Redfish Interoperability Profiles

DMTF Scalable Platforms Management Forum
DSP2072 v1.0 (December 2017)
Redfish Interoperability Profiles

- Provide common ground for implementers, software developers, and users
  - A profile applies to a particular category or class of product (e.g. “Front-end web server”, “Enterprise-class database server”, “Rack-level PDU”)
  - 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
  - Provides baseline expectations for client software utilizing Redfish
  - Enables customers to easily specify desired Redfish functionality in RFQs

- Profile published as a machine-readable JSON document
  - Document must also be human-readable
  - Can be created by dev/ops personnel and non-CS professionals

- Profiles can be authored by DMTF, partner organizations, and others
- Open source tools available to produce documentation and test conformance

www.dmtf.org
Redfish Interoperability Profile Ecosystem

- Profile document specification (DSP0272)
  - JSON document specified by DMTF document DSP0272
  - https://www.dmtf.org/sites/default/files/standards/documents/DSP0272_1.0.0_0.pdf

- Profile bundle (DSP8013)
  - Includes schema (RedfishProfile.v1_0_0.json) for Profile document
  - Includes sample profile
  - Future releases will include DMTF (re)-published Profiles

- Profile Interop Validator – open source tool
  - Uses Profile document to check an implementation for conformance
  - https://github.com/DMTF/Redfish-Interop-Validator

- Documentation Generator – open source tool
  - Combines Profile document with Redfish schemas to produce “user guide”
  - https://github.com/DMTF/Redfish-Tools
Redfish Interoperability Profiles

JSON DOCUMENT FORMAT
Redfish Interoperability Profile Document

- 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)
  - Apply requirements to Redfish Protocol features, Resources (Schemas), 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"
  }
},
"Protocol": {
  "MinVersion": "1.0.0",
  "Discovery": "Mandatory",
  "HostInterface": "Recommended"
},

• 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",
  "UpdateResource": true,
  "PropertyRequirements": {
    "CurrentTime": {},
    "DestinationTime": {},
    "IsGrandfatherAlive": {
      "ReadRequirement": "Recommended"
    },
    "ParadoxDetected": {
      "ReadRequirement": "IfImplemented"
    }
  }
}

- Organized by schema name
- Profile can include requirements from any number of standard or OEM-defined schemas
- Resource level “ReadRequirement” sets need for schema-required properties
- Create/Update/Delete support can be required.
- Property level requirements contained in resource-level object
- “MinVersion” – minimum schema version required
Property level - basic features

- 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
- “WriteRequirement”:
  - If property must support PATCH or PUT
- “MinCount”:
  - Minimum count of non-NULL items in array
- “Values”:
  - Require specific or “any of” 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": {
                "ParameterValues": ["ForceOff", "PowerCycle"],
                "RecommendedValues": ["On", "Graceful Shutdown"]
            }
        }
    }
}
```

- Organized by Action name within each Resource (schema)
- Allows for parameter requirements
  - Required values
  - Recommended values
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
Q&A & Discussion

www.dmtf.org