Redfish DCIM Work in Progress

DMTF Redfish Forum – DCIM Task Force

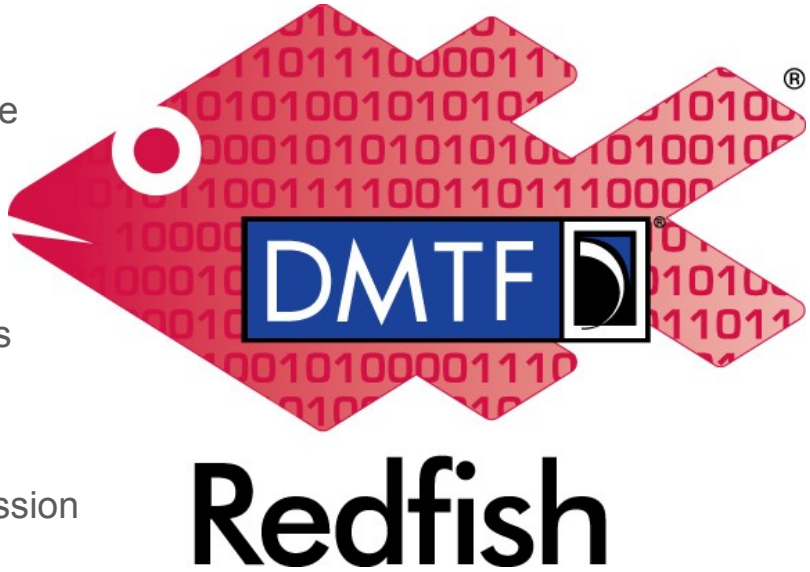
Version 0.9 – May 2019

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



DCIM Work in Progress v0.9

- Power Equipment and HVAC resource definitions
- Facility model to locate equipment and tie it to a physical space
- Power and Cooling Domain group concepts
- Models for power distribution equipment
  - Rack PDU's
  - Floor PDU's
  - Transfer switches
- Supporting models for power equipment
  - Circuit, Outlet, Outlet Groups
  - Power Distribution Unit Metrics
- DSP-IS0005 README includes a "DCIM Schema Guide"
  - Documents all DCIM work-in-progress schemas
  - Text is generated from schema "description" contents

SENSOR MODEL

Sensor definition

- A Redfish Sensor is a monitoring device which produces a single reading
  - May include other related, time-coherent readings
  - Reports physical context and other “purpose” identification properties
  - Metadata describing accuracy, sampling frequency, etc.
  - Simple thresholds which indicate a change of state of the monitored item

- Base sensor definition:
  - “ReadingType” – The sensor type (temperature, voltage, etc.)
  - “Reading” – Value of the sensor (no Units in name to allow for generic software use)
  - “ReadingUnits” – Applies to all values in sensor (thresholds, reading)
    - Units shall be explicitly defined for each ReadingType to avoid mis-identification.
    - Keep to one order-of-magnitude usage per Unit of Measure if possible
    - Large-scale difference in magnitude may warrant separate ReadingTypes (e.g. mV, kV)
  - “Thresholds” – Set of defined thresholds with a common structure
    - Severity, hysteresis, etc.

Example: Voltage Sensor

```
{
   "@odata.id": "/redfish/v1/Chassis/1/Sensors/VRM1",
   “@odata.type”: “Sensor.v1_0_0.Sensor”,
   “ReadingType”: “Voltage”,
   "Name": "VRM1 Voltage",
   "SensorNumber": 11,
   "Status": {
      "Health": "OK"
   },
   "Reading": 12,
   “ReadingUnits”: “V”,
   “Thresholds: {
      "UpperCritical": {
         “Reading”: 13
      }
   },
   "MinReadingRange": 0,
   "MaxReadingRange": 20,
   "PhysicalContext": "VoltageRegulator",
   "RelatedItem": [
       { "@odata.id": "/redfish/v1/Systems/1" } ]
},
```

- DCIM models utilize a common “Sensor” schema to model devices which provide a single reading (or a set of coherent readings) as its function.
- New “ReadingType” used to differentiate between types of sensors
- “Reading” is separate from “ReadingUnits” in the Sensor definition to allow common telemetry usage (an exception to the Redfish naming convention).
- Value of “ReadingUnits” applies to all Reading-related properties, and is defined by the ReadingType with only one unit allowed per type for interoperability.
- SensorNumber is a legacy IPMI concept (used in existing Power/Thermal schemas) which we may be able to abandon here…

Sensor Collections, Filtered Arrays and Excerpts

- *Sensor* Collection is a Resource Collection which holds all sensors housed in a Chassis or Facility (perhaps other containment)
- A Filtered Array allows convenient access to a subset of a collection
  - E.g. Temperatures[] contains all temperature sensors
  - This shows up in the WIP mockup for RackPDU "TriggeredAlarms"
- Redfish Excerpt concept allows "copies" of sensor data to exist where desired for common use cases without duplication of the entire Sensor resource contents

RESOURCE EXCERPTS

Resource Excerpt Goals

- Provide compact response for sensor payloads
  - Deliver sensor reading and *enough* context for normal usage
  - End users “just need to get temperatures…”
- Ensure full data can be discovered
  - Avoid requirement of query parameters or other “hidden” protocol features
- Minimize implementation overhead
  - Share code between the sensor resource and any “compact” version
- Create concept that could apply elsewhere in Redfish, not just Sensor

Excerpt Property Concept

- Allow a copy of selected data from a Collection member
  - But avoids addition of Links/Members or other schema-induced overhead
- Include only *essential* or *dynamic* properties for common usage
  - Reduce static payload size and processing of frequently polled resources
  - Definition of Excerpt properties set in the Collection member's schema
- Provide link to the full resource instance in a Collection
  - "DataSourceUri" property (value is URI for the Member of a Collection)
- Excerpt properties are not necessarily required
  - Properties may be Required and an Excerpt: "Reading", "ReadingUnits"
  - But optional properties may also be Excerpt: "PhysicalContext"

Creating an Excerpt

- Add an object to contain the subset of properties from another resource
  - Contents will be the Excerpt properties from the target resource
- This object can be a single instance or a Filtered Array
  - “Temperature”:{excerpt} or “Temperatures”: [{excerpt}, {excerpt}, …]
- Filtered Array instances all share the same sub-type (key property)
  - Follows normal array pattern for Redfish
  - Example: “Temperatures”[ ] contains only temperature sensors, not humidity, fans, or other types of Members in the Sensors Collection
- Service returns only supported Excerpt properties
  - Properties marked in referenced schema as “Redfish.Excerpt”
  - Only those properties supported by implementation
  - If resource contains no marked properties, Service returns entire resource

Example: Voltage Sensor

```
{
   "@odata.id": "/redfish/v1/Chassis/1/Sensors/VRM1",
   “@odata.type”: “Sensor.v1_0_0.Sensor”,
   “ReadingType”: “Voltage”,
   "Name": "VRM1 Voltage",
   "SensorNumber": 11,
   "Status": {
      "Health": "OK"
   },
   "Reading": 12,
   “ReadingUnits”: “V”,
   “Thresholds: {
      "UpperCritical": {
         “Reading”: 13
      }
   },
   "MinReadingRange": 0,
   "MaxReadingRange": 20,
   "PhysicalContext": "VoltageRegulator",
   "RelatedItem": [
       { "@odata.id": "/redfish/v1/Systems/1" } ]
},
```

Highlighted data included in Excerpt

Excerpts tied to a single ReadingType

Name needed for context within arrays

Status is important data about the sensor

Reading (with Units) is the primary property

Thresholds and Ranges are static data and therefore not included in excerpts

Physical context used to identify excerpts

Example: Excerpt Copy of Voltage Sensor

```
"Voltages": [
  {
     "DataSourceUri": "<path>/Sensors/VRM1",
     "Name": "VRM1 Voltage",
     "Status": {
        "Health": "OK"
     },
     "Reading": 12,
     "ReadingUnits": "V",
     "PhysicalContext": "VoltageRegulator"
  },
  . . .
],
```

- Voltages[] Filtered Array would appear where useful in other (non-Sensor) resources
- DataSourceUri link points to Sensor collection
  - This property is constructed for the Excerpt, and does not appear in the referenced collection member
- Includes Excerpt properties from "Sensor":
  - Reading
  - ReadingUnits
  - Name (from Resource)
  - Status (from Resource)
  - PhysicalContext – Need for multiple sensors (CPU vs Ambient)
- Excludes non-Excerpt properties:
  - Thresholds
  - Sensor capabilities
  - RelatedItems

Redfish Schema Annotations for Excerpt properties

- **Excerpt:** Property should appear in referenced copies
  - <Annotation Term="Redfish.Excerpt" String="Sensor.ReadingType\RPM"/>
  - Optional string value indicates a "key" property and enumeration value used to indicate that the property is only used for certain types
  - If no string value is provided, then the property is Excerpt for all uses
  - This conditional inclusion is used during conversion to other schema languages and for documentation generation, although it may be useful for code generation utilities.
- **ExcerptCopy:** Object references Excerpt properties from a resource
  - <Annotation Term="Redfish.ExcerptCopy" String="Sensor.ReadingType\RPM"/>
  - The optional string again references the key property and value, and it is this value that is used to match the conditional inclusion of Excerpt properties.
- **ExcerptCopyOnly:** Property only populated in copies, not original resource
  - <Annotation Term="Redfish.ExcerptCopyOnly"/>
  - Used for pointer property to the original resource, which would be a duplicate 'self' pointer in that original resource.

Excerpt Query Parameter

- When used, service returns only the “excerpt” properties in payload
  - Optional, but recommended, protocol feature
  - Example: `GET /redfish/v1/Chassis/Sensors/CPUTemp1?excerpt`
- Lacks “leading-$” naming to allow service to safely ignore as unknown
  - Specification requires service to reject unknown “$” parameters
  - In this case, no harm to returning entire payload
- Combines well with $expand, $filter on collections
  - `GET \redfish\v1\Chassis\1\Sensors?excerpt&$expand&`
    `$filter=“Members/ReadingType”%20eq%20“Temperature”`

Recommended usage

- Ad hoc request, simple clients: GET resources with Excerpts defined
- Discovery of sensors: GET Sensor Collection with $expand
- Sensor details or settings: GET Sensor Collection member
- Frequent polling individual sensor: GET member with “excerpt” query
  - “Get reading for temperature sensor #4”
  - But use TelemetryService features to avoid polling!
- Polling by sensor type (GUI): Sensor collection with $expand, $filter
  - Can combine with “excerpt” for better performance

ALARMS

Alarm definition

- Many types of equipment encounter conditions needing immediate attention
  - But in many cases these are difficult to represent as properties (data), as the condition may not be directly measured or observed
  - Or the condition could only be represented in “normal” / “attention required” states
  - In the past, these conditions were rendered as SNMP Traps
  - Examples: “Circuit overload”, “Moisture detected”, etc.
- Rather than create large numbers of properties that have static “normal” values except in rare cases, Redfish defines a collection of “Alarms”
  - Leverages Redfish LogEntry resource format – inherit by copy…
  - But unlike a Log, the entries are static based on Alarm definitions (not removed from collection)
    - One “entry” (collection member) per supported Alarm
    - This allows discovery of available alarms
    - AlarmStatus allows configuration, enable/disable of specific alarms and actions
- TriggeredAlarms[] array
  - Filtered Array and Excerpt of Alarm collection for active alarms
  - Provides simple annunciator panel output

Alarm Example

```
"@odata.id": "/redfish/v1/DCIMPower/default/RackPDU/1/Alarms/Overload,
"@odata.type": "#Alarm.v0_9_0.Alarm",
"Id": "Overload",
"Name": "PDU Unit Overload",
"AlarmState": "Triggered",
“Acknowledged": false,
"Severity": "Critical",
"TriggerTime": "2018-08-07T14:44:00Z",
"AutomaticReArm": true,
"Message": "Rack PDU Overload Condition",
"MessageId": "DCIM.0.1.0.Overload",
"MessageArgs": [
    "58703"
],
"Links": {
    "RelatedSensor": {
        "@odata.id": “<Sensor URI>/ACMainPower"
    },
    "Oem": {}
},
```

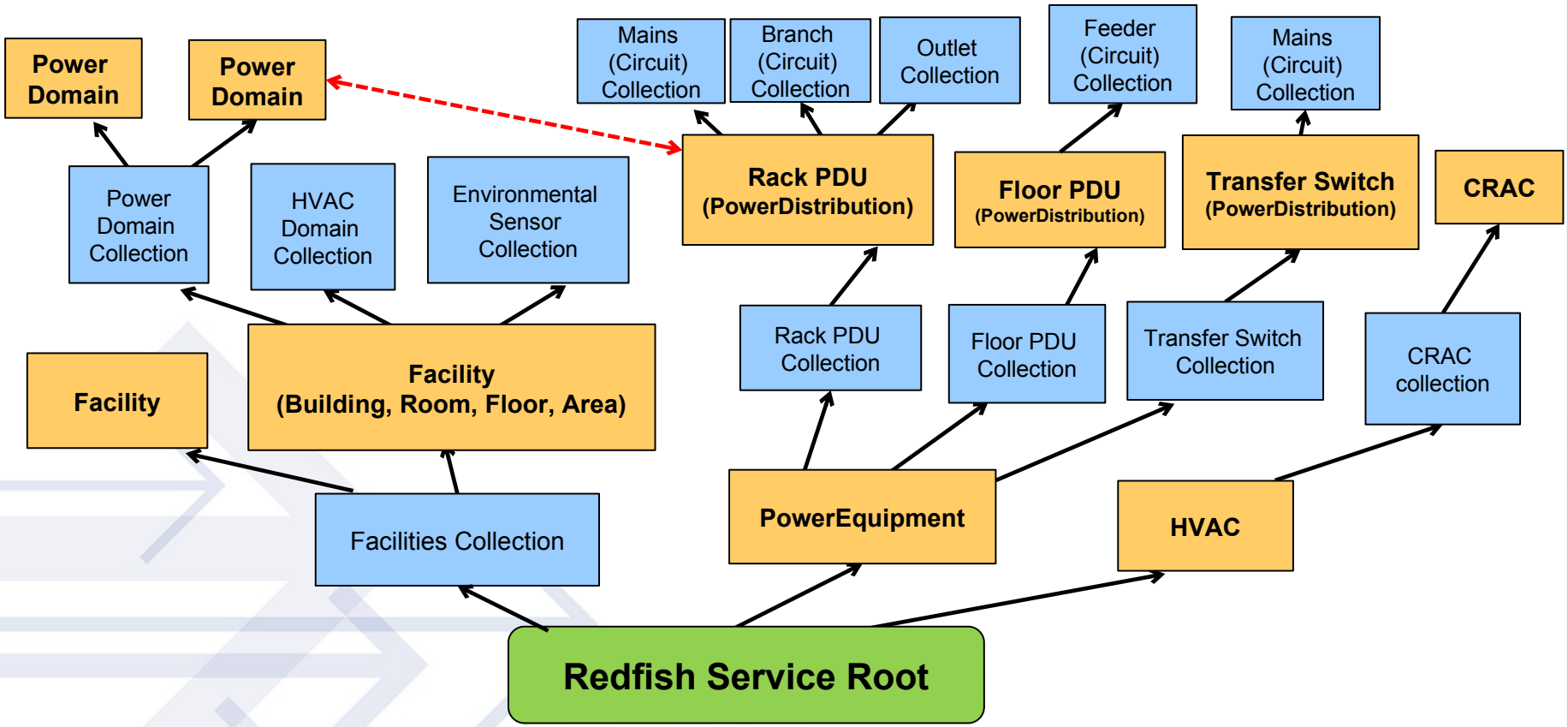
- Uses Redfish Message concepts
- Allows for automatic re-arm
  - May need more options
  - Re-arm at reset, time elapsed, etc.
- AlarmState = “Disabled”, “Armed”, “Triggered”
- Use PATCH to acknowledge & clear
- Links to sensors, related items

FACILITY MODEL

Facility resource

- New Resource Collection off of the ServiceRoot
  - Facility provides a physical model of infrastructure components
  - Similar to the “Chassis” model used in conjunction with ComputerSystem
  - FacilityType enumeration: Room, Floor, Building, etc.
  - Contains / ContainedBy links to show Facility relationships
- Sensor collection (e.g. environmental sensors for a room)
- Power and HVAC Domain Collections
  - User-defined groups of related gear
  - Can be a logical or physical grouping
- Reference links to physical equipment contained within the Facility
  - Power Distribution equipment
  - HVAC equipment
  - Chassis

DCIM Additions to Redfish Resource Tree



HVAC MODEL

HVAC resource

- Single resource instance off of the ServiceRoot
  - Contains links to all HVAC related equipment
  - Used primarily for discovery of managed equipment
  - Isolates ServiceRoot from future, rapid expansion of DCIM coverage
- Links to Collections of HVAC-related equipment
  - This is currently a placeholder for future work

- Call to Action: HVAC equipment schema definitions needed
  - Join the Redfish Forum and DCIM Task Force
  - Submit proposals to DMTF through the feedback portal

POWER EQUIPMENT MODEL

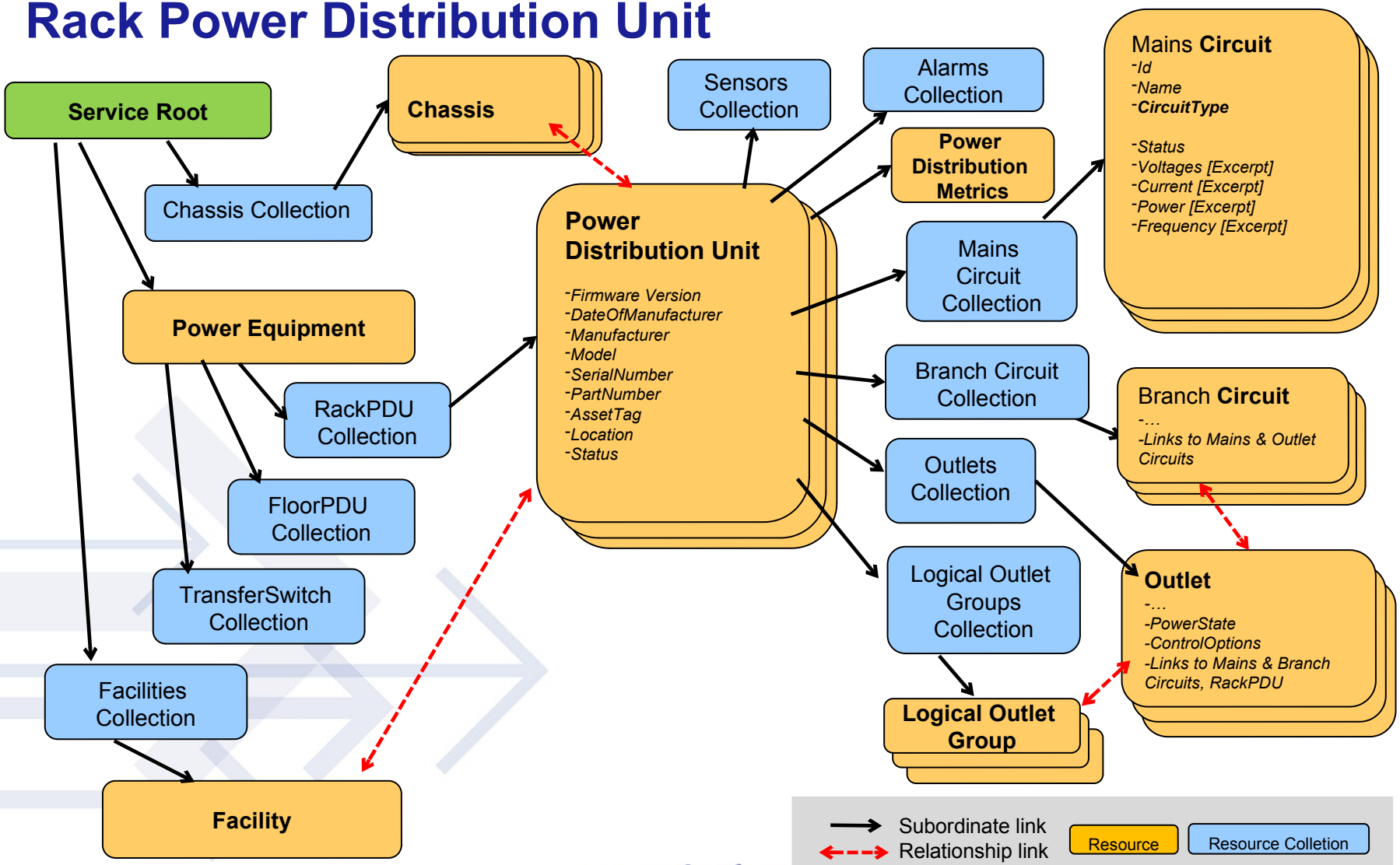
Power Equipment resource

- Single resource instance off of the ServiceRoot
  - Contains links to all power or energy-related equipment
  - Used primarily for discovery of managed equipment
  - Isolates ServiceRoot from future, rapid expansion of DCIM coverage
- Links to Resource Collections of:
  - Rack PDUs
  - Floor PDUs
  - Transfer Switches
  - Switchgear
  - UPSs (expected future work)
  - Generators (expected future work)
- Call to Action: Additional power equipment schema definitions needed
  - Join the Redfish Forum and DCIM Task Force
  - Submit proposals to DMTF through the feedback portal

PowerDistribution schema and resources

- Unified schema defines several types of power distribution gear
  - Share common modeling and property definitions
  - *EquipmentType* property provides specific identification
  - Separate collections of each type linked from PowerEquipment resource
- Resource contents
  - General product identification – model, manufacturer, serial number, etc.
  - Versioning – Hardware revision, firmware version, date of manufacture
  - TriggeredAlarms – Filtered Array of the Alarm collection
- Links to subordinate Resources and Resource Collections
  - Alarm Collection, Sensor Collection, Metrics (entire unit)
  - Mains (input) Circuit Collection
  - Branch, Feeder, Subfeed Circuit Collections (as applicable)
  - Outlet Collection for all receptacles in unit
  - Logical Outlet Groups to allow for user or vendor-defined groupings

Rack Power Distribution Unit



Circuit schema

- Single schema type to describe input and output circuits
  - **NEW**: *Outlet* schema now a copy of *Circuit* with outlet-specific properties
- Ratings
- Metrics
  - Single and multi-phase current and voltage structures
  - Power, Energy, Frequency
- **NEW**: PlugType – IEC, NEMA or other standard plug type
- Outlets - links to outlets provided by circuit
- Actions
  - PowerControl
  - BreakerControl
  - ResetStatistics

Circuit payload example

```
{
   "@odata.type": "#Circuit.v0_9_0.Circuit",
   "Id": "A",
   “Name": "Branch Circuit A",
   "Status": { < Status object> },
   "CircuitType": "Branch",
   "PhaseWiringType": "TwoPhase3Wire",
   "NominalVoltage": "AC240V",
   "RatedCurrentAmps": 16,
   "BreakerState": "Normal",
   “PowerState”: “On”,
   “VoltageSensor”: { < Single-phase voltage sensor > },
   "PolyPhaseVoltageSensors": { < Voltage per phase sensors > },
   "CurrentSensor": { < Total Current sensor > },
   "PolyPhaseCurrentSensors": { < Current per phase sensors > },
   "PowerSensor": { < Total Power sensor > },
   "PolyPhasePowerSensors": { < Power per phase sensors > },
   "FrequencySensor": { < Frequency sensor > },
   "EnergySensor": { < Energy sensor > },
   "Outlets": [{ < links to contained outlets > }],
   “Actions": { < ResetBreaker, ResetStatistics > }
   "@odata.id": "/redfish/v1/PowerEquipment/RackPDUs/1/Branches/A",
}
```

- Circuit
- Standard Status object
- Ratings and electrical details
- Power and Breaker states
- Sensor (excerpts) for Voltage, Current, Power, Energy and Freq.
- List of associated Outlets

Circuit schema: Sensor details

- Each *Circuit* resource contains numerous sensor excerpts
  - Readings for Voltage, Current, Power, Energy, and Frequency
  - Energy and Power sensor excerpts may contain additional properties to report reactive values and power factor
  - May contain *PeakReading*
  - All readings within an excerpt follow *ReadingUnits* (Watts in this example)

```
"PowerSensor": {
   "DataSourceUri": "/redfish/v1/PowerEquipment/RackPDUs/1/Sensors/PowerB",
   "Name": "Branch B Power",
   "Reading": 977.8,
   "PeakReading": 1000.9,
   "ReadingUnits": "W",
   "ApparentVA": 1104.2,
   "ReactiveVAR": 512.9,
   "PowerFactor": 0.88
},
```

Circuit schema: Poly-phase Sensor objects

- Poly-phase circuits will contain multiple sensors of the same type
  - Voltage and Current are measured between phases or neutral
  - The context of the electrical measurement (e.g. “Line 1 to Neutral”) for each of these is important
- JSON object contains a nested set of sensor excerpts
  - Nested excerpt object names match *ElectricalContext* of the sensor
  - Object contains only the excerpts that apply to the *Circuit*
  - This allows easy, deterministic programmatic access “by name”
- Single-phase circuits also contain a singular sensor excerpt
  - E.g. *VoltageSensor* excerpt in addition to *PolyPhaseVoltageSensors*
  - Frequent use case is to monitor single-phase outlets
    - Removes requirement for software to be aware of phase configuration
  - The poly-phase object will contain the (duplicate) single sensor
    - Provides consistency for phase-aware client software

Circuit schema: Poly-phase Sensor example

```
"PolyPhaseVoltageSensors": {
   "Line2ToNeutral": {
      "DataSourceUri": "/redfish/v1/PowerEquipment/RackPDUs/1/Sensors/VoltageBL2N",
      “Name": "Branch B Voltage L2N",
      "Reading": 116.7,
      "ReadingUnits": "V"
   },
   "Line2ToLine3": {
      "DataSourceUri": "/redfish/v1/PowerEquipment/RackPDUs/1/Sensors/VoltageBL2L3",
      "Name": "Branch B Voltage L23",
      "Reading": 203.6,
      “ReadingUnits": "V"
   }
},
```

Outlet schema

- **NEW**: Schema type to describe individual receptacles
  - *Outlet* schema a copy of *Circuit* with outlet-specific properties
  - Ability to render un-managed or un-monitored outlets as well
- Ratings
- Metrics
  - Single and multi-phase structures
- **NEW**: OutletType
  - Re-defined enumerations to list NEMA, IEC and other types in one list
- Actions
  - PowerControl
  - ResetStatistics

Outlet example

```
{
  "@odata.type": "#Outlet.v0_9_0.Outlet",
  "Id": "A3",
  "Name": "Outlet A3, Branch Circuit A",
  "Status": {
    "Health": "OK",
    "State": "Enabled"
  },
  "PhaseWiringType": "TwoPhase3Wire",
  "VoltageType": "AC",
  "OutletType": "C13",
  "RatedCurrentAmps": 12,
  "NominalVoltage": "AC240V",
  "IndicatorLED": "Lit",
  "PowerOnDelaySeconds": 10,
  "PowerOffDelaySeconds": 0,
  "PowerState": "On",
  "PowerEnabled": true,
  "VoltageSensor": {
    "DataSourceUri": <URI of sensor resource>,
    "Name": "Outlet A3 Voltage L12",
    "Reading": 202.3,
    "ReadingUnits": "V"
  },
  "PolyPhaseVoltageSensors": {
    "Line1ToLine2": {
       "DataSourceUri": <URI of sensor resource>,
       "Name": "Outlet A3 Voltage L12",
       "Reading": 202.3,
       "ReadingUnits": "V"
    }
  },
```

```
  "CurrentSensor": {
     "DataSourceUri": <URI of sensor resource>,
     "Name": "Outlet A3 Current",
     "Reading": 1.73,
     "PeakReading": 2.50,
     "ReadingUnits": "A"
  },
  “PolyPhaseCurrentSensors": {
     "Line1": {
        "DataSourceUri": <URI of sensor resource>,
        "Name": "Outlet A3 Current",
        "Reading": 1.73,
        "PeakReading": 2.50,
        "ReadingUnits": "A"
     }
  },
  "PowerSensor": {
     "DataSourceUri": <URI of sensor resource>,
     "Name": "Outlet A3 Power",
     "Reading": 349.9,
     “PeakReading": 505.7,
     "ReadingUnits": "W",
     "ApparentVA": 349.9,
     "ReactiveVAR": 0.0,
     "PowerFactor": 1.00
  },
```

<< continued on next slide >>

Outlet example, continued

```
<< continued from previous slide >>

  "FrequencySensor": {
     "DataSourceUri": "/redfish/v1/PowerEquipment/RackPDUs/1/Sensors/FrequencyA3",
     "Name": "Outlet A3 Frequency",
     "Reading": 60.0,
     "ReadingUnits": "Hz"
  },
  "EnergySensor": {
     "DataSourceUri": "/redfish/v1/PowerEquipment/RackPDUs/1/Sensors/EnergyA3",
     "Name": "Outlet A3 Energy",
     "Reading": 61848,
     "ReadingUnits": "kW"
  },
  "Actions": {
     "#Outlet.PowerControl": {
         "target": "/redfish/v1/PowerEquipment/RackPDUs/1/Outlets/A3/Outlet.PowerControl"
     },
      "#Outlet.ResetStatistics": {
         "target": "/redfish/v1/PowerEquipment/RackPDUs/1/Outlets/A3/Outlet.ResetStatistics"
     }
  },
  "Links": {
    "BranchCircuit": {
         "@odata.id": "/redfish/v1/PowerEquipment/RackPDUs/1/Branches/A"
     }
  },
  “@odata.id": "/redfish/v1/PowerEquipment/RackPDUs/1/Outlets/A3“
}
```

QUESTIONS FOR INDUSTRY

Open topics and further work in progress

- Expect additional object(s) in *PowerDistribution* schema to provide details on Automatic Transfer Switch configuration
  - Conditions for switching
  - Configuration of ‘primary’ input
  - May leverage existing Redfish *Redundancy* schema
- UPS schema under development
  - Expected to be a superset of *PowerDistribution* schema
    - Separate schema, but shares property, object, and link definitions
  - Expect coverage of rack and room-scale UPS products
  - Will likely add a link to a Resource Collection for Batteries
  - Battery subsystem may span multiple resources (depending on scale)

Q&A & Discussion