Redfish for Thermal Equipment

WORK IN PROGESS

DMTF Redfish Forum

October 2022

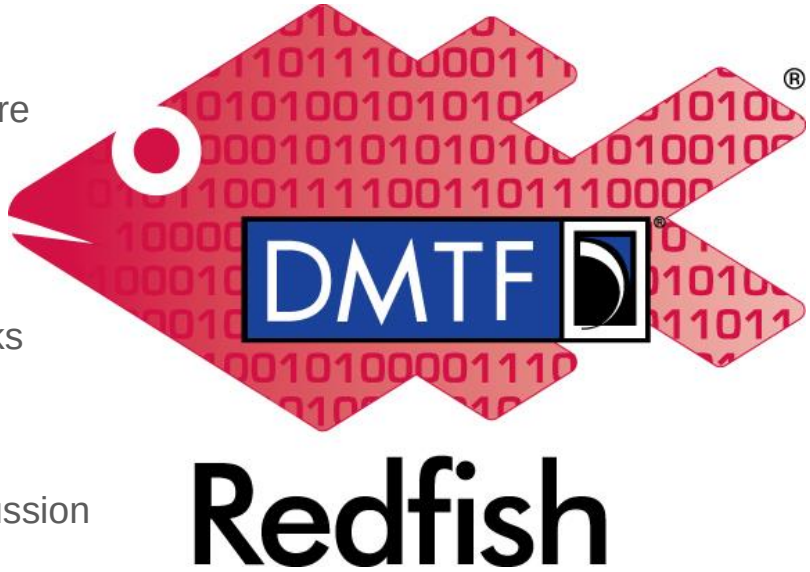
V0.9

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 DMTF website: www.dmtf.org

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

- Proposal to extend Redfish DCIM models to incorporate cooling units
  - Support for rack-based Cooling Distribution Units (CDUs)
  - Support for immersion cooling units
  - Models should apply generally to other liquid cooling gear
    - Rear-door heat exchangers, air conditioners, etc.
  - Expect the model to also cover air-cooling systems
    - Explicit coverage is not shown in this proposal, but some notes are mentioned
  - Intend to model all equipment types covered by OCP requirements
- Leverages existing Redfish DCIM models and style
  - Adapts the Power Distribution Unit concepts, schemas and properties
  - Controls – several instances of valves for liquid flow
  - Sensors – New types for pressure, flow rates, etc.
    - Additional “discrete” sensor definitions

Expected Release Timeline

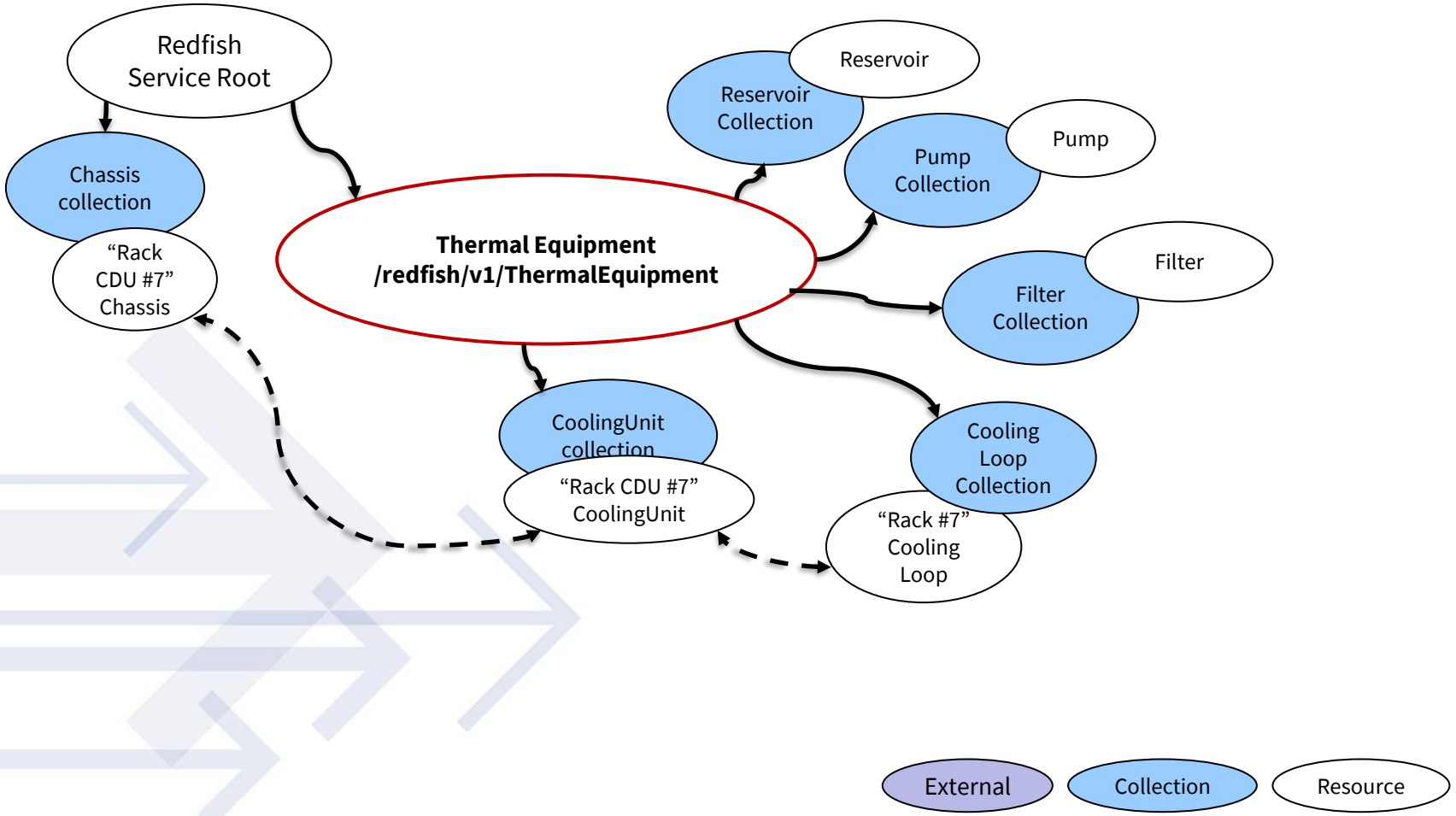
- Work-in-Progress release v0.9
  - Incorporated feedback from alliance partner organizations and others
- Intend to release v1.0 of this work by end of 2022
  - CoolingEquipment, CoolingUnit, CoolingLoop schemas
  - Subsystem schemas: Pump, Reservoir, Filter
  - DiscreteSensor schema
  - PowerEvent and ThermalEvent message registries
- Support in v1.0 expected for:
  - Rack-based or free-standing CDU's
  - Immersion cooling systems
  - Liquid-cooled (self-contained) servers
  - Rear-door heat exchangers
- Expect further additions in the Redfish 2023.1 and future releases

THERMAL EQUIPMENT MODEL

NEW ThermalEquipment resource

- Single resource under **ServiceRoot**
  - Follows design pattern used for **PowerEquipment**
  - Contains links to all cooling systems and related equipment
  - Used primarily for discovery of managed equipment
- Links to Resource Collections of:
  - Cooling Distribution Units (CDU's)
  - Immersion cooling units
  - Air Handler (CRAH) units
  - Air Conditioners (CRAC) units
  - Cooling Loops
    - Both facility-level (FWS) and rack/secondary (TCS) loops
  - Free standing Pumps, Filters, and Reservoirs
    - Equipment not included within a CDU
  - Other cooling equipment?

Thermal Equipment Model

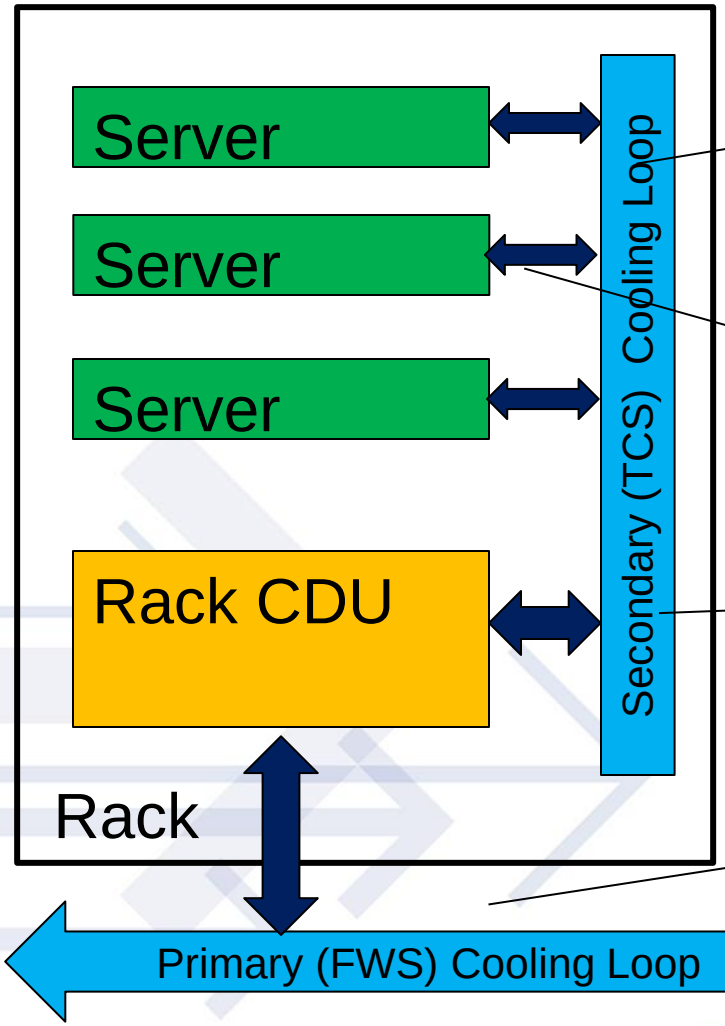


COOLING LOOP MODEL

Cooling Loop Model

- **CoolingLoop** model
  - Describes the physical characteristics and capacities of a cooling loop
  - Loop can be self-contained (within a rack or group of racks)
    - Or can be facility-wide (primary loops from external chillers, etc.)
  - Shows connectivity to equipment
    - Provides means for both “names” (strings) and links to resources
- **CoolingConnection** models connections to a **CoolingLoop**
  - Models the “supply” and “return” side of the managed equipment
    - An instance is either a connection pair, or an individual supply or return
  - Metrics are gathered at these connection points
    - Allows independent metrics for each piece of equipment connected to the loop
  - Provide information about the connected loop if available
    - User-entered “loop name” provides a connection path through the infrastructure

Cooling Loop – Rack-level self-contained example



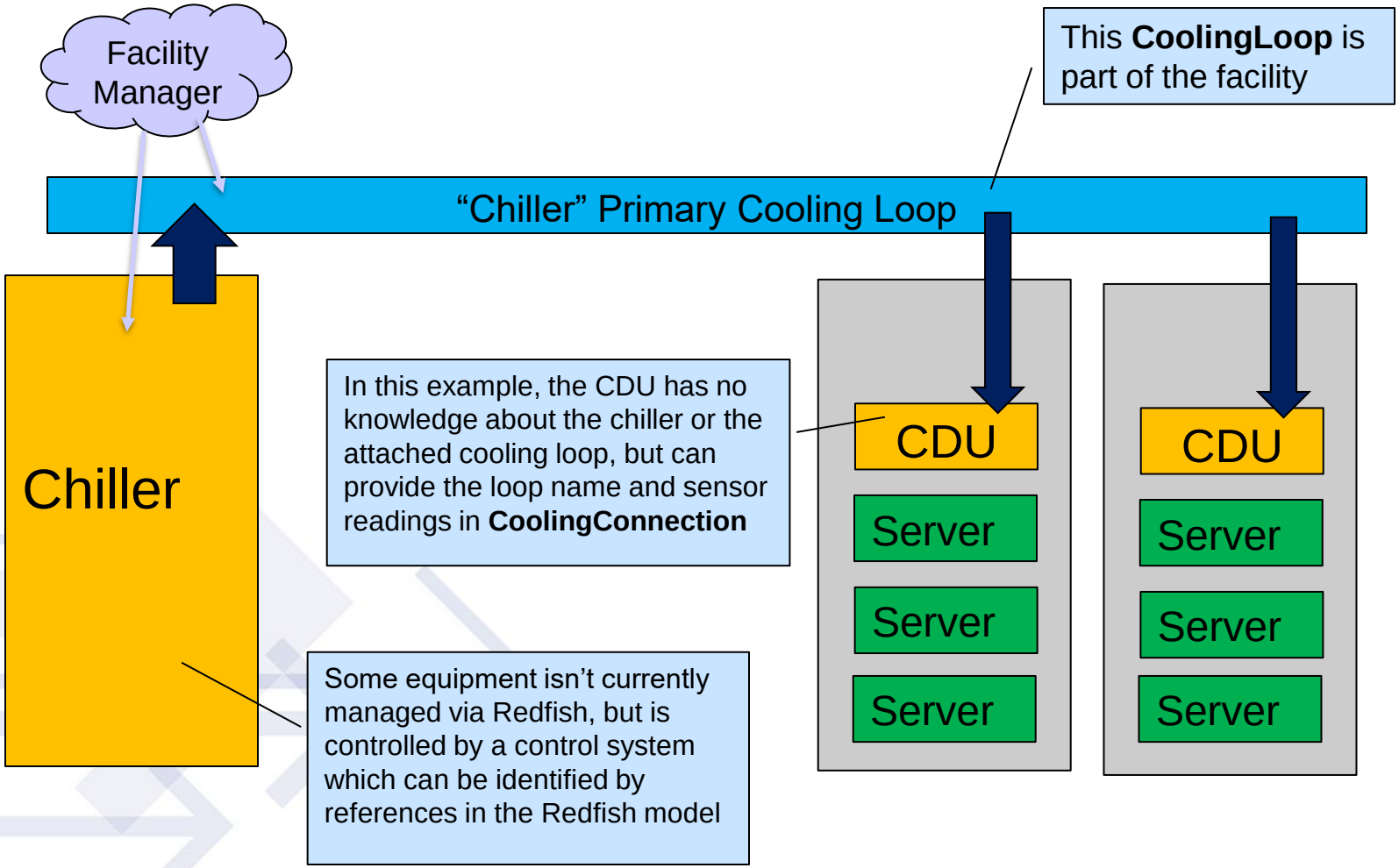
This **CoolingLoop** is within the rack, and can be modeled and populated by the RackCDU's Redfish service

System's **CoolingConnection** links to the external **CoolingLoop** (in the Rack CDU)

Rack CDU *secondary* **CoolingConnection** connects to a **CoolingLoop** resource

Rack CDU *primary* **CoolingConnection** links to a **CoolingLoop** resource in the facility, or provides just the *LoopName*

Cooling Loop – facility level example



NEW CoolingLoop schema

- **CoolingLoopCollection** placed under **ThermalEquipment**
- Reports product, location, and capacity for the loop
  - *CoolingLoopType* – Condenser, Facility, Technology, Immersion, Internal
- Describes the coolant properties
  - *FluidType* – Water, GlycolMixture, Dielectric
  - *FluidQuality* – Normal or Abnormal
  - *FluidLevelStatus* – OK, Warning, Critical
- Methods to represent connections to related equipment
  - *ConsumingEquipmentNames[]* – User-defined string for unmanaged gear
  - *ConsumingEquipment[]* – R/W array of links to **Chassis** resources
  - *CoolingManagerUri* – User-defined link to a management console

NEW CoolingLoop resource


```
{
    "@odata.type": "#CoolingLoop.v1_0_0.CoolingLoop",
    "Id": "BuildingChiller",
    "Name": "Feed from building chiller",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "CoolingLoopType": "Facility",
    "UserLabel": "Building Chiller",
    "FluidType": "Water",
    "FluidLevelStatus": "OK",
    "FluidQuality": "Normal",
    "FluidSpecificHeatJoulesPerKgK": 4184.0,
    "FluidDensityKgPerLiter": 1.0,
    "FluidLevelPercent": {
        "Reading": 95
    },
    "HeatRemovedkW": {
        "Reading": 473.4
    },
    "SupplyEquipmentNames": ["Chiller"],
    "ConsumingEquipmentNames": ["Rack #1 CDU", "Rack #2 CDU", "Rack #3 CDU", "Rack #4 CDU"],
    "Links": {
        "ConsumingEquipment": [{
            "@odata.id": "/redfish/v1/ThermalEquipment/CDUs/1"
        }]
    },  << TRUNCATED >>
```


Details about fluid used in the loop

Sensor excerpts for fluid level and total heat removed

EquipmentNames allow users to manually add non-Redfish devices to help complete the model

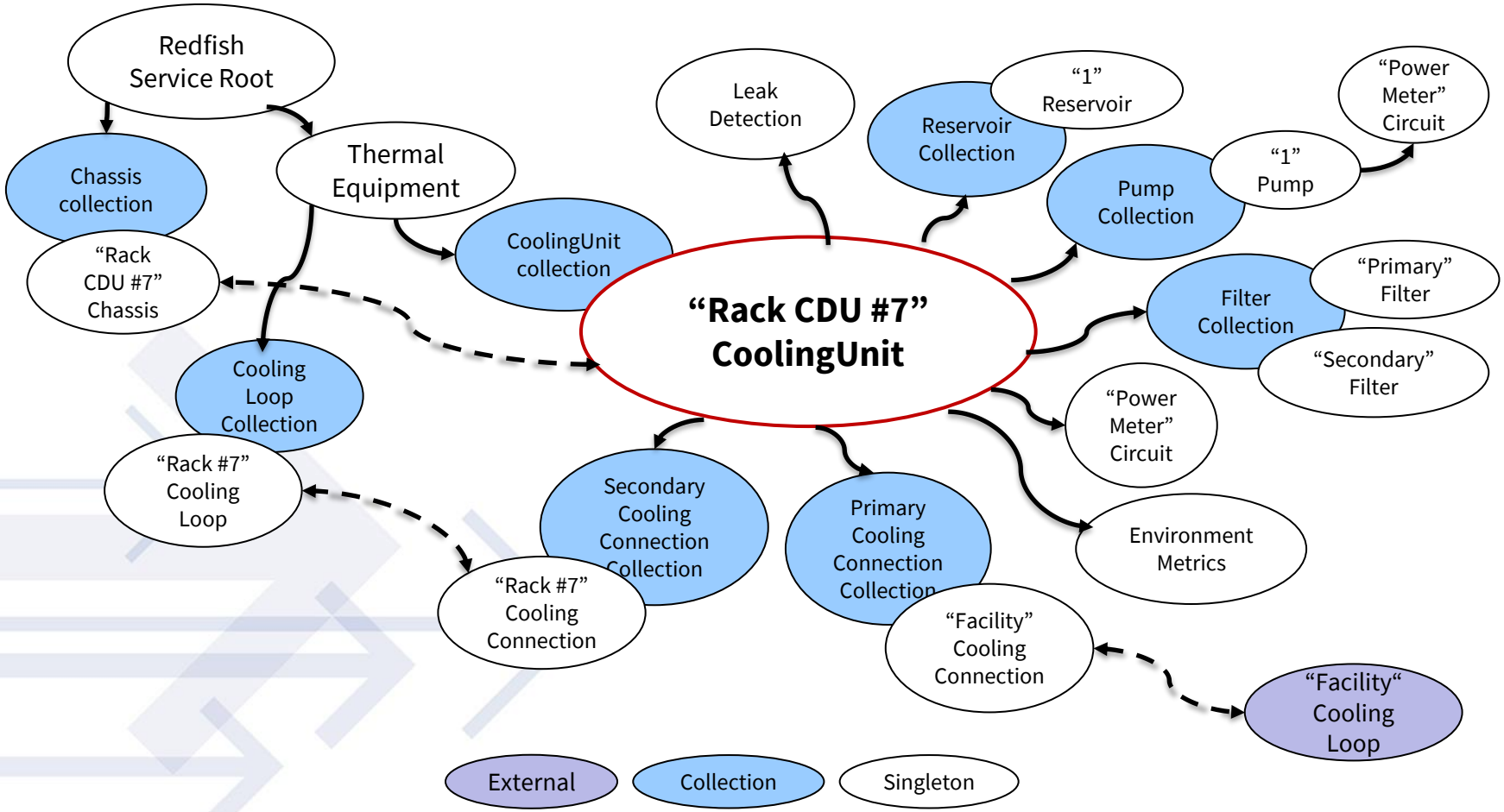
Links to Redfish-managed *Consuming* and *Source* resources

COOLING UNIT MODEL

NEW CoolingUnit schema and resources

- Unified schema covers many types of cooling gear
  - Equipment that cannot be modeled by a **Chassis** and **ThermalSubsystem**
    - Heat exchangers and manifolds expected to be covered as a Chassis
    - CoolingUnit equipment will have a containing Chassis resource
  - Share common modeling and property definitions
  - *EquipmentType* property provides specific identification
- Resource contents
  - General product identification – model, manufacturer, serial number, etc.
  - Versioning – Hardware revision, firmware version, date of manufacture
- *Links* to subordinate and related resources
  - Sensors, DiscreteSensors, Metrics (entire unit)
  - Primary (input) and Secondary (output) CoolingConnections
  - Subsystems: Pumps, Filters, Reservoirs
  - Chassis that contains the equipment

Cooling Unit Model



NEW CoolingUnit schema

```
{
    "@odata.type": "#CoolingUnit.v1_0_0.CoolingUnit",
    "Id": "1",
    "EquipmentType": "CDU",
    "Name": "Rack #4 Cooling Distribution Unit",
    "FirmwareVersion": "3.2.0",
    "Version": "1.03b",
    "ProductionDate": "2020-12-24T08:00:00Z",
    "Manufacturer": "Contoso",
    "Model": "BRRR4000",
    "SerialNumber": "29347ZT536",
    "PartNumber": "ICE-9",
    "UUID": "32354641-4135-4332-4a35-313735303734",
    "AssetTag": "PDX5-92381",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "PrimaryCoolingConnections": { “@odata.id”: < Link to CoolingConnectionCollection > },
    "SecondaryCoolingConnections": { “@odata.id”: < Link to CoolingConnectionCollection > },
    "Pumps": { “@odata.id”: < Link to PumpCollection > },
    "Filters": { “@odata.id”: < Link to FilterCollection > },
    "EnvironmentMetrics": { “@odata.id”: < Link to EnvironmentMetrics > },
    “PowerMeter”: { “@odata.id”: < Link to Circuit > },
    "Sensors": { “@odata.id”: < Link to SensorCollection > },
    “Controls”: { “@odata.id”: < Link to ControlCollection > },
    < TRUNCATED >
```

NEW CoolingConnection schema

- The connection between the cooling unit and a CoolingLoop resource
  - Modeled either a connection pair, or an individual “supply” or “return”
  - Provides numerous sensor readings and controls
    - Flow, Temperature, Pressure on both supply and return
    - Valve controls
  - If known, provide link to **CoolingLoop**
    - Or the loop name and Manager URI if known and populated by end user
- Main monitoring resource for the cooling unit’s functionality
  - Primary cooling connections – input from facility chillers or other sources
  - Secondary cooling connections– output from the cooling unit to feed “consuming” equipment

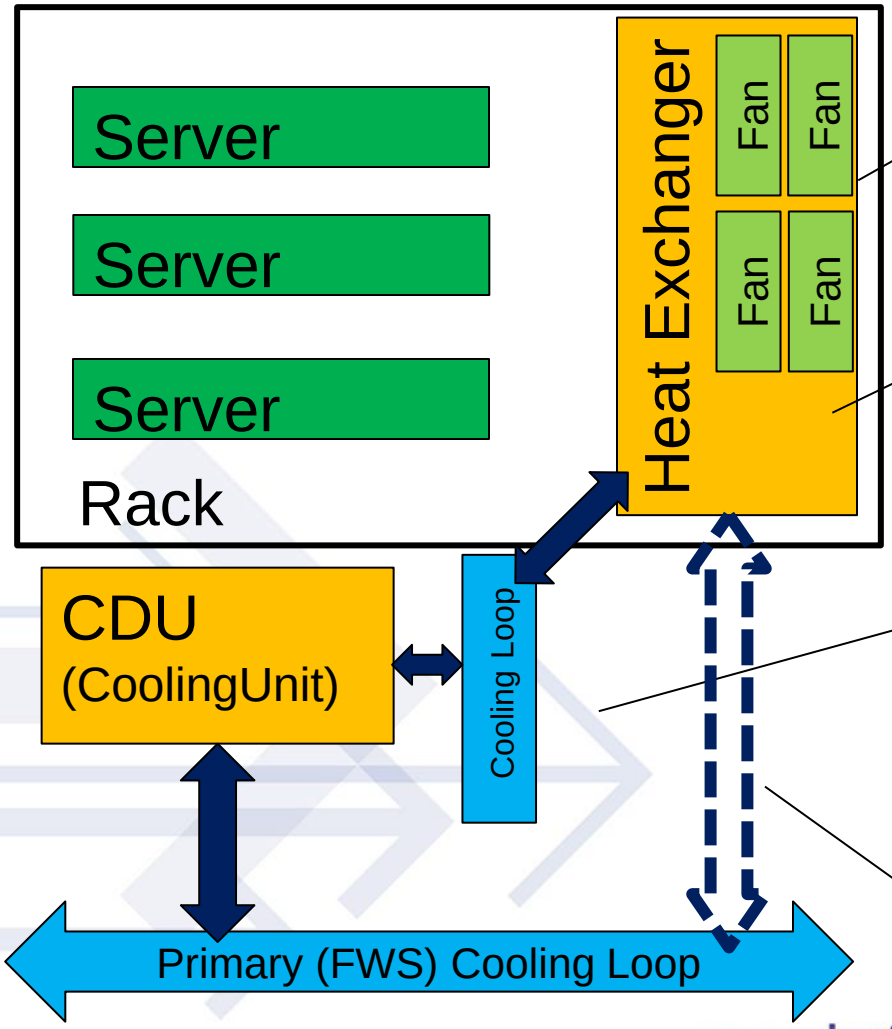
NEW CoolingConnection schema

```
{
    "@odata.type": "#CoolingConnection.v1_0_0.CoolingConnection",
    "Id": "Chiller",
    "Name": “Primary Input from Chiller",
    "Status": {
        "Health": "OK"
    },
    "CoolingConnectionType": “Primary",
    “CoolingLoopType”: “Facility”
    “FluidType”: “GlycolMixture”,
    “GlycolPercent”: 20,
    "RatedFlowLSeconds": 30,
    "SupplyFlowValve": {
        "DataSourceUri": "/redfish/v1/CoolingEquipment/RackCDUs/1/Controls/ChillerSupplyValve",
        "SetPoint": 70,
        "SetPointUnits": "%",
        "Reading": 9.5,
        "ReadingUnits": “L/s"
    },
    "SupplyTemperatureCelsius": {
        "DataSourceUri": "/redfish/v1/CoolingEquipment/RackCDUs/1/Sensors/LoopASupplyTemp",
        "Reading": 14.8
    },
   “SupplyPressurePa": {
        "Reading": 319.6
    },
    "ReturnTemperatureCelsius":  < SENSOR EXCERPT >
    < TRUNCATED >
```

As *CoolingConnectionType* is really a description of the loop – this may become a more general descriptor to indicate if this resource models a “pair” of supply/return connections or a single (typically facility-scale) supply OR return connection.

Sensor excerpts and *Control* excerpt for valves

Example: Rear Door Heat Exchanger – CDU and Chassis



The heat exchanger is modeled as a **Chassis** with a **ThermalSubsystem** that describes the **Fans**

ThermalSubsystem may need additional sensors to describe airflow (TBD)

CDU *primary* **CoolingConnection** attaches to the facility cooling loop and its *secondary* **CoolingConnection** connects to a technology loop, which in turn connects to the heat exchanger via **CoolingConnection**

OR the heat exchanger directly connects to the facility loop via a **CoolingConnection**

Cooling Loops / Cooling Connections – open questions

- Support for Air / Phase Change / Liquid loops
  - Phase change (refrigerant) loops can be modeled with this pattern as well
    - Not recommended for initial release, wait for industry feedback
  - LoopType = Air, Liquid, Primary / Secondary?
    - How best to differentiate primary / secondary connections
      - Currently in separate collections (and focused on liquid coolant)
      - Added CoolingLoopType to CoolingConnection to show “where the connection goes”
    - An “air loop” could be the room, plenum, or ductwork
    - Air handling may not be ‘contained’, but the “loop interface” may still apply
- Chassis links and physical containment
  - Need to validate that model supports adequate and consistent links for physical containment of cooling units and cooling loops
  - Support any mix of managed / unmanaged gear while minimizing the types of links and resources

NEW PowerMeter resource

- Immersion cooling units have additional power-related requirements
  - Ability to report the power consumption of all “immersed” equipment
  - Ability to control the power state of all immersed equipment
    - Both from a breaker (over current) and user-actuated control perspective
- Leverage existing **Circuit** schema for this purpose
  - Add new *CircuitType* of “PowerMeter”
  - Allows a **Circuit** resource to appear under **CoolingUnit**
  - Enables use of *PowerControl* and *BreakerControl* actions
- Power consumption and other monitoring of the cooling equipment itself is shown in **EnvironmentMetrics** or **PowerSubsystem** under the **Chassis** related to the **CoolingUnit**
  - Would like to reduce the number of choices here for interoperability
  - Use of **EnvironmentMetrics** on the **Chassis** may be a consistent answer

Circuit schema as PowerMeter example

```
{
   "@odata.type": "#Circuit.v1_6_0.Circuit",
   "Id": “PowerMeter",
   “Name": “Pump #1 Power Meter",
   "Status": { < Status object> },
   "CircuitType": “PowerMeter",
   "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 > },
   “Actions": { < ResetBreaker, ResetStatistics > }
   "@odata.id": "/redfish/v1/CoolingEquipment/RackCDUs/1/Pumps/1/PowerMonitor",
}
```

NEW Subsystems for CoolingUnit and ThermalEquipment

- Equipment that may appear as a subsystem or component of a CoolingUnit, or may be a free-standing device
  - Model allows for this equipment to reside under ThermalEquipment, or as subordinate resources to an individual CoolingUnit
- For release v0.9, these schema contain only basic inventory and identification data
  - Expect to add more specific properties as feedback is received
  - But even the basic part and product information is useful to customers
- **Pump** Resource Collection
  - Will have differential pressure / absolute pressure, flow, etc.
  - Variable Frequency Drive may need an object
  - *PowerMeter* (Circuit) subordinate resource
    - May be 3-phase, have a breaker, etc.

NEW Subsystem schemas, continued

- **Reservoir** Resource Collection
  - Fill level, pressure sensors
  - Air bleed value (controls), fill valve, drain valve
  - May have connections between reservoirs (balancing)
- **Filter** Resource Collection
  - Pressure sensors
  - Service time / install time, life etc.
  - ASHRAE requirements / classifications
  - Flush / clean actions?

DISCRETE SENSORS AND LEAK DETECTION

NEW LeakDetection schema

- Resource to describe leak detection equipment and report leaks
  - Allows discovery of detection equipment to validate customer requirements
- *LeakDetectorGroups* supports multiple “zones” of detection
  - Each group represents a detection zone
  - Made up of one or more *LeakDetection* sensors
  - Can also include a humidity sensor
  - “Policy” for what constitutes a reported leak is left to implementation
    - Assumes this is manufacturer or configuration based, not user-defined
- *Status* object provides means to report leaks
  - Will define messages for reporting leaks as *Conditions*

LeakDetection example


```
{
    "@odata.type": "#LeakDetection.v1_0_0.LeakDetection",
    "Name": "Leak Detection Systems",
    "Status": {
        "State": "Enabled",
        "Health": "OK",
        "Conditions": []
    },
    "LeakDetectorGroups": [{
        "GroupName": "Detectors under and around the CDU",
        "HumidityPercent": {
            "Reading": 45
        },
        "DiscreteDetectors": [{
                "DataSourceURI": "/redfish/v1/ThermalEquipment/CDUs/1/DiscreteSensors/LeakDetection",
                "DeviceName": "Moisture-type Leak Detector",
                "DiscreteState": "OK"
            },
            {
                "DataSourceURI": "/redfish/v1/ThermalEquipment/CDUs/1/DiscreteSensors/Overflow",
                "DeviceName": "Overflow Float Switch",
                "DiscreteState": "OK"
            }
        ]
    }],
```


In this example with one LeakDetectorGroup, there are three sensors, with the implementation deciding the policy under which a leak is reported

Humidity reading, with an internal threshold to indicate a leak

Two types of DiscreteSensor, which will indicate a leak with a DiscreteState of “Alert”

NEW Discrete Sensor model

- Leak detection devices tend to be discrete / state-based devices
- There are other cases for sensors which are just “dry contact” inputs
  - Many of these should be modeled as properties with enumerations
    - e.g. “DoorState”: “Open” / “Closed”/ “Locked” is the preferred definition compared to “DoorOpen” boolean
  - But many trigger events on state changes, have product information, and other metadata to include in the model
- Propose leveraging the **Sensor** model to create **DiscreteSensor**
  - *DiscreteState* is the primary property
    - Supports an enumerated list of possible states
      - “OK”, “Alert” are the first proposed states.
      - “Disconnected” may be added to show sensor wire has been disconnected
  - *SupportedDiscreteStates* provides the list of supported states
  - *DiscreteThresholds* provides reactions to *DiscreteStates values*

DiscreteSensor schema as Leak Detector example

```
{
    "@odata.type": "#DiscreteSensor.v1_0_0.DiscreteSensor",
    "Id": "LeakDetector",
    "Name": "Moisture-type Leak Detector",
    "DiscreteSensorType": "Moisture",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "DiscreteState": "OK",
    "SupportedStates": [ "OK", "Alert" ],
    "PartNumber": "3493-A44",
    "Manufacturer": "Contoso Water Detection Systems",
    "Location": {
        "PartLocation": {
            "Reference": "Bottom",
            "ServiceLabel": "Leak Detector"
        }
    },
    "DiscreteThresholds": {
        "Caution": {
            "DiscreteStates": [ "Alert" ],
            "DwellTime": "PT1M"
        }
    },
```

DiscreteState is analogous to Reading in a Sensor. *SupportedStates* shows what states may be reported by this sensor

DiscreteThresholds shows the mapping of state values to reported Status or Conditions.

QUESTIONS FOR INDUSTRY

Status and open topics

- Model is solidifying with several name changes since last release
  - Expect more property additions prior to v1.0, depending on feedback
- Expect support for air-liquid cooling units
  - Reviewing model to ensure this support can be added
  - Likely finalized after the v1.0 release of this material (add to next release)
- Significant number of common messages to define for Events / Alarms
  - Expect to define new message registries
    - Should be able to harvest existing SNMP trap definitions as a starting point
  - Will be prioritizing this portion of the effort to enable products to be fully managed using Redfish (and without requiring SNMP “in practice”)

Q&A & Discussion