Disclaimer

• The information in this presentation represents a snapshot of work in progress within the DMTF.
• This information is subject to change without notice. The standard specifications remain the normative reference for all information.
• For additional information, see the Distributed Management Task Force (DMTF) website.
Goals

- An “Interoperability Profile” provides a common ground for Service implementers, client software developers, and users
  - 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
- Create a machine-readable Profile definition
  - Document must be human-readable
  - Can be created by dev/ops personnel and non-CS professionals
- Enable authoring of Profiles by DMTF, partner organizations, and others
- Create open source tools to document and test conformance
Implementation

• Redfish Interoperability Profile is a Machine-readable JSON document
  • Schema-backed (RedfishProfile.v1_0_0) JSON definition
  • This file will be read by conformance and documentation tools
• DMTF specification (DSP0272) provides instructions to create a profile
• Creating open source tools for conformance testing
  • Leverages existing Redfish Conformance tools and applies profile requirements
• Creating a tool for generating profile documentation
  • Documentation generator produces profile-specific schema/property view
  • Uses a combination of the JSON profile document and a Markdown ‘supplement’
  • Supplemental text provides context and clarification on the Profile’s purpose
  • Tool can produce ‘for review’ output that shows schemas and properties that are not included (no requirements) in the profile definition
Profile Document Functionality

• Required resources (schema), objects, or properties
  • Simple requirements apply to every instance of the Resource
  • Conditional requirements make additions for specific cases

• “If Implemented” resources, object, or properties
  • Must appear if underlying feature is implemented in the product
    • Example: Fan[] array required in Chassis that have fans…
  • “If Implemented” conformance usually not testable by automated tools

• Conditional Requirements
  • Items required under certain circumstances or for sub-classes of products
  • Based on values of adjacent properties or location in the resource tree
    • Example: EthernetInterface resource required under each ‘Manager’

• Registry Requirements
  • Support for standard messages for errors and events
Redfish Interoperability Profiles

JSON DOCUMENT FORMAT
Redish 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

"ProfileName": "Anchovy",
"Version": "1.0.2",
"Author": "Pizza Box Project",
"Purpose": "This is a sample Redfish profile.",
"ContactInfo": "pizza@contoso.com",
"RequiredProfiles": {
  "DMTFBasic": {
    "MinVersion": "1.0.0",
  },
  "ContosoPizza": {
    "OwningEntity": "Other",
    "OwningEntityName": "Contoso",
    "Source": "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

- 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
- Property level requirements contained in resource-level object
- “MinVersion” – minimum schema version required
Property level - basic features

"ComputerSystemCollection": {
    "PropertyRequirements": {
        "Members": {
            "MinCount": 1
        }
    }
},
"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” 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",
  "Conditions": [{
    "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",
    "Source": "redfish.dmtf.org/registries",
    "Messages": {
      "Success": {},
      "GeneralError": {},
      "Created": {},
      "PropertyDuplicate": {}
    }
  },
  "ContosoPizzaMessages": {
    "OwningEntity": "Other",
    "OwningEntityName": "Contoso",
    "Repository": "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

Redfish

www.dmtf.org