Disclaimer

- The information in this presentation represents a Work In Progress publication of the DMTF.
    - DMTF WIP Document - DSP-IS0005_0.2a
- 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.
    - http://www.dmtf.org/standards/redfish

Redfish Data Center
Power and Cooling Equipment Modeling

Jon Hass

DMTF – Board of Directors Chair

Dell – Distinguished Engineer – Office of CTO

The Distributed Management Task Force

- An Industry Standards Organization
  - Developing manageability standards for 24 years (est. 1992)
  - Membership includes 65 companies and industry organizations
  - With active chapters in China and Japan
- Allied with
  - 14 standard development organizations (alliance partners)
  - 80+ universities and research organizations (academic alliance partners)
- Focused on manageability standards
  - For the management of on-platform, off-platform, network services and infrastructure domains
  - Which are recognized nationally (ANSI/US) and internationally (ISO)

Agenda

- Overview – Redfish APIs for Data Center Equipment
- Power and Cooling Domains
- Equipment modeling patterns
- Inventory information modeling
- Location modeling
- Sensor modeling
- Setpoint and Deadband modeling
- Alarm modeling

Redfish APIs for Power and Cooling Equipment

Objectives:

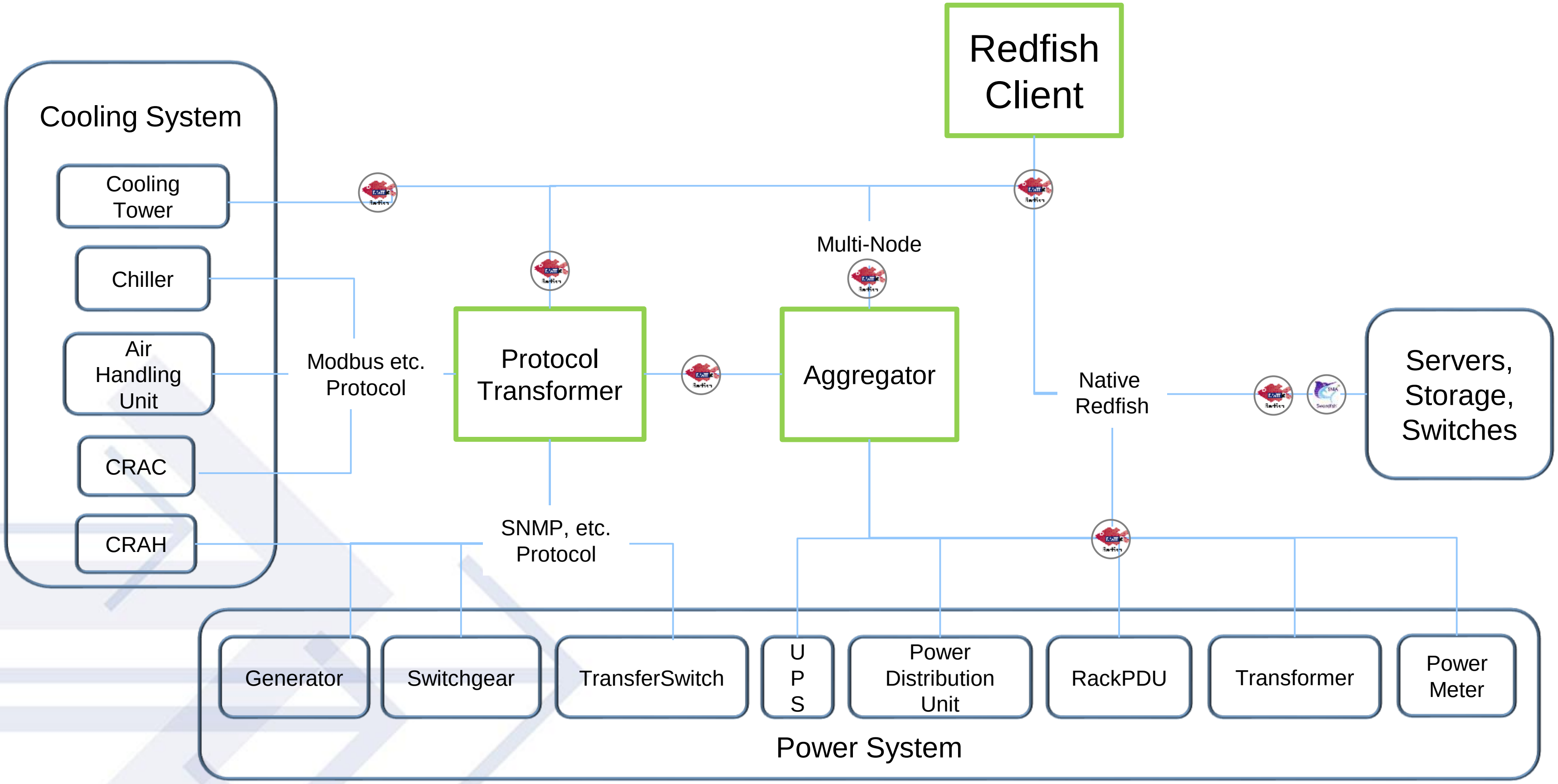
- Provide a public standards based equipment instrumentation interface definition for inventory, monitoring, configuration and lifecycle maintenance for data center power and cooling equipment.
- Establish a management interface for data center facilities equipment that uses the same protocol, security, scalability and is interoperable with IT equipment management interfaces.

Equipment Targeted:

Environmental Sensors
Cooling Tower
Chiller
Air Handling Unit
CRAH
CRAC
Pump
Fan

Power Meter
Generator
UPS
Power Distribution Unit
RackPDU
Transformer
Rectifier
Transfer Switch
SwitchGear

Redfish API Ecosystem – Infrastructure Examples



Power and Cooling Domain Modeling

Within a data center there may be one or more power or cooling domains with each domain consisting of the equipment providing service to the portion of the data center designated as a domain.

The Redfish DCIM modeling approach establishes a Power Domain collection and a Cooling Domain collection subordinate to the Redfish Service Root.

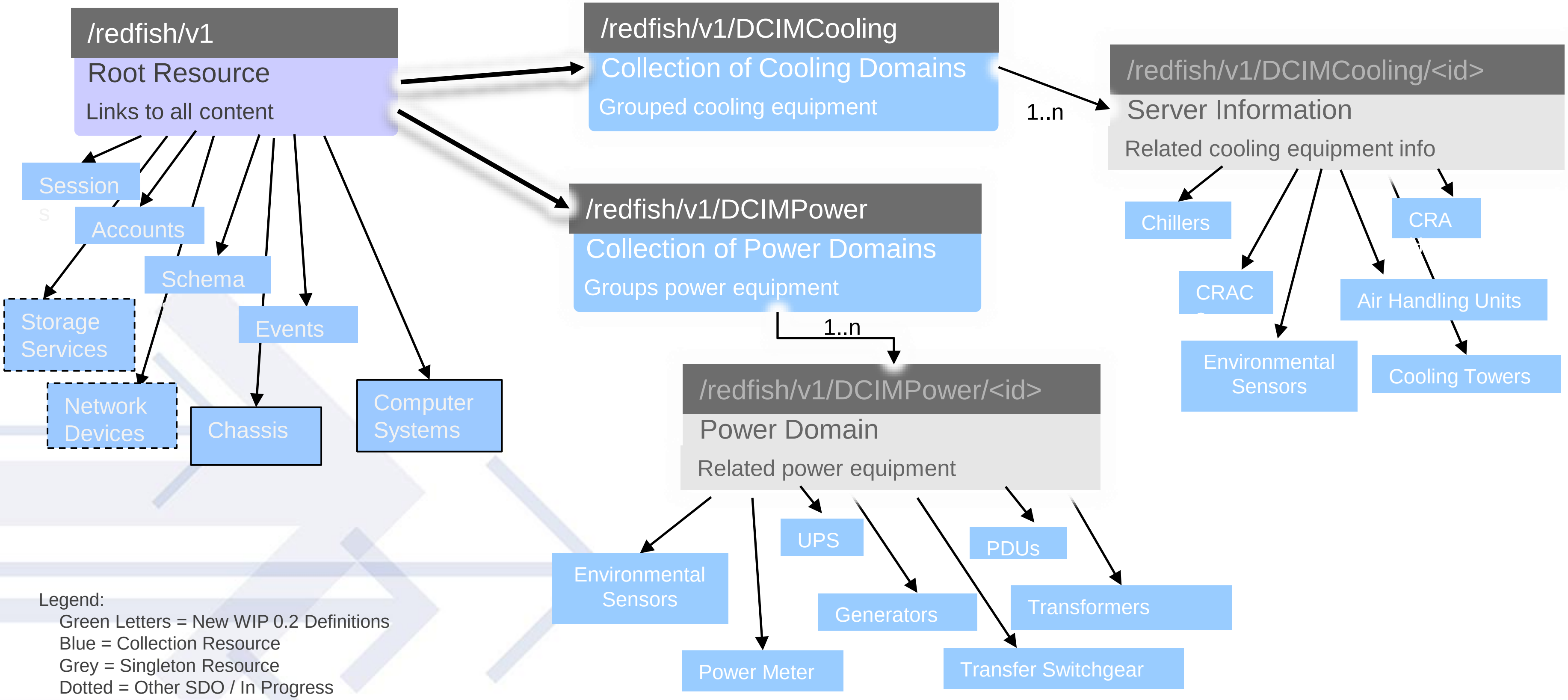
Domains

- Domain Collections located in Service Root and referenced from Service Root document.
- Organize data center facility equipment into 2 domains
  - Power – Name proposal “DCIMPower”
  - Cooling – Name proposal “DCIMCooling”
- Each Domain Collection consists of 1 or more members, with a domain member representing the group of equipment in the domain being instrumented and information/metrics at the domain level.
- A domain member consists of a document with references to collections of specific equipment types and properties representing information and domain level metrics.

Specific Equipment Collections

- An equipment collection consists 0 or more members, with an equipment collection member representing a specific piece of equipment.

Power & Cooling “Domain” URI Namespaces



/redfish/v1
Root Resource
Links to all content

Session

Accounts

Schema

Storage Services

Events

Network Devices

Chassis

Computer Systems

/redfish/v1/DCIMCooling
Collection of Cooling Domains
Grouped cooling equipment

1..n

/redfish/v1/DCIMCooling/<id>
Server Information
Related cooling equipment info

Chillers

CRA

CRAC

Air Handling Units

Environmental Sensors

Cooling Towers

/redfish/v1/DCIMPower
Collection of Power Domains
Groups power equipment

1..n

/redfish/v1/DCIMPower/<id>
Power Domain
Related power equipment

UPS

PDUs

Environmental Sensors

Generators

Transformers

Power Meter

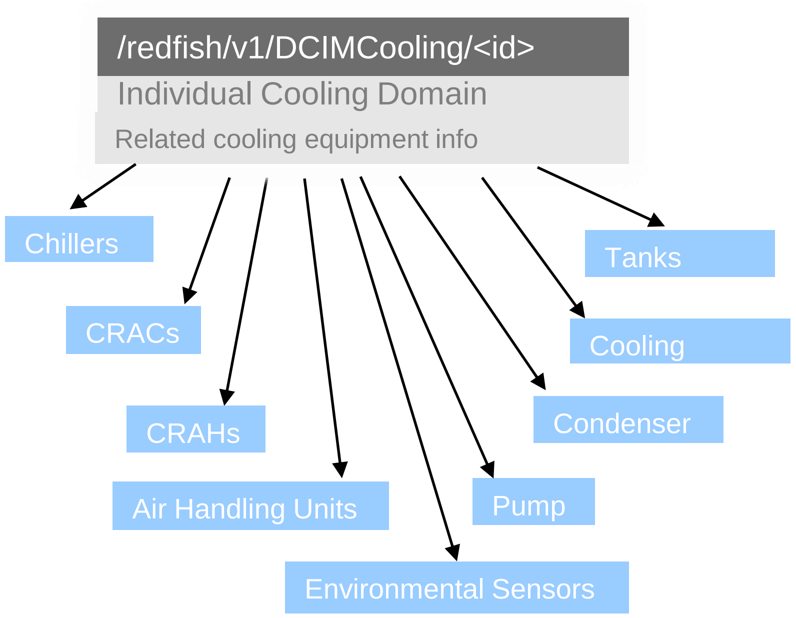
Transfer Switchgear

Legend:
Green Letters = New WIP 0.2 Definitions
Blue = Collection Resource
Grey = Singleton Resource
Dotted = Other SDO / In Progress

Cooling “Domain” Model Concept Example

Example of Cooling Domain JSON document organization

```
{
   “Name” : “CoolingDomain1”,

   “Condensers” : “<collection ref URI>”,
   “Chillers” : “<collection ref URI>”,
   “Pumps” : “<collection ref URI>”,
   “Towers” : “<collection ref URI>”,
   “Tanks” : “<collection ref URI>”,
    “CRACs” : “<collection ref URI>”,
    “CRAHs” : “<collection ref URI>”,
    “AirHandlingUnit” : ”<collection ref URI>”,

   “Sensors” : “<collection ref URI>”,

   “Status” : {
                    “State” : “Enabled”
                    “Health” : “OK”  },

   “Links” : {
                    “ContainedSystems” = [ “<ref>”, “<ref>”]
                    }

   “Actions” : “<actions that affect entire domain>”

    <Other Domain properties>
}
```


/redfish/v1/DCIMCooling/<id>
Individual Cooling Domain
Related cooling equipment info
Chillers
CRACs
CRAHs
Air Handling Units
Environmental Sensors
Pump
Condenser
Cooling
Tanks

Legend:
 Blue = Collection Resource
 Grey = Singleton Resource

Power and Cooling Equipment Modeling Pattern

The following data definition design pattern applies to each data center facilities equipment type:

a) Represented as a CSDL/JSON entity in schema - named after the equipment type.
b) Top level properties.
- **Equipment inventory information**
- **Status**
- **Metrics**
- **Unique equipment properties and actions**

c) One Sensor Collection for all equipment and sub-component Sensors
- **Represented in one collection referenced as top level navigation property**

d) Specific equipment top level Sensors
- **Represented in SensorType arrays, an array for each SensorType**

e) SetPoints and Deadbands
- **Represented in SetPoint array of references to Sensor collection members with SetPoint and Deadband properties**

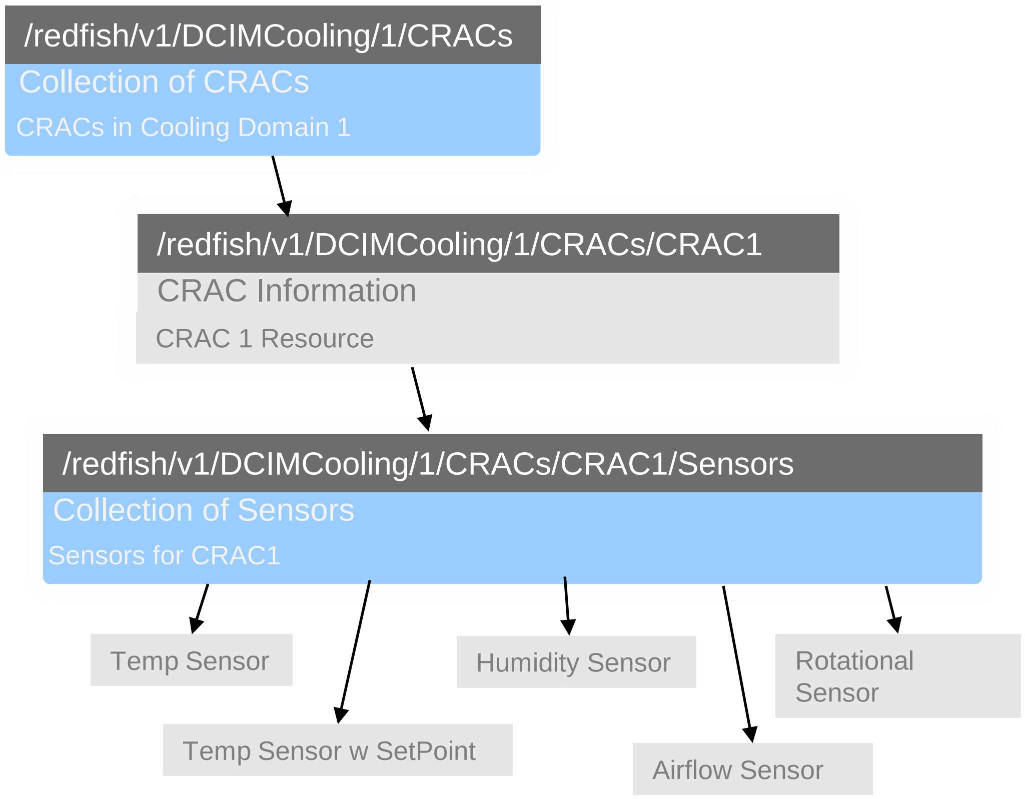
f) Alarms
- **All Alarms represented in subordinate Alarms collection**
- **Triggered Alarms represented in array of references to Alarms collection members**

g) Equipment sub-components are represented as 1 or more members of top level array property.
- **Each sub-component is represented as a property of object type with the top level properties**
    - **Component inventory information**
    - **Status**
    - **Metrics**
    - **SensorType arrays, one array for each SensorType in the sub-component**
    - **Unique component properties and actions**

Example CRAC Resource w Sensor Collection Modeling

Example of CRAC JSON document organization

```
{
    “Name” : “CRAC 1”,
    “Model” : “ABC123”,
    “SerialNumber” : “1234567”,
    “FWRevision” : “1.5.0”,
    “HWRevision” : “2.0.0”,
    “ManufactureDate” : “01012017”,
    “PhysicalLocation: {<…>}”,

    “CoolOutputKW” : “2.3”,
    “CoolDemandKW” : “1.4”,
    “AvgSupplyTempC” : “25”,
    “AvgReturnTempC” : “28”,
    “AvgFanSpeedRPM” : “2300”,
    “AvgAirFlowCFM” : “15”,

     “Sensors” : “<sensors collection URI>”,
     “Temperature” : [“<sensor URI>”, “<sensor URI>”],
     “Humidity” : [“<sensor URI>”],
     “AirFlow” : [“<sensor URI>”],
     “Rotational” : [“<sensor URI>”],
     “SetPoint” : [“<temp sensor w SetPoint URI>”]
     “Status” : {
                      “State” : “Enabled”
                      “Health” : “OK”  },
    “Links” : {<…>},
    “Actions” : “<actions that affect CRAC>”

}
```



Legend:
Blue = Collection Resource
Grey = Singleton Resource

Equipment Inventory Modeling

Example Inventory property JSON data

```
"Name": "Generator",
"FirmwareRevision": "1.0.0",
"DateOfManufacture": "2017-01-11T08:00:00Z",
"Manufacturer": “ABC Manufacturer",
"Model": “20KW Model",
"SerialNumber": “ABC123",
"PartNumber": “XYZ123",
"AssetTag": “Backup Generator #1",
```

Equipment inventory information is represented by a consistent set of top level properties in the equipment entity schema.

Every DCIM (and other Redfish IT equipment schema) equipment schema includes the common inventory properties.

Equipment Location Modeling

Example PhysicalLocation property JSON data

```
"PhysicalLocation": {
        "Longitude": 45.52,
        "Latitude": 122.67,
        "PostalAddress": {
            "Country": "US",
            "Territory": "OR",
            "City": "Portland",
            "Street": "1001 SW 5th Avenue",
            "Name": "Distributed Management Task Force, Inc.",
            "PostalCode": "97204",
            "Building": "3A",
            "Floor": "2",
            "Room": "213"
        },
        "Placement": {
            "Row": “2-North"
        }
    },
```

Equipment location is represented by the PhysicalLocation top level property in the equipment entity schema.

Every DCIM equipment schema includes the PhysicalLocation property.

The PhysicalLocation property is a common Redfish property of type Resource.Location object.

The Resource.Location object has properties supporting international addresses and other types of location information.

Sensor Modeling

Example Sensor array JSON data

```
 "Temperature": [
       {
          "Id": "0",
          "Name": "Generator Exhaust Temperature Sensor",
          "SensorType": "Temperature",
          "Status": {
             "State": "Enabled",
             "Health": "OK"
          },
          "Reading": 55,
          "SetPoint": 52,
          "DeadBand": 5,
          "UpperThresholdWarning": 90,
          "LowerThresholdWarning": 20,
          "UpperThresholdCritical": 100,
          "LowerThresholdCritical": 10,
          "MinReadingRange": 0,
          "MaxReadingRange": 110,
          "PhysicalContext": "Exhaust",
}
  ]
```

All sensors are modeled in a common way with a SensorType property differentiating between different kinds of sensors.

Sensors of various types are represented in the equipment schema as individual members of a Sensors Collection subordinate to each equipment model entity.

Array properties of specific sensor types are located:

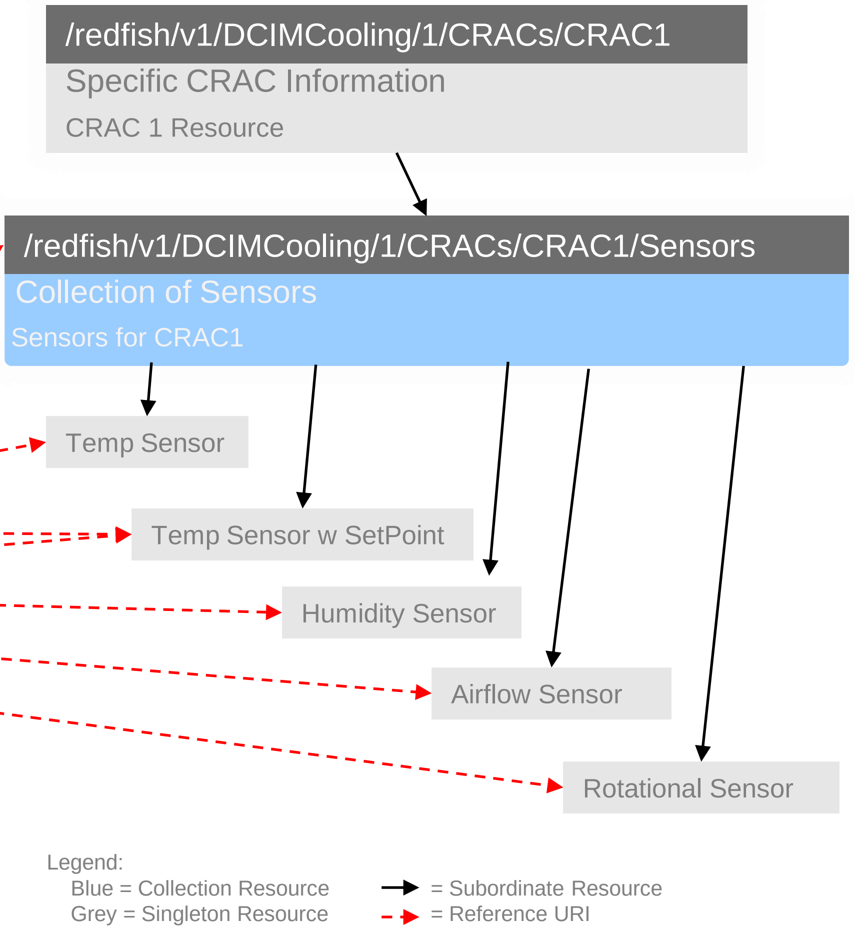
- **As top level properties in equipment entity schema**
- **Within sub-component definitions.**

SensorType enumeration values		
Temperature	Humidity	Amperage
ACVoltage	DCVoltage	Frequency
Pressure	LiquidLevel	Rotational
AirFlow	LiquidFlow	Barometric
Altitude		

Sensor Modeling Concept Example

Example of Sensor JSON document organization

```
{
   “Name” : “CRAC 1”,
   “Model” : “ABC123”,
   “SerialNumber” : “1234567”,
   “FWRevision” : “1.5.0”,
   “HWRevision” : “2.0.0”,
   “ManufactureDate” : “01012017”,
   “CoolOutputKW” : “2.3”,
   …

    “Sensors” : “<sensors collection URI>”,

    “Temperature” : [“<sensor URI>”, “<sensor URI>”],
    “SetPoint” : [“<temp sensor w SetPoint URI>”]
    “Humidity” : [“<sensor URI>”],
    “AirFlow” : [“<sensor URI>”],
    “Rotational” : [“<sensor URI>”],

    “Status” : {
                    “State” : “Enabled”
                    “Health” : “OK”  },
    “Links” : {},
}
```



/redfish/v1/DCIMCooling/1/CRACs/CRAC1

Specific CRAC Information

CRAC 1 Resource

/redfish/v1/DCIMCooling/1/CRACs/CRAC1/Sensors

Collection of Sensors

Sensors for CRAC1

Temp Sensor

Temp Sensor w SetPoint

Humidity Sensor

Airflow Sensor

Rotational Sensor

Legend:

Blue = Collection Resource → = Subordinate Resource

Grey = Singleton Resource ⇢ = Reference URI

SetPoint and Deadband Modeling

Example SetPoint/Deadband JSON data

```
"Temperature": [
      {
         "Id": "0",
         "Name": "Generator Exhaust Temperature Sensor",
         "SensorType": "Temperature",
         "Status": {
            "State": "Enabled",
            "Health": "OK"
         },
         "Reading": 55,
         "SetPoint": 50,
         "DeadBand": 5,
         "UpperThresholdCritical": 100,
         "LowerThresholdCritical": 10,
         "MinReadingRange": 20,
         "MaxReadingRange": 80,
         "PhysicalContext": "Exhaust",
         "Oem": {}
      }
]
```

A SetPoint is defined as the desired or target value for a process value of a system as represented by the Reading value of a Sensor. Departure of the Reading from its SetPoint is one basis for automatic control of equipment.

A Deadband is defined as the band of values around the SetPoint where the control system does not respond to the sensor Reading varying from the SetPoint setting.

SetPoint and DeadBand properties are optional properties located in the Sensor schema and would be present when an equipment setpoint is related that that specific sensor.

A SetPoint array property located as an equipment top level property contains references to the set of Sensor resources that have SetPoints defined.

Alarm Modeling Goals and Approach

Goals:

- Define what an alarm is and related behavior
- Create “Alarm” data definitions for DCIM and other IT use cases
- Enable discovery of available and active alarms
- Enable re-arm, enable/disable and acknowledge of specific alarms
- Create Alarm Registries (similar to Event/Message Registries) to define ‘Alarms’
  - Supporting both standard and OEM alarm definitions
- Need ability for pre-defined alarms and ability for users to create their own

Modeling Approach:

Analysis of existing types of Alarm representations in existing equipment interfaces indicated similarities with asynchronous eventing and logging mechanisms.

Existing Redfish event and log message modeling constructs provide a basis to model Alarms and this approach enables an Alarm to have an affinity with log entries and events and to leverage already established behavior.

Alarm Modeling

Example Alarm JSON data

```
{
   "Id": "3",
   "Name": "Engine Low Oil Pressure Warning Alarm",
   "Severity": "Warning",
   "TriggerTime": "2012-03-07T14:45:00Z",
   "Message": "Engine Low Oil Pressure Warning Alarm",
   "MessageId": "Alarm.1.0.EngineLowOilPressureWarningAlarm",
   "MessageArgs": [],
   "AlarmState": "Triggered",
   "Acknowledged": true,
   "Links": {
      "RelatedSensor": {
         "@odata.id": "/redfish/v1/DCIMPower/default/Generator/0/Sensors/8"
      }
   },
}
```

An Alarm is a resource that has a latch type behavior. It is designed to be used to persist sensor threshold crossing activity. Also, an alarm can reflect the momentary state of another property or equipment characteristic.

An Alarm can be triggered by an event such as sensor reading or property value changes. An Alarm can be re-armed by instrumentation or by a client.

An equipment representation has a subordinate Alarms collection whose members are all the Alarms instrumented for that equipment.

A TriggeredAlarms top level equipment array property contains references to specific Alarm resources in the equipment Alarms collection that are triggered.

Alarms are defined in one or more Alarm Registry JSON resources similar to a Message Registry. There can be standard and OEM Alarm definitions.

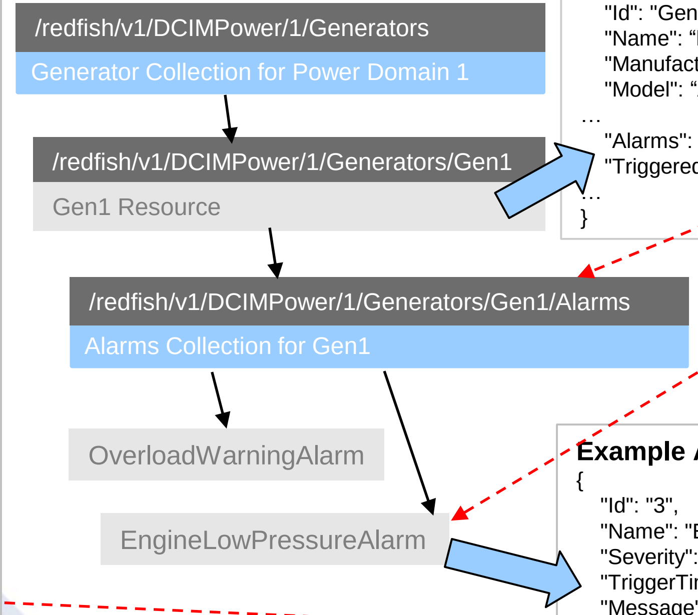
The Alarm data definition is similar to an Event definitions with these additional properties:

- AlarmState = Can be “Armed”, “Triggered”, “Disabled”
- Acknowledged = true/false
- Links.RelatedSensor = reference to sensor the alarm is related to
- Links.RelatedProperty = reference to a property the alarm is related to

Alarm Modeling Concept Example

Example of Alarm Registry JSON document

```
{
…

   "Id": "Alarm.0.1.0",
   "Name": "Alarm Registry",
   "Language": "en",
   "Description": “Base Alarms Registry for Redfish",
   "RegistryPrefix": "Base",
   "RegistryVersion":  "0.1.0",
   "OwningEntity": "DMTF",
   "Messages": {
      "OverloadWarningAlarm": {
         "Description": "Generator Alarm",
         "Message": "Overload Warning Alarm",
         "Severity": "Warning",
         "NumberOfArgs": 0
      },
      "OverloadCriticalAlarm": {
         "Description": "Generator Alarm",
         "Message": "Overload Critical - Shutdown Alarm",
         "Severity": "Critical",
         "NumberOfArgs": 0
      },
      "EngineLowOilPressureWarningAlarm": {
         "Description": "Generator Alarm",
         "Message": "Engine Low Oil Pressure Warning",
         "Severity": "Warning",
         "NumberOfArgs": 0
      },

…

}
```



/redfish/v1/DCIMPower/1/Generators
Generator Collection for Power Domain 1
/redfish/v1/DCIMPower/1/Generators/Gen1
Gen1 Resource
/redfish/v1/DCIMPower/1/Generators/Gen1/Alarms
Alarms Collection for Gen1
OverloadWarningAlarm
EngineLowPressureAlarm

Example Generator Resource JSON Document

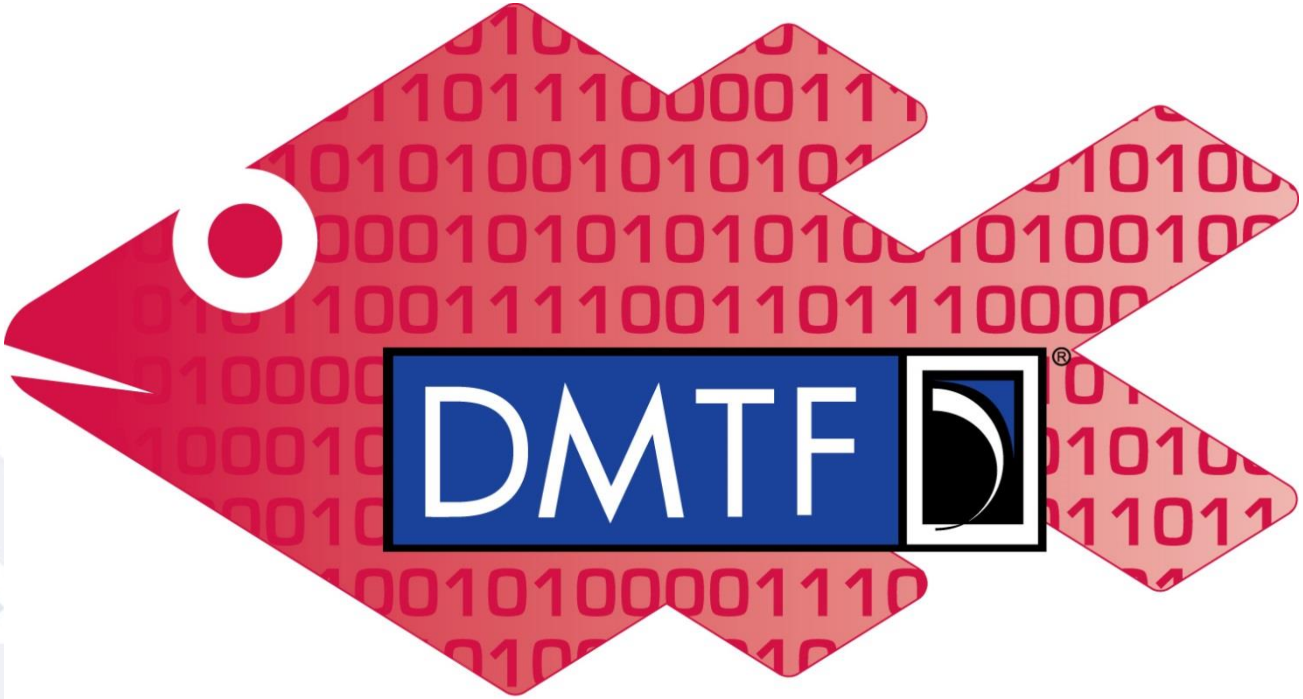
```
{
   "@odata.type": "#Generator.v0_2_0.Generator",
   "Id": "Gen1",
   "Name": “Lab Generator 1",
   "Manufacturer": “ABC Power Equipment",
   "Model": “AC Generator Model X",
…
   "Alarms": { <Alarm Collection URI> }
   "TriggeredAlarms" : [ { <Alarm Reference URI } ],
…
}
```

Example Alarm Resource JSON Document

```
{
   "Id": "3",
   "Name": "Engine Low Oil Pressure Warning Alarm",
   "Severity": "Warning",
   "TriggerTime": "2012-03-07T14:45:00Z",
   "Message": "Engine Low Oil Pressure Warning Alarm",
   "MessageId": "Alarm.1.0.EngineLowOilPressureWarningAlarm",
   "MessageArgs": [],
   "AlarmState": "Triggered",
   "Acknowledged": true,
   "Links": {
      "RelatedSensor": { <Sensor Resource Reference URI> }
      }
   },
…
}
```

Legend:
Blue = Collection Resource
Grey = Singleton Resource

Q&A & Discussion



Redfish