Redfish Control schema and Sensor enhancement proposal

DMTF Redfish Forum

August 2020

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.

Getting involved in Redfish

- Redfish Standards page
  - Schemas, Specs, Mockups, White Papers & more
  - http://www.dmtf.org/standards/redfish
- Redfish Developer Portal
  - Redfish Interactive Resource Explorer
  - Educational material, documentation & other links
  - http://redfish.dmtf.org
- Redfish User Forum
  - User forum for questions, suggestions and discussion
  - http://www.redfishforum.com
- DMTF Feedback Portal
  - Provide feedback or submit proposals for Redfish standards
  - https://www.dmtf.org/standards/feedback
- DMTF Redfish Forum
  - Join the DMTF to get involved in future work
  - http://www.dmtf.org/standards/spmf

Introduction

- General need to represent user or system-owned control points in a system
- These are typically coupled with one or more sensors that provide readings and feedback into a control loop
- Like a sensor, there is desire to display a small, primary set of data about the control, while having the ability to retrieve a large set of detailed, mostly static information about the control

Definition – “Managed Element”

- Generic term for the device or system being managed
  - Examples: ComputerSystem, PowerDistributionUnit, etc.
- A Redfish resource for a managed element will have many properties
  - General information about the element
    - PartNumber, SerialNumber, etc.
    - Status – State, Health, HealthRollup
  - Links to additional (subordinate) resources that describe subsystems
  - May have one or more controls
  - May have one or more associated sensors
- Desire to provide summary of control settings and sensors readings
  - Redfish models attempt to provide this with a single GET
  - Define schemas “excerpt” of the Control and Sensor resources
    - These provide links to Sensor and Control resources to obtain details

CONTROL SCHEMA

Control schema proposal

- New schema heavily leveraged from Sensor
- Describes an individual control point plus associated sensor(s) that measures the effects of that control point
  - More formally known as an “effector”
- *SetPoint* is the primary property for a Control
  - The desired value for a Reading resulting from the Control setting
- *ControlLoop* object to expose and control parameters of a control loop
- Includes a “pass-through” of a single *Reading* from a Sensor
  - Sensor *Reading* might use different measurement units than the *SetPoint*
    - Example: Liquid flow valve (percent open), with sensor Reading in liters/second
  - Controls with multiple associated sensors can build Sensor excerpts in context for more complex relationships

Control Definition (example w/single associated Sensor)

```
{
    "@odata.type": "Control.v1_0_0.Control",
    "@odata.id": "/redfish/v1/Chassis/1U/Controls/Thermostat",
    "Id": "Thermostat",
    "ControlType": "Temperature",
    "ControlUnits": "Cel",
    "SetPoint": 27,
    “ReadingDeadBand": 1.5,
    “ReadingUnits”: “Cel”,
    "OperatingMode": "Automatic",
    “Sensor”: {
       "Reading": 27,
       “DataSourceUri”: “/redfish/v1/Chassis/1U/Sensors/CabinetTemp”
    },
    “Algorithm”: “Average”,
    “MinValue”: 10,
    “MaxValue”: 35,
    “Precision”: 1,
    “Accuracy”: .25,
    “ControlLoop”: {
       “Proportional”: 0.232,
       “Integral”: 0.095,
       “Derivative”: 0.049
     },
    "Oem": {}
}
```

SetPoint is a primary property

OperatingMode allows for “Manual” or “Automatic” operation – definition will depend on the type of Control

Reading from a Sensor instance (a Sensor excerpt)

Algorithm would allow model to expose how the control loop operates

ControlLoop would allow model to expose parameters for a PID-based system, or even allow modification

Control Definition (multiple associated Sensors example)

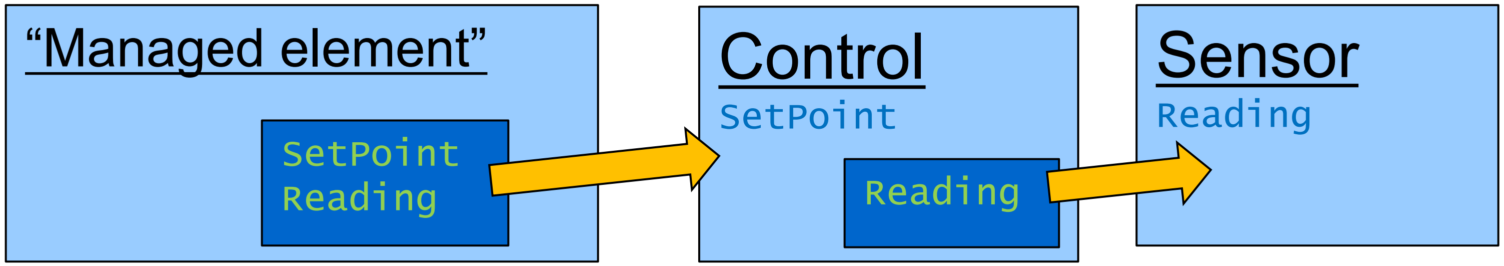
```
{
    "@odata.type": "Control.v1_0_0.Control",
    "@odata.id": "/redfish/v1/Chassis/1U/Controls/Thermostat",
    "Id": "Thermostat",
    "ControlType": "Temperature",
    "ControlUnits": "Cel",
    "SetPoint": 27,
    “SetPointDeadBand": 1.5,
    “ControlDelaySeconds”: 180,
    "OperatingMode": "Automatic",
    "ReadingType": "Temperature",
    “MinValue”: 10,
    “MaxValue”: 35,
    “Precision”: 1,
    “Accuracy”: .25,
    “Links”: {
        “AssociatedSensors": [
                { "@odata.id": "/redfish/v1/Chassis/1U/Sensors/CabinetTemp“ },
                { "@odata.id": "/redfish/v1/Chassis/1U/Sensors/CabinetIntake“ },
                { “@odata.id”: “/redfish/v1/Chassis/1U/Sensors/CabinetExhaust” }
        ]
    },
    "Oem": {}
}
```

Array of links to multiple Sensor instances instead of a single Sensor excerpt

Control usage of schema excerpts

- An excerpt inserts a copy of certain properties from another resource
  - Allows high-value properties to appear in managed element resources
    - Enables consistent definitions for Sensors and Controls across the data model
  - Also provides a link to the original source of the data
  - Excerpted properties are defined using annotations in Redfish schema
- Control can include a Sensor excerpt
  - Provides *Reading* from a single associated Sensor
  - This property can be further reflected into a managed element resource
- Unit-specific *Reading<units>* are added if a Control need is identified
  - Useful for Control instances when the associated Sensor has different units
  - These would appear in managed element resources as excerpts of Control
  - Examples
    - *ReadingPercent* – Fan speed where input is a PWM value
    - *ReadingGPM* – Liquid flow where input is a valve percent open

Sensor and Control excerpt flow



- Sensor resource
  - *Reading* is defined as an excerpt property
- Control resource
  - Sensor excerpt appears as an object, with *Reading* included
  - *SetPoint* is defined as an excerpt property
  - *DataSourceUri* points to the associated Sensor that provided *Reading*
- "Managed element" resources
  - Control excerpts appears as objects, with SetPoint included
    - Also includes Reading from the Control's Sensor excerpt as a "pass-through"
    - Reading might be rendered as *Reading*<units> to match the Control needs
  - *DataSourceUri* points to the associated Control

Control excerpt examples

- An excerpt appears in a managed element resource as an object containing the excerpted properties and the link to source

```
"ThermostatCelsius": {
   "Reading": 27,
   "SetPoint": 27,
   "DataSourceUri": "/redfish/v1/Chassis/1U/Controls/Thermostat"
},

"WaterValvePercent": {
   "ReadingGPM": 7.93,
   "SetPoint": 70,
   "DataSourceUri": "/redfish/v1/Chassis/1U/Controls/Valve1"
},

"WaterValvePercent": {
   "Reading": 7.93,
   “ReadingUnits”: “GPM”,
   "SetPoint": 70,
   "DataSourceUri": "/redfish/v1/Chas
},
```

Example #1:*SetPoint* has the same units as *Reading* from a single Sensor instance

Example #2: *SetPoint* and *Reading* use different units. Excerpt includes *Reading* property from Sensor renamed as *Reading<unit>* to clarify the difference in units.

DataSourceUri points to the Control resource. The Control resource will contain a Sensor excerpt with *DataSourceUri* further pointing to the Sensor resource.

Example #3: Alternate proposal of #2 using *Reading* and *ReadingUnits* when different units of measure are needed

Complex Control example

- Multiple excerpts appear in this managed element resource as objects containing the excerpted properties and links to each data source
- This provides a compact summary of the element in a single resource

```
{
“PrimaryControl”: {
   “ComplexControlPercent": {
       "SetPoint": 75,
       "DataSourceUri": "/redfish/v1/Chassis/1U/Controls/PrimaryValve"
   },
   “LiquidFlowGPM": {
       "Reading": 7.93,
       "DataSourceUri": "/redfish/v1/Chassis/1U/Sensors/PrimaryValveFlow"
   },
   “InputPressurePsi”: {
       "Reading": 35.81,
       "DataSourceUri": "/redfish/v1/Chassis/1U/Sensors/IntakePressure"
   },
   “OutputPressurePsi": {
       "Reading": 11.24,
       "DataSourceUri": "/redfish/v1/Chassis/1U/Sensors/OutflowPressure"
   }
}
```

This Control has multiple Sensor associations

Control excerpt contains the *SetPoint*

Note that object names include Reading or SetPoint units

Three Sensor excerpts show the resulting values as Reading

Fan resource concept

- “Fan” is an example of a managed element resource
- Contains a Sensor excerpt, and may expose a Control as well

```
{
    “SpeedControlPWM": {
        “SetPoint": 125,
        "DataSourceUri": "/redfish/v1/Chassis/1U/Controls/FanBay1"
    },
    “SpeedPercent”: {
        “Reading”: 55,
        “SpeedRPM”: 2300,
        “DataSourceUri”: “/redfish/v1/Chassis/1U/Sensors/FanBay1”
    }
}
```

Control excerpt to show SetPoint, if supported

Sensor excerpt to show Reading and “extra fan excerpt property” SpeedRPM

Fan instances that expose the control include the *SpeedControlPWM* object (excerpt of Control), while fans without the control exposed include only the *SpeedPercent object (excerpt of Sensor)*

SENSOR ENHANCEMENTS

User-defined Threshold

- Add User-defined Threshold support in Sensor
  - Clearly define service vs user-defined thresholds
  - Implementation may support user-defined thresholds for each sensor
  - Some existing *Thresholds* usage may move to *UserThresholds*
    - If user can define reaction behavior (but perhaps not change value)?
- Follow *Thresholds* structure with a parallel object definition
  - *UserThresholds {}*
    - *UpperCaution, UpperCritical, LowerCaution, LowerCritical*, etc.
- Or, add new sub-objects to *Thresholds* object with “user” naming?
  - *UpperCautionUser, UpperCriticalUser, LowerCautionUser, LowerCriticalUser*

Control links

- Add Controls links to show relationships
  - “What control(s) do I use to affect changes to this reading?”
  - Provide array of Links to locate Controls that affect the sensor

```
“Links": {
    “Controls”: [{
       { “@odata.id”: “/redfish/v1/Chassis/1U/Controls/PrimaryValve“ },
       { “@odata.id”: “/redfish/v1/Chassis/1U/Controls/SecondaryValve“ }
    ]
},
```

Reactions for Thresholds

- Add *Reaction* object under each *Threshold* to describe actions taken when a threshold is violated
  - Allows service to expose those actions
  - Provide user the ability to define what occurs when a threshold is violated
    - User could be given option to disable the reaction, for example, choosing not to perform a graceful shutdown due to high temperatures
- *Reaction* options include:
  - Log a Message
  - Send an Event
    - Event messages can be used to notify aggregation points or higher-level control systems that they need to take further action
  - Trigger a Metric Report
  - Execute a Job
    - Specify the Job name to execute – allows re-use in multiple sensors
    - Could pre-define frequently-used reactions for ease of use (Shutdown, etc.)
    - Add Job support to execute a Job on receipt of an Event Message

Threshold Reaction

```
{
    "@odata.type": “Sensor.v1_3_0.Sensor",
    "@odata.id": "/redfish/v1/Chassis/1U/Sensors/RoomTemp",
    "Id": “RoomTemp",
    "ReadingType": "Temperature",
    "Reading": 27,
    "ReadingUnits": "Cel",
    "Thresholds": {
        "Caution": {
            "Activation": "Increasing",
            "DwellTime": "PT5M",
            “ThresholdEnabled”: “Mandatory | Enabled | Disabled”,
            "Reading": 30,
            "Reaction": {
                “Description”: “Set fans to 100%”,
                “TriggerMetricReport”: true,
                “GenerateEvent”: “Thermal.CautionTempHigh",
                “ExecuteJob”: “FanFullSpeed”
            }
        }
    },
    "Oem": {}
}
```

Q&A & Discussion