlying hardware/software features of a product
- Provides a target for implementers to meet customer requirements
- Provide baseline expectations for client software developers utilizing Redfish
- Enable customers to easily specify Redfish functionality / conformance in RFQs

Redfish Interoperability Profile Ecosystem

- Machine-readable profile definition file
  - JSON document specified by DMTF document DSP0272
  - https://www.dmtf.org/sites/default/files/standards/documents/DSP0272_1.0.0_0.pdf
  - Simple design allows non-programmers to create profiles
  - Schema available (RedfishProfile.v1_0_0.json) for use with JSON schema validators
- Open source Profile Interop Validator
  - Uses Profile document to check an implementation for conformance
  - https://github.com/DMTF/Redfish-Interop-Validator
- Open source documentation generator
  - Combines Profile document with Redfish schemas to produce “user guide”
  - https://github.com/DMTF/Redfish-Tools

Redfish Interoperability Profile – Document Format

Profile Document Format

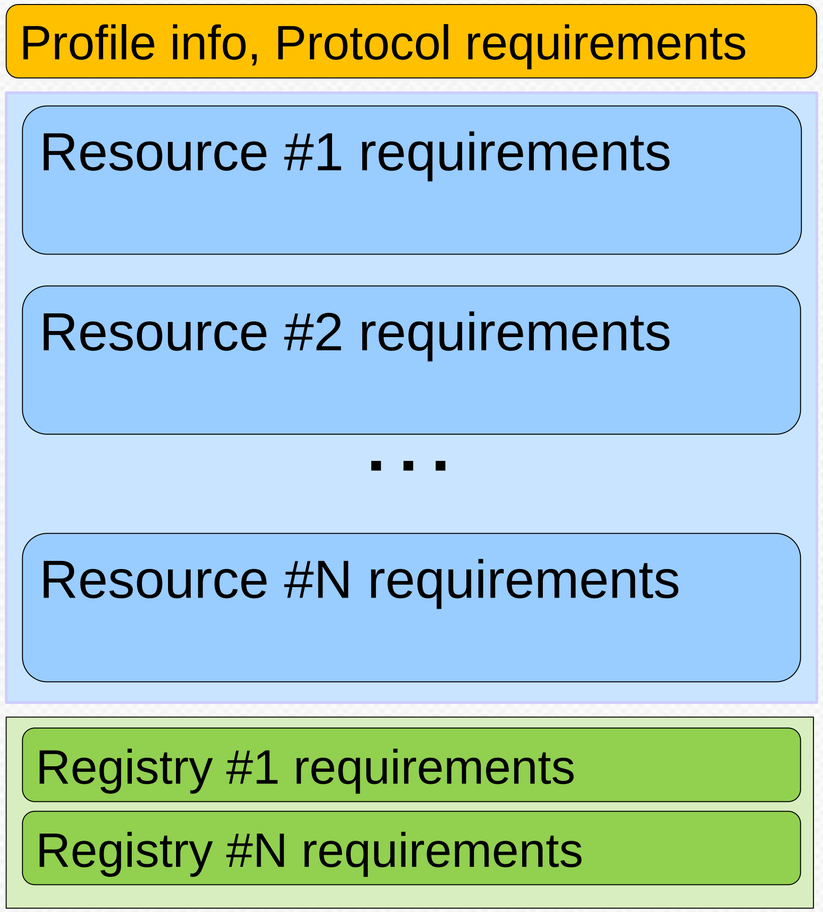
JSON document with simple structure to list resources and properties

- Format allows easy comparison to a retrieved Redfish payload
  - Ex. “PropertyRequirements” object with Redfish properties
- Can build definition on top of other Profile(s)
- Also allows specification of Redfish Protocol features, Resources/Properties, Actions and Registries.

Versioning support in both Profile and Resource requirements

- Profile is a static definition once published
  - Does not increase in scope as schemas are revised
- Recommend that changes to profile occur with “major” revisions
  - Allow for errata, but Profile should be built for longevity
  - Example: “Basic Server v1”, “Basic Server v2”

Profile Document Structure



- Each section a JSON object
- Resource (schema) and Registry objects follow the names of the defining schema
- e.g. “EthernetInterface”
- Property-level requirement nested within Resource requirements, named to follow the defined property name
- e.g. “AssetTag”, “SpeedMbps”

Profile-level information and Protocol Requirements

```
“SchemaDefinition”: “RedfishInteroperabilityProfile.v1_0_0”,
"ProfileName": "Anchovy",
“ProfileVersion": "1.0.2",
“OwningEntity": "Pizza Box Project",
"Purpose": "This is a sample Redfish profile.",
"ContactInfo": "pizza@contoso.com",
"RequiredProfiles": {
   "DMTFBasic": {
      "MinVersion": "1.0.0",
   },
   “ContosoPizza”: {
      “Repository”: “http://contoso.com/profiles”,
      “MinVersion”: “1.0.0”
   }
 },
“ProtocolRequirements”: {
   “MinVersion”: “1.0.0”,
   “DiscoveryRequired”: false
},
```

- Basic information
- Name, version, author, etc.

- Ability to include other Profiles to build upon past work
- But profile cannot loosen requirements included from other profiles, only add additional requirements

- “Protocol requirements” are Redfish features which are not part of the JSON response payload(s).

Resource (schema) level requirements

```
"ContosoTimeMachine": {
   “Repository": “http://www.contoso.com/schemas",
   “ReadRequirement”: “Mandatory”,
   "MinVersion": "1.2.0",
   "PropertyRequirements": {
      "CurrentTime": {},
      "DestinationTime": {},
      "IsGrandfatherAlive": {
         “ReadRequirement": "Recommended"
      },
      "ParadoxDetected": {
         “ReadRequirement": "IfImplemented"
      }
    }
```

- Organized by schema name
- Profile can include requirements from any number of standard schemas
- Resource level “ReadRequirement” sets need for schema-required properties
- Property level requirements contained in resource-level object
- “MinVersion” – minimum schema version required

Property level - basic features

```
"ComputerSystem": {
   "MinVersion": "1.1.0",
   "PropertyRequirements": {
      “SystemType”: {
         “Values”: [“Physical”],
         “ReadRequirement”: “Mandatory”
      },
      “AssetTag": {
         “ReadRequirement": “Mandatory“,
         “WriteRequirement”: “Mandatory”
      },
      "Manufacturer": {},
      "Model": {
         “ReadRequirement": "Recommended"
      },
 . . .
```

JSON objects follow property names

- Un-listed properties have no requirements
- Empty objects are by default ‘Mandatory’

“ReadRequirement”:

- Default value is ‘Mandatory’
- Recommended, If-Implemented, and Conditional support

“MinCount”:

- Minimum count of non-NULL items in array

“WriteRequirement”:

- If property must support PATCH or PUT

“Values”:

- Require specific or “any of these” values for a property. Also supports arrays

Property level – Conditional requirements

```
"EthernetInterface": {
   "PropertyRequirements": {
      "MACAddress": {},
      “HostName": {
         “ReadRequirement": "Recommended",
         “ConditionalRequirements": [{
              "SubordinateToResource":
                 ["ComputerSystem“,
                  ”EthernetInterfaceCollection”],
              “ReadRequirement": “Mandatory"
          }]
      },
      "IPv4Addresses": {
         “ReadRequirement": “Mandatory",
         "MinCount": 1,
         “ConditionalRequirements": [{
             "SubordinateToResource":
               ["ComputerSystem“,
                ”EthernetInterfaceCollection”],
              “ReadRequirement": “Mandatory“,
              "MinCount": 2
         }]
}
```

- “ConditionalRequirements”: Apply to the property if one or more conditions are met
- “Purpose”: Text provides justification for the conditional requirement
- “SubordinateToResource”: If resource matches the parent hierarchy, requirement applies
- Comparison Property / Values: Using another property within the resource as key, add requirement if value of the key matches a list

Property level – ‘Conditional’ Value example

```
"IndicatorLED": {
   “ReadRequirement": "Recommended",
   “WriteRequirement”: “Recommended”,
   "ConditionalRequirements": [{
      "Purpose": "Physical and composed Systems
must have a writable Indicator LED“,
      “ReadRequirement": “Mandatory",
      “WriteRequirement”: “Mandatory”,
      “Comparison": "AnyOf",
      “CompareProperty": "SystemType",
      “CompareValues": ["Physical", "Composed“]
   }]
}
```

- ‘Comparison’ provides test
- ‘CompareProperty’ name
- May be at current object level or in parent objects (no peers)
- ‘CompareValues’ – one or more values to test against
- Requirement – applies if condition met
- ‘ConditionalRequirements’ is an array, allowing multiple conditions for a given property

Action level features

```
“ActionRequirements": {
   "Reset": {
      “ReadRequirement": “Mandatory",
      "Parameters": {
         "ResetType": {
         “MinSupportedValues": ["ForceOff",
                                "PowerCycle"]
         }
      }
   }
}
```

- Organized by Action name within each Resource (schema)
- Allows for parameter requirements
- AllowableValues support

Registry level features

```
"Registries": {
   "Base": {
      "MinVersion": "1.0.0",
      “Repository": “http://redfish.dmtf.org/registries",
      "Messages": {
         "Success": {},
         "GeneralError": {},
         "Created": {},
         "PropertyDuplicate": {}
      }
   },
   "ContosoPizzaMessages": {
      “Repository": “http://contoso.com/registries",
      "ReadRequirement": “Mandatory"
   }
}
```

- Organized by registry name
- Allows for multiple registries
- Ability to include OEM registries
- Resource level “ReadRequirement” sets need for full Registry requirement
- Messages listed with individual ‘Requirement’ as needed

OCP SUMMIT