Redfish OpenAPI Support (Work-In-Progress)

Mike Raineri (Dell), Redfish Forum Co-Chair

June 2018

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.

What Is OpenAPI?

- OpenAPI is a framework for defining RESTful APIs
- Implementers can create a RESTful API definition in a YAML or JSON file
  - The OpenAPI specification details the formatting supported
  - For the purpose of this discussion, we'll focus on the YAML file since most of the examples and editors available appear to be YAML based
- The OpenAPI community has created tools for taking YAML files and auto-generating code libraries for various languages

Elements to Define in the YAML File(s)

- Prescriptive URIs for all Redfish resources
  - All URIs are well known by the client; no dynamic discovery via walking from Service Root needed
  - Possible to describe a parameter in a URI (such as a Chassis ID)
- URI descriptors contain possible HTTP methods and their responses
  - HTTP status codes and payload definitions
- Annotations for elements defined by the YAML file
  - Can map many of the standard terms to our existing annotations
  - Custom terms can be made by starting with “x-”
- OpenAPI allows for the service document to make references to external files using “$ref”

YAML Sample: Header

```
info:
  contact:
    name: DMTF
    url: https://www.dmtf.org/standards/redfish
  description: This contains the definition of a Redfish service.
  title: Redfish API
  version: '2018.2'
openapi: 3.0.0
```

YAML Sample: Paths

```
paths:
  /redfish/v1/:
    get:
      responses:
        '200':
          content:
            application/json:
              schema:
                $ref: http://redfish.dmtf.org/schemas/v1/ServiceRoot.v1_3_1.yaml#/components/schemas/ServiceRoot
          description: Resource response
        default:
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RedfishError'
          description: Error condition
    head:
      responses:
        '204':
          description: Success, but no response data
        default:
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RedfishError'
          description: Error condition
```

YAML Sample: Paths (cont.)

```
/redfish/v1/SessionService/Sessions/{SessionId}:
  get:
    parameters:
    - in: path
      name: SessionId
      required: true
      schema:
        type: string
    responses:
      '200':
        content:
          application/json:
            schema:
              $ref: http://redfish.dmtf.org/schemas/v1/Session.v1_1_0.yaml#/components/schemas/Session
        description: Resource response
      default:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/RedfishError'
        description: Error condition
```

YAML Sample: Schemas

```
components:
  schemas:
    Session:
      additionalProperties: false
      description: The Session resource describes a single connection (session) between
        a client and a Redfish service instance.
      properties:
        Id:
          $ref: http://redfish.dmtf.org/schemas/v1/Resource.yaml#/components/schemas/Id
          readOnly: true
        Name:
          $ref: http://redfish.dmtf.org/schemas/v1/Resource.yaml#/components/schemas/Name
          readOnly: true
        UserName:
          description: The UserName for the account for this session.
          nullable: true
          readOnly: true
          type: string
          x-longDescription: The value of this property shall be the UserName that
            matches a registered account identified by a ManagerAccount resource registered
            with the Account Service.
      required:
      - Id
      - Name
      type: object
```

Additions to Redfish for Support of OpenAPI

- For payload definitions, OpenAPI is heavily based on JSON Schema
  - There are some minor deviations
  - Our existing JSON Schema files are about 95% of the way to how OpenAPI defines the “components/schemas” body of their documents
- Create a lightweight service document that lists out the URIs and points to the latest version of each of the schemas
  - This will be regenerated on each release of the Redfish schema bundle
- Use annotations in the CSDL and JSON Schema definitions for assisting with the generation of the OpenAPI service document
  - Redfish.Uris (new): A collection of strings that contain the valid URI patterns for the resource
  - Capabilities.UpdateRestrictions: If “PATCH” or “PUT” are allowed methods
  - Capabilities.DeleteRestrictions: If “DELETE” is an allowed method
  - Capabilities.InsertRestrictions: If “POST” is an allowed method

Annotation Example for CSDL

```
<Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="ManagerAccount">
     <Annotation Term="Redfish.OwningEntity" String="DMTF"/>

     <EntityType Name="ManagerAccount" BaseType="Resource.v1_0_0.Resource" Abstract="true">
       <Annotation Term="Capabilities.InsertRestrictions">
         <Record>
           <PropertyValue Property="Insertable" Bool="false"/>
         </Record>
       </Annotation>
       <Annotation Term="Capabilities.UpdateRestrictions">
         <Record>
           <PropertyValue Property="Updatable" Bool="true"/>
         </Record>
       </Annotation>
       <Annotation Term="Capabilities.DeleteRestrictions">
         <Record>
           <PropertyValue Property="Deletable" Bool="true"/>
         </Record>
       </Annotation>
       <Annotation Term="Redfish.Uris">
         <Collection>
           <String>/redfish/v1/AccountService/Accounts/{AccountId}</String>
         </Collection>
       </Annotation>
     </EntityType>
```

Annotation Example for JSON Schema

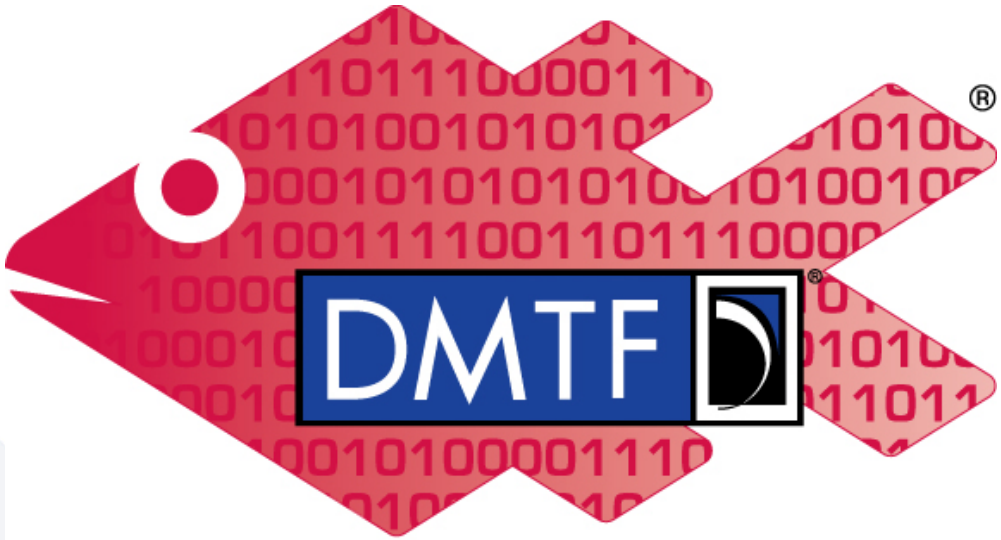
```
{
    "$ref": "#/definitions/ManagerAccount",
    "$schema": "http://redfish.dmtf.org/schemas/v1/redfish-schema.v1_5_0.json",
    "copyright": "Copyright 2014-2018 Distributed Management Task Force, Inc. (DMTF). For the full DMTF
copyright policy, see http://www.dmtf.org/about/policies/copyright",
    "definitions": {
        "ManagerAccount": {
            "anyOf": [ ... ],
            "deletable": true,
            "description": "Account description...",
            "insertable": false,
            "longDescription": "Account long description...",
            "updatable": true,
            "uris": [
                "/redfish/v1/AccountService/Accounts/{AccountId}"
            ]
        }
    },
    "owningEntity": "DMTF",
    "title": "#ManagerAccount.ManagerAccount"
}
```

OpenAPI Generation Tool

- First step: perform a one to one conversion process of each JSON Schema file to create the standalone YAML files
  - “definitions” becomes “components/schemas”
  - “longDescription” becomes “x-longDescription”
  - “readonly” becomes “readOnly”
  - Etc
- Second step: generate the OpenAPI service document after all files are created
  - Scan each of the converted files for the “uris”, “insertable”, “updateable”, and “deleteable” terms
  - Based on the discovered terms, populate the paths portion of the document, fill in the appropriate methods, and point to the appropriate schema files
  - Scan each of the converted files for all action definitions, and generate the path for the actions

Implications to the Redfish Specification

- These changes will introduce new requirements on implementations
  - Implementations will be required to support the URIs specified within the schema files
- In order to allow shipping implementations to continue operating, a new minor version of the specification will be produced to differentiate the cut-over point
  - Proposed version 1.6.0 of the specification should add the normative language about supporting the URIs specified in the schema files
  - Existing implementations can report 1.5.X (or older) until they are ready to be updated to conform to the new URI patterns
  - This would be considered a backwards compatible change since clients do not need to modify their software to move forward



Redfish