Open. Together.
Tools and Process for Creating a Redfish Profile

Jeff Autor
Distinguished Technologist
Hewlett Packard Enterprise
Redfish Profiles and OCP

- Standard format for specifying Redfish support
- Easy method to communicate level of implementation
  - Redfish has independent versions for each schema
  - Common question “What version of Redfish do you support?” is not meaningful as this doesn’t equate to implementation
- HW Management project has produced a “Baseline” profile
- Server project is extending that with a “Server” profile
- Other projects can extend Baseline for their equipment
Redfish Profile Document

Simple JSON document structure

• Based on resource and property requirements
• 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.
Profile Versioning

Versioning support in both Profile and Resource requirements

- Profile is a static definition once published, and does not increase in scope as schemas add functionality
- Recommend “new versions” of Profiles just use a new name
- Schema (resource) Versions are “<major>.<minor>.<errata>”
  - Major version: Client and Service must match major version
  - Minor version: New, backwards-compatible functionality added
  - Errata: Bug fixes or clarifications (no functional change)
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

- 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)
- For OCP profiles, these are covered by Baseline – and are included here just as an example…

```json
{
"SchemaDefinition": "RedfishInteroperabilityProfile.v1_0_0",
"ProfileName": "OCPServerHardwareManagement",
"ProfileVersion": "0.2.3",
"Purpose": "Specifies the OCP management requirements for the Redfish interface on OCP Server platforms",
"OwningEntity": "Open Compute Project",
"ContactInfo": "redfish@opencompute.org",
"RequiredProfiles": {
   "OCPBaselineHardwareManagement": {
      "MinVersion": "1.0.0"
   }
},
"Protocol": {
   "MinVersion": "1.0",
   "Discovery": "Recommended",
   "HostInterface": "Recommended"
}
}
Resource (schema) level requirements

- 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

"Chassis": {
  "MinVersion": "1.0.0",
  "PropertyRequirements": {
    "AssetTag": {
      "ReadRequirement": "Recommended",
      "WriteRequirement": "Recommended"
    },
    "ChassisType": {},
    "IndicatorLED": {
      "ReadRequirement": "Recommended",
      "WriteRequirement": "Recommended"
    },
    "Manufacturer": {},
    "Model": {},
    "SerialNumber": {},
    "SKU": {
      "ReadRequirement": "Recommended"
    },
    ...
}
"PropertyRequirements": {
  "Temperatures": {
    "ReadRequirement": "Mandatory",
    "MinCount": 3,
    "PropertyRequirements": {
      "ReadingCelsius": {},
      "PhysicalContext": {
        "Comparison": "AllOf",
        "Values": ["CPU", "Intake", "SystemBoard"]
      },
      "UpperThresholdFatal": {
        "ReadRequirement": "Recommended"
      },
      "UpperThresholdCritical": {
        "ReadRequirement": "Recommended"
      }
    }
  }
}
Property-level Conditional Requirements

- "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

```
"EthernetInterface": {
  "PropertyRequirements": {
    "HostName": {
      "ReadRequirement": "Recommended",
      "ConditionalRequirements": [{
        "SubordinateToResource": [
          "ComputerSystem",
          "EthernetInterfaceCollection"
        ],
        "ReadRequirement": "Mandatory"
      }]
    },
    "IPv4Addresses": {
      "ReadRequirement": "Mandatory",
      "MinCount": 1,
      "ConditionalRequirements": [{
        "SubordinateToResource": [
          "ComputerSystem",
          "EthernetInterfaceCollection"
        ],
        "ReadRequirement": "Mandatory",
        "MinCount": 2
      }]
    }
  }
}
```
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
Message Registry-level features

- 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
- Note that OCP Profiles currently have no Registry requirements
Profile specifications

- Download from DMTF site: http://www.dmtf.org/standards/redfish
- Redfish Interoperability Specification (DMTF document DSP0272)
  - Defines the JSON document structure used in this presentation
- Profile Bundle (DMTF document DSP8013)
  - Includes a JSON Schema document useful for validating a Profile
  - Sample profile document showing examples of the profile structure and features
  - Future releases of this bundle will contain published Profiles
Profile tools

- Use a JSON schema validator to verify your profile document
  - Ensures document meets format defined by the specification
  - Such as: https://www.jsonschemavalidator.net/

- Profile Interop Validator
  - Open source tool to check a Redfish Service implementation against the Profile
  - https://github.com/DMTF/Redfish-Interop-Validator

- Documentation generator
  - Open source tool combines Profile document with Redfish schemas
  - Will automatically produce a “user guide” specific to your Profile
  - https://github.com/DMTF/Redfish-Tools
Call to Action

Create profile(s) for your OCP project
- Build from the OCP Baseline profile
- Get OCP Incubation Committee approval for the profile

Publish your profile document (and user guide)
- On OCP project page
- Submit for re-publication via DMTF’s profile repository
  - Published profiles included in DSP8013 bundle

Direct implementations to test conformance using the OCP Conformance Test Suite
Open. Together.
OCP Global Summit | March 14-15, 2019