Document Identifier: DSP2055

Date: 2019-08-06

Version: 1.0.0

Quick Start for Authoring Redfish Schema

Supersedes: none

Document Class: Informative

Document Status: Published

Document Language: en-US

CONTENTS

Foreword

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

The DMTF acknowledges this individuals for their contributions to the Redfish standard, including this document and Redfish Schemas, interoperability profiles, and message registries:

Editor:

- Bill Scherer - Hewlett Packard Enterprise

Contributors:

- Jeff Autor - Hewlett Packard Enterprise
- Mike Garrett - Hewlett Packard Enterprise
- Jeff Hilland - Hewlett Packard Enterprise
- Corey Morrison - Hewlett Packard Enterprise
- Michael Raineri - Dell Inc.

1. Abstract

This brief and easy-to-use guide helps novice Redfish developers quickly create a Redfish Schema.

Rather than exhaustively cover every aspect of the Redfish Schema, this guide covers 90% of the schema data that is required to get started with schema creation.

Anyone can use this guide to create schema for the first time without wading through the technical documentation to become an expert in the more unusual schema definition components.

2. References

For questions and information that this guide does not cover, see the published DMTF schemas examples.

For details, see these technical specifications:

- DMTF DSP0266, Redfish Scalable Platforms Management API Specification 1.6, 23 August 2018
- ECMAScript® Language Specification, 5.1 Edition, June 2011
- OData Version 4.01. Part 1: Protocol, 30 January 2018
- OData Version 4.01. Part 2: URL Conventions, 30 January 2018
- OData Version 4.01. Part 3: Common Schema Definition Language (CSDL), 22 June 2017
- The OpenAPI Specification

3. Schema creation: Before you begin

Before you can create the schema, you must:

	Step	Summary
1.	Understand the schema formats.	Redfish Schemas are defined in the OpenAPI, JSON Schema, and CSDL formats. You must create your schema in the CSDL format.
2.	Understand the schema data types.	Redfish Schema support the string, numeric, boolean, array, enumeration, and complex types.
3.	Gather your data and its information.	Gather your data and its information, such as the names, descriptions, constraints, types, and

	Step	Summary
		permissions.
4.	Organize the data.	Organize your data into a hierarchical structure that you use as a skeleton for your schema.

3.1. Understand the schema formats

Redfish Schemas are defined in the OpenAPI, JSON Schema, and CSDL formats, which essentially contain the same information in different formats.

You must develop a schema in the CSDL format because:

- The Redfish Forum maintains a CSDL-to-JSON conversion tool that automatically builds JSON Schemas from CSDL sources. You can use this tool to:
  - Maintain parallel versions of schemas.
  - Sanity check the CSDL schemas.
  - Provide the input JSON Schema for a JSON Schema to OpenAPI tool, which is also maintained by the Redfish Forum for creating OpenAPI definitions.
- To track the version history of schema revisions, you can use either:
  - A single CSDL schema file. Developers append new version information to the single CSDL file as needed..
  - Multiple JSON Schema files. A separate schema is required for each JSON Schema file revision. You must also modify an unversioned JSON Schema file to add each revision.
- For embedded device-supported schemas, the Redfish Device Enablement (RDE) dictionary builder requires CSDL format as input.

3.2. Understand the schema data types

These subsections describe the schema data types:

- Strings
- Numerics
- Booleans
- Arrays
- Enumerations
- Complex types

3.2.1. Strings

To define a string property, use the `Edm.String` type.

This example defines the dog's owner's name as a string property:

```
<Property Name="Owner" Type="Edm.String">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/ReadWrite"/>
  <Annotation Term="OData.Description" String="The dog's owner's name."/>
  <Annotation Term="OData.LongDescription" String="This property shall be the name of
the dog's owner."/>
</Property>
```

3.2.2. Numerics

To define a numeric property, use the `Edm.Int64` or `Edm.Decimal` types. The `Edm.Int64` type should be used for counts or properties that will always contain an integer value, while `Edm.Decimal` should be used in cases where a floating point value could appear.

This example defines the year that the dog was vaccinated as an integer property:

```
<Property Name="YearApplied" Type="Edm.Int64" Nullable="false">
  <Annotation Term="OData.Description" String="The year this dog was vaccinated."/>
  <Annotation Term="OData.LongDescription" String="This property shall contain the
year this dog was vaccinated."/>
</Property>
```

3.2.3. Booleans

To define a boolean property, use the `Edm.Boolean` type.

This example defines the dog's spayed or neutered status as a boolean property:

```
<Property Name="SpayedOrNeutered" Type="Edm.Boolean">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/ReadWrite"/>
  <Annotation Term="OData.Description" String="An indication of whether the dog has
been spayed or neutered.  If true, the dog has been spayed or neutered."/>
  <Annotation Term="OData.LongDescription" String="This property shall indicate
whether the dog has been spayed or neutered.  If true, the dog has been spayed or
neutered."/>
</Property>
```

For boolean property descriptions, be explicit about what `true` and `false` values mean.

3.2.4. Arrays

To define an array of a specific type, use the `Collection(type)` type. The *type* can be any valid type, including enumeration and the complex type.

This example defines the dog breeds as an array of string values:

```
<Property Name="Breeds" Type="Collection(Edm.String)">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
  <Annotation Term="OData.Description" String="The dog's breed."/>
  <Annotation Term="OData.LongDescription" String="This property shall be a name from
the international registry of dog breeds."/>
</Property>
```

3.2.5. Enumerations

An enumeration is comprised of the enumerated type and its members and the property that references the enumerated type.

3.2.5.1. The enumerated type and its members

An enumerated type defines its name and members.

To define an enumerated type, use the `EnumType` element. Define that element inside the `Schema` element but outside the `EntityType` element for the schema properties.

Define the members of the enumerated type in the `EnumType` element.

This example defines the `PrimaryColor` enumerated type and its members:

```
<EnumType Name="PrimaryColor">
  <Member Name="Black">
    <Annotation Term="OData.Description" String="A black dog."/>
  </Member>
  <Member Name="Brown">
    <Annotation Term="OData.Description" String="A brown dog."/>
  </Member>
  <Member Name="Yellow">
    <Annotation Term="OData.Description" String="A yellow dog."/>
  </Member>
  <Member Name="Red">
    <Annotation Term="OData.Description" String="A red dog."/>
  </Member>
```

```
  <Member Name="White">
    <Annotation Term="OData.Description" String="A white dog."/>
  </Member>
</EnumType>
```

The `EnumType` element contains these elements:

Element	Description
`<Member Name="Yellow">`	A member of the enumerated type. The `Name` attribute defines the name of the member. In this case, the name is `Yellow`.
`<Annotation Term="OData.Description"/>`	The short description for an enumerated type's member.
`<Annotation Term="OData.LongDescription"/>`	The normative, or long, description for an enumerated type's member.

As a reminder, enumerated type member names cannot contain spaces.

3.2.5.2. The property that references the enumerated type

A property defines its name and enumerated type attributes, and its short and long description elements.

This example defines a property that references the `Dog.v1_0_0.PrimaryColor` enumerated type for the dog's primary color:

```
<Property Name="PrimaryColor" Type="Dog.v1_0_0.PrimaryColor">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
  <Annotation Term="OData.Description" String="The main color of the dog."/>
  <Annotation Term="OData.LongDescription" String="This property shall be the main
color of the dog."/>
</Property>
```

The `Property` element contains these attributes:

Attribute	Description
`Name="PrimaryColor"`	The name of the property.
`Type="Dog.v1_0_0.PrimaryColor"`	The enumerated type. The

Attribute	Description
	`Dog.v1_0_0.PrimaryColor` enumerated type contains these components: • `Dog`. The schema namespace that defines the enumerated type. • `v1_0_0`. The schema version that defines the enumerated type. For more information about version support, see Multi-version support in Advanced topics. • `PrimaryColor`. The name of the enumerated type. The enumerated type name can but does not have to match the name of the property that references this type. These names do not have to match if you want to multiple properties to reference the same enumerated type.

The `Property`element also contains these elements:

Element	Description
`<Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>`	The permissions for the enumerated type's members.
`<Annotation Term="OData.Description"/>`	The short description for the property.
`<Annotation Term="OData.LongDescription"/>`	The normative, or long, description for the property.

3.2.6. Complex types

As with enumerations, complex types are represented in schema in two parts. The first part looks identical to the first part for an enumeration, and consists of a property typed to a separately defined type. For example, our array of vaccination records references the `VaccinationRecord` type:

```
<Property Name="VaccinationRecord" Type="Collection(Dog.v1_0_0.VaccinationRecord)"
Nullable="false">
```

```
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/ReadWrite"/>
  <Annotation Term="OData.Description" String="The vaccination records for this dog."/>
  <Annotation Term="OData.LongDescription" String="This property shall be an array of
records for the vaccinations applied to this dog."/>
</Property>
```

`Dog.v1_0_0.VaccinationRecord` is a reference to a formal type that the `Dog` schema version 1.0.0 defines elsewhere.

In this case, the `Collection(Dog.v1_0_0.VaccinationRecord)` type marks `VaccinationRecord` as an array of vaccination records rather than a singleton. As with enumerations, you define the named resource type. In this case `VaccinationRecord` is defined inside the scope of the `<Schema>` block but outside the `Dog.v1_0_0` `<EntityType>` block scope:

```
<ComplexType Name="VaccinationRecord">
  <Annotation Term="OData.AdditionalProperties" Bool="false"/>
  <Annotation Term="OData.Description" String="The vaccination applied to this dog."/>
  <Annotation Term="OData.LongDescription" String="This type shall contain the
vaccination applied to this dog."/>
  <Property Name="VaccinationName" Type="Edm.String" Nullable="false">
    <Annotation Term="OData.Description" String="The name of a vaccination applied to
this dog."/>
    <Annotation Term="OData.LongDescription" String="This property shall contain the
standard medical name for the vaccination applied to this dog."/>
  </Property>
  <Property Name="YearApplied" Type="Edm.Int64" Nullable="false">
    <Annotation Term="OData.Description" String="The year this dog was vaccinated."/>
    <Annotation Term="OData.LongDescription" String="This property shall contain the
year the vaccination was applied to this dog."/>
  </Property>
</ComplexType>
```

The `VaccinationName` and `YearApplied` properties in the complex type have the same structure as they have inside the `<EntityType>` block for main properties.

Complex types in Redfish may contain other complex types and enumerations with no limitation on the number of nesting levels.

3.3. Gather your data and its information

After you determine which data you want to capture in the schema, gather this information for your data:

Information	Description
Name	A short but descriptive camel-case name for the datum. **Camel case:** Each word or abbreviation in the phrase begins with a capital letter with all other letters in lower case and no intervening spaces or punctuation. For units, the Redfish specification requires that the units are appended to the property name. For example, `SpeedRPM` or `CapacityMiB`.
Description	A textual explanation of what the datum represents. Be sure to include any special usage instructions and units for numeric data, where applicable. You need normative information about the datum and information, which a human can read. Redfish follows ISO language conventions.
Type	What kind of values can the datum carry? Redfish supports: • Primitive types: Enumerations, or multiple-choice string values, integers, strings, and booleans. • Complex types: Complex types and primitive types. • Arrays: Complex or primitive types. • References to data in other schemas.
Permissions	Is the datum static, or `OData.Permission/Read`, or configurable, or `OData.Permission/ReadWrite`?
Constraints	For numeric data, the minimum and maximum values. For strings, the minimum and maximum lengths, or a particular pattern, or regular expression, to which the the string should map. Not all data have constraints.

The `Dog` schema is the running example in this quick start.

In this example, dogs have these attributes:

Property	Type	Permissions	Notes
Name	string data	`OData.Permission/ReadWrite`	
Primary color	string data enumeration	`OData.Permission/Read`	

Property	Type	Permissions	Notes
Birth year	numeric	`OData.Permission/Read`	
Spayed or neutered status	boolean	`OData.Permission/ReadWrite`	
Owner	person	`OData.Permission/Read`	
Breeds	array of string data of the breed names	`OData.Permission/Read`	We could treat this as an enumeration, but it would be painful to type in the hundreds of registered breeds!
Vaccination record	array of individual records	`OData.Permission/ReadWrite`	Each record includes a vaccination name and the year it was administered.

3.4. Organize the data

After you gather the raw data, you must determine how to group and sort it.

If you have multiple data items that go together logically, you can create and place them within a complex type that represents them as a set.

Complex types can themselves contain complex types.

Repeat this process until you arrange all the data into a hierarchical form rooted at the name of your schema.

This organization is the skeleton that your schema will represent.

The organization of the `Dog` schema is:

- `Dog`
    - `Name`
    - `PrimaryColor`
    - `BirthYear`
    - `SpayedOrNeutered`
    - `Owner`
    - `Breeds` array

        - `VaccinationRecord` array
            - `VaccinationName`
            - `YearApplied`

You should create a mockup of your data. To check schema into the Redfish tree, you need both the schema and its mockup. Mockups consist of an example JSON payload for the data that follows your schema.

To test our schema, we use this mockup:

```
{
    "@odata.type": "Dog.v1_0_0.Dog",
    "Name": "Torchwood",
    "PrimaryColor": "White",
    "BirthYear": "2006",
    "SpayedOrNeutered": true,
    "Owner": "Marie",
    "Breeds": [
        "Jack Russell Terrier",
        "Boston Terrier"
    ],
    "VaccinationRecord": [
        {
            "VaccinationName": "Rabies",
            "YearApplied": "2015"
        },
        {
            "VaccinationName": "Distemper",
            "YearApplied": "2015"
        }
    ]
}
```

The continuous integration (CI) tools in the DMTF GitHub validate your schema against your mockup.

4. Create the schema

This section provides step-by-step instructions for building the schema:

- Step 1: Name the schema
- Step 2: Create the skeleton
- Step 3: Add OpenAPI paths
- Step 4: Add properties

4.1. Step 1: Name the schema

Name your CSDL schema file *Schema_name*_v1.xml, where *Schema_name* matches the Namespace name in your schema.

The example schema file is named Dog_v1.xml because its namespace is Dog:

```
<Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Dog">
```

As with any XML file, your schema file is text-based. Standard XML rules for character-set encoding apply. The example in this guide is UTF-8.

Redfish follows these naming conventions:

- Do not use a plural name in the file name unless the corresponding schema represents multiple items.
- To define a collection, append Collection to the data type reference name.

This example includes the Dog and DogCollection schema files. You would not name the dog collection schema file, Pack, as much as it may seem appropriate.

4.2. Step 2: Create the skeleton

A fair bit of the schema file consists of standard boilerplate. Copy the template from CSDL schema template and save it to your file. Make sure to replace all *SchemaName* instances with the name of your schema. If you read through the skeleton, it contains comments that document each of its parts.

4.3. Step 3: Add OpenAPI paths

Just inside the <EntityType> block of the schema, a section adds OpenAPI paths for your resource.

Generally, containment defines the paths:

A child resource inherits its parent's OpenAPI path. The child resource's OpenAPI path has the name of the parent's OpenAPI path appended with your schema's name.

Do not forget that collections are generally pluralized.

4.4. Step 4: Add properties

Each property that you add to your schema consists of a <Property> block in a schema section.

```
<Property Name="SampleProperty" Type="Edm.String" Nullable="false">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
  <Annotation Term="OData.Description" String="A sample property."/>
  <Annotation Term="OData.LongDescription" String="This property shall contain a
longer description of the sample property."/>
</Property>
```

This block defines these items:

Item	Description
`<Property>`	The block of data that introduces the property.
`Name`	The name by which to reference the property from Redfish. The previous example shows the `SampleProperty` property.
`Type`	The OData type for the property. The subsections discuss the most common types.
`Nullable`	An indication of whether a property is mandatory or optional. • If `false`, the property is required. • If `true` or this qualifier is omitted, the property is optional. In Redfish, the convention is that unless a strong reason exists for a property to be required, make it optional.
`<Annotation>`	Additional information about the property. These annotations are important for all properties: • `Term`. Names the specific annotation being applied to a property. • `OData.Permissions`. Documents whether the property can be read or read/write. The values `OData.Permission/Read` and `OData.Permission/ReadWrite` characterize the property. • `OData.Description`. A brief description of the property that generally describes what it is. • `OData.LongDescription`. A longer description of how the property should be used and other additional information.

Item	Description
	Drawing the line between what goes into `OData.Description` and `OData.LongDescription` is a bit of a balancing act. You do not want the description to be overly long, but a user should not need to go to the long description for every property. For additional annotations, see Advanced Topics.

A quick note on the term, ***annotation***. The annotations discussed in this document are schema annotations and not payload annotations, which a response body of a request returns.

5. Advanced topics

Beyond the basics of how to write schemas, a few advanced topics merit discussion.

5.1. Actions

Redfish Schemas may define actions:

1. Declare a general `<Action>` block within the schema. The skeleton template does this for you because it is the same for all schemas.
2. Declare specific actions that reference the actions type. For `Dog`, we define a `Bark` action, as follows:

```
 <Action Name="Bark" IsBound="true">
   <Parameter Name="Dog" Type="Dog.v1_0_0.Actions"/>
   <Parameter Name="BarkCount" Type="Edm.Int64">
     <Annotation Term="OData.Description" String="The number of times to
bark."/>
     <Annotation Term="OData.LongDescription" String="This parameter shall
contain the number of times the dog should bark."/>
   </Parameter>
   <Annotation Term="OData.Description" String="This action instructs the dog
to bark loudly."/>
   <Annotation Term="OData.LongDescription" String="This action shall instruct
the dog to bark loudly."/>
 </Action>
```

5.1.1. Notes on actions

- In Redfish, actions are defined in unversioned namespaces.
- You can specify input parameters to actions. In this case, we have added one parameter: A bark count of type integer.
- The `<Action>` block goes inside the `<Schema>` block but is outside the `<EntityType>` block.

5.2. Constraints

In some cases you may want to add constraints to the values that can be stored for properties. While this generally helps ensure that values are reasonable and can be validated, be careful not to be overly constrained if hardware capabilities, for example, can change.

5.2.1. Numeric ranges

You add `Validation.Minimum` and `Validation.Maximum` term annotations to a property to specify minimum and maximum values for numeric fields.

For example, for `Dog`, we define the `BirthYear` as between the years 0 and 2100 (sorry Astro):

```
<Property Name="BirthYear" Type="Edm.Int64">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
  <Annotation Term="Validation.Minimum" Int="0"/>
  <Annotation Term="Validation.Maximum" Int="2100"/>
  <Annotation Term="OData.Description" String="The year when the dog was born."/>
  <Annotation Term="OData.LongDescription" String="This property shall be the dog's
birth year."/>
</Property>
```

5.2.2. Constrained string values

You add the `Validation.Pattern` term annotation to a property to constrain string data to data that matches a particular regular expression.

For `Dog`, we introduce the (unrealistic) requirement that the dog's name must begin with a capital letter followed by zero or more lowercase letters:

```
<Property Name="Name" Type="Edm.String">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
  <Annotation Term="Validation.Pattern" String="^[A-Z]([a-z])\*$"/>
  <Annotation Term="OData.Description" String="The name for the dog."/>
```

```
  <Annotation Term="OData.LongDescription" String="This property shall be the name by
which the dog is addressed."/>
</Property>
```

The regular expressions for `Validation.Pattern` follow the ECMAScript® Language Specification regular expression dialect.

5.3. Multi-version support

Over time, requirements for schemas can change. A core tenet of Redfish is that while you can add properties and actions, you shall remove them to maintain backwards compatibility for minor versions of the schema. Where necessary, you can deprecate a property.

Note that major version changes, which break backward compatibility with an earlier schema version, require a major version change. You must introduce major schema version changes with a new CSDL file, such as `Dog_v2.xml`. The rest of the discussion in this section only considers minor schema version changes.

5.3.1. To add a schema version

1. Add a `<Schema>` block that is derived from the latest earlier version. Typically, the new `<Schema>` block immediately follows the `<Schema>` block for the earlier version. This code example adds `Dog` v1.1.0:

```
 <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Dog">

   <!-- Existing unversioned definitions for "Dog" -->

   <Action Name="Sit" IsBound="true">
     <Parameter Name="Dog" Type="Dog.v1_0_0.Actions"/>
     <Annotation Term="OData.Description" String="This action instructs the
dog to sit."/>
     <Annotation Term="OData.LongDescription" String="This Action shall
instruct the dog shall sit down."/>
     <Annotation Term="Redfish.Revisions">
       <Collection>
         <Record>
           <PropertyValue Property="Kind" EnumMember="Redfish.RevisionKind/
Added"/>
           <PropertyValue Property="Version" String="v1_1_0"/>
         </Record>
      </Collection>
```

```
    </Annotation>
  </Action>

 </Schema>

 <!-- Existing 1.0.0 definition for "Dog" -->

 <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm"
Namespace="Dog.v1_1_0">
  <Annotation Term="Redfish.OwningEntity" String="DMTF"/>
  <EntityType Name="Dog" BaseType="Dog.v1_0_0.Dog">
    <Property Name="NickName" Type="Edm.String">
      <Annotation Term="OData.Permissions" EnumMember="OData.Permission/
ReadWrite"/>
      <Annotation Term="OData.Description" String="The nickname for this
dog."/>
      <Annotation Term="OData.LongDescription" String="This property shall
contain the most recently used nickname for this dog."/>
    </Property>
  </EntityType>

  <ComplexType Name="VaccinationRecord"
BaseType="Dog.v1_0_0.VaccinationRecord">
    <Property Name="VeterinarianName" Type="Edm.String" Nullable="false">
      <Annotation Term="OData.Description" String="The name of the
veterinarian who applied this vaccination to this dog."/>
      <Annotation Term="OData.LongDescription" String="This property shall
contain the veterinarian's name, in `Last, First` format."/>
    </Property>
  </ComplexType>
 </Schema>
```

Examine each of these items in the `Dog.v_1_1_0` schema:

Item	Description
`<Schema>`	The `Dog.v1_1_0` namespace. This namespace definition ensures that references to the new properties and actions can be resolved appropriately.
`Dog.v1_1_0`	The `Dog` `EntityType`, which is the same name as we had in `Dog.v1_0_0`. This entity type definition ties it to the previous version as an extension.
`BaseType`	The base type. For the `Dog.v1_1_0` entity type, the base type is

Item	Description
	`Dog.v1_0_0`, which also ties it to the earlier version.
`Dog.v1_1_0.Nickname`	The additional top-level properties. Add these properties, such as `Dog.v1_1_0.Nickname`, in the top level of the `<EntityType>` block, just as we did for the initial schema version.
`Dog.Sit`	The additional actions. Add these actions, such as `Dog.Sit`, as `<Action/>` blocks within the `<Schema>` block for the unversioned namespace, but outside the `<EntityType>` block. Take note of the `Redfish.Revisions` term that specifies when the action was added.

Though not shown here, adding complex types and enumeration types works the same way as it did for the original schema version.

Adding properties to a complex type in the earlier schema version is slightly trickier.

1. Duplicate the declaration of the complex type with `BaseType` set to the previous schema version.
    - In this example, `ComplexType Dog.v1_1_0.VaccinationRecord` is declared with `BaseType="Dog.v1_0_0.VaccinationRecord"`.
2. For a complex type that is nested more deeply, duplicate and extend each containing complex type.
3. Add properties to the extended complex type.
    - Here, we add a `VeterinarianName` property to `Dog.v1_1_0.VaccinationRecord`.

To add enumeration values, you add values to the original definition and use the `Redfish.Revision` annotation to annotate each new value.

5.4. External references

The Redfish Schemas are linked together to form a single configuration hierarchy.

These schema linkage types bear discussion:

Schema linkage type	Method	Description
Inclusion inheritance	Add a property with a commonly-used and standard	This inheritance type is similar to the `#include` mechanism in the C and C++

Schema linkage type	Method	Description
	schema-defined type.	programming languages.
Link to subordinate resources	Pull in an entire schema as a direct link under the current schema.	The OpenAPI path to the linked schema includes the present schema's path as a prefix.
Link to related resources	Pull in an entire schema as a related item to the current schema.	The OpenAPI path to the target schema does not include the path to the present schema as a prefix.

In all cases, you must declare the external schema in the initial references section in an `<edmx:Reference/>` block at the top of the schema to be able to reference it. For example, to include a reference to an IP address from the standard Redfish `IPAddresses` schema, add this code:

```
<edmx:Reference Uri="http://redfish.dmtf.org/Schemas/v1/IPAddresses_v1.xml">
  <edmx:Include Namespace="IPAddresses"/>
</edmx:Reference>
```

In this example:

- `http://redfish.dmtf.org/Schemas/v1/IPAddresses_v1.xml` is the standard URI for the schema to which to link.
- `IPAddresses` is the namespace in that schema to which to link.

5.4.1. Inclusion inheritance

Inclusion inheritance, which is the simplest form of inheritance, requires that you define a property with the type set to a declared type within the target schema. For example, you might reference the standard `IPv4Address` type in the `IPAddresses_v1` schema:

```
<Property Name="IPv4Address" Type=" IPAddresses.IPv4Address" Nullable="false">
  <Annotation Term="OData.Description" String="The IPv4 addresses currently assigned
to this dog."/>
  <Annotation Term="OData.LongDescription" String="This property shall be the dog's
current address, or the IPv4 address, on the Internet."/>
</Property>
```

Notes:

- Because it does not make sense to plug a dog into the internet, this property is not present in the `Dog_v1` schema in Dog_v1.xml schema.

- Redfish Schemas very seldom use this kind of inheritance. To create a schema from which you can inherit, consult with a Redfish expert because developing one like this is extremely uncommon.

5.4.2. Link to subordinate resources

Linking to subordinate resources looks very similar to declaring a property. The main difference is that the subordinate resource link is declared as a `<NavigationProperty>` block inside the `<EntityType>` block, parallel to regular properties. For example, the standard DMTF `Storage_v1` schema contains a link to a subordinate `VolumeCollection_v1` reference through this navigation property:

```
<NavigationProperty Name="Volumes" Type="VolumeCollection.VolumeCollection"
ContainsTarget="true" Nullable="false">
  <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
  <Annotation Term="OData.Description" String="The set of volumes produced by the
storage controllers represented by this Resource."/>
  <Annotation Term="OData.LongDescription" String="A collection that indicates all the
volumes produced by the storage controllers that this Resource represents."/>
  <Annotation Term="OData.AutoExpandReferences"/>
</NavigationProperty>
```

Notes:

- The `ContainsTarget` attribute on the navigation property indicates that the data at the target link is contained within the context of the current property. This means that when the resource is no longer available, the resources it links with `ContainsTarget` set to `true` are also expected to no longer be available.
    - Generally speaking, set `ContainsTarget` to `true` when the data at the target link is both not in a resource collection and not in the `Links` object.
    - In this case, the volumes are defined within this particular Storage Resource.
- The `OData.AutoExpandReferences` annotation on indicates that the references contained in the volume collection should be automatically displayed in list under `Volumes` without making a separate, standalone, link for them that has be navigated through.
- Not shown here, an `OData.AutoExpand` annotation would indicate that the data contained within a versioned schema should be automatically displayed within the present schema. For example, the `StorageControllers_v1` collection under `Storage_v1` is declared with `OData.AutoExpand`. OData.AutoExpand should be used instead of `OData.AutoExpandReferences` when the target schema contains data other than references.

5.4.3. Link to related resources

Related links, rather than being contained inline with other properties of a schema, are gathered together into a special property set, or complex type, that is, by convention, named `Links`.

For example, the `Links` property of the `Storage` schema is declared as follows:

```
<Property Name="Links" Type="Storage.v1_1_0.StorageControllerLinks" Nullable="false">
  <Annotation Term="OData.Description" String="Contains references to other resources
that are related to this Resource."/>
  <Annotation Term="OData.LongDescription" String="The Links property, as described by
the Redfish Specification, shall contain references to resources that are related to,
but not contained by (subordinate to), this Resource."/>
</Property>
```

The complex type for the `Links` property is defined with `BaseType="Resource.Links"` to show that it is a link gathering set. The individual related links are simply declared as standard navigation properties.

Here is the example from the `Storage_v1` schema:

```
<ComplexType Name="StorageControllerLinks" BaseType="Resource.Links">
  <Annotation Term="OData.Description" String="Contains references to other resources
that are related to this Resource."/>
  <Annotation Term="OData.LongDescription" String="This type, as described by the
Redfish Specification, shall contain references to resources that are related to, but
not contained by (subordinate to), this Resource."/>
  <NavigationProperty Name="Endpoints" Type="Collection(Endpoint.Endpoint)">
    <Annotation Term="OData.Description" String="An array of references to the
endpoints that connect to this controller."/>
    <Annotation Term="OData.LongDescription" String="This property shall be a
reference to the resources that this controller is associated with and shall reference
a resource of type Endpoint."/>
    <Annotation Term="OData.AutoExpandReferences"/>
  </NavigationProperty>
</ComplexType>
```

6. CSDL schema template

```
<?xml version="1.0" encoding="UTF-8"?>
<!----->
<!--################################################################################
```

```
   -->
<!-- Redfish Schema:  SchemaName
v1.0.0                                                    -->
<!--
   -->
<!-- Copyright 2019
DMTF.                                                                  -->
<!-- For the full DMTF copyright policy, see http://www.dmtf.org/about/policies/
copyright -->
<!--##################################################################################
   -->
<!----->
<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">
  <!-- This section consists of references to standard Redfish Schema bits. -->
  <!-- Do not modify these -->
  <edmx:Reference Uri="http://docs.oasis-open.org/odata/odata/v4.0/errata03/csd01/
complete/vocabularies/Org.OData.Core.V1.xml">
    <edmx:Include Namespace="Org.OData.Core.V1" Alias="OData"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://docs.oasis-open.org/odata/odata/v4.0/errata03/csd01/
complete/vocabularies/Org.OData.Capabilities.V1.xml">
    <edmx:Include Namespace="Org.OData.Capabilities.V1" Alias="Capabilities"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://redfish.dmtf.org/Schemas/v1/Resource_v1.xml">
    <edmx:Include Namespace="Resource"/>
    <edmx:Include Namespace="Resource.v1_0_0"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://redfish.dmtf.org/Schemas/v1/RedfishExtensions_v1.xml">
    <edmx:Include Namespace="RedfishExtensions.v1_0_0" Alias="Redfish"/>
    <edmx:Include Namespace="Validation.v1_0_0" Alias="Validation"/>
  </edmx:Reference>
  <!-- You can add your own schema-specific references here -->
  <edmx:Reference Uri="http://redfish.dmtf.org/Schemas/v1/ReferencedSchemaName_v1.xml">
    <edmx:Include Namespace="ReferencedSchemaName"/>
  </edmx:Reference>

  <!-- This section contains all the data properties for your schema -->
  <edmx:DataServices>

    <!-- This bit declares the namespace for your schema, enabling it to be referenced
-->
    <!-- from within Redfish -->
    <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="SchemaName">
      <Annotation Term="Redfish.OwningEntity" String="DMTF"/>

      <!-- This declares the main entity type for your schema, establishing a root for
the data hierarchy in it -->
```

```
      <EntityType Name="SchemaName" BaseType="Resource.v1_0_0.Resource"
Abstract="true">
        <Annotation Term="OData.Description" String="The SchemaName schema defines the
properties of a SchemaName attached to a system."/>
        <Annotation Term="OData.LongDescription" String="This resource shall be used
to represent a SchemaName attached to a system."/>
        <Annotation Term="OData.AdditionalProperties" Bool="false"/>
        <Annotation Term="Redfish.Uris">
          <!-- These are the standard Redfish OpenAPI URIs for instances of your
schema -->
          <Collection>
            <String>/redfish/v1/Chassis/{ChassisId}/SchemaNames/{SchemaNameId}</String>
            <String>/redfish/v1/
Systems/{ComputerSystemId}/SchemaNames/{SchemaNameId}</String>
          </Collection>
        </Annotation>
      </EntityType>

      <!-- Define actions for your resource here, if needed -->

    </Schema>

    <!-- This section contains information on the various properties contained -->
    <!-- within version 1.0.0 of your schema -->
    <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm"
Namespace="SchemaName.v1_0_0">
      <Annotation Term="Redfish.OwningEntity" String="DMTF"/>
      <EntityType Name="SchemaName" BaseType=" SchemaName.SchemaName">
        <!-- The top-level properties for your schema go here. -->
        <Property Name="SampleProperty" Type="Edm.String" Nullable="false">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
          <Annotation Term="OData.Description" String="A sample property."/>
          <Annotation Term="OData.LongDescription" String="This property shall contain
a longer description of the sample property."/>
        </Property>
      </EntityType>

      <!-- Additional complex and enumeration types are defined here,  -->
      <!-- within the context of the schema -->

    </Schema>

    <!-- Additional versions of your schema can go here -->

  </edmx:DataServices>
</edmx:Edmx>
```

7. Dog_v1.xml schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!----->
<!--##################################################################################
   -->
<!-- Redfish Schema:  Dog
v1.1.0                                                                -->
<!--
   -->
<!-- Copyright 2019
DMTF.                                                                 -->
<!-- For the full DMTF copyright policy, see http://www.dmtf.org/about/policies/
copyright -->
<!--##################################################################################
   -->
<!----->
<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">
  <edmx:Reference Uri="http://docs.oasis-open.org/odata/odata/v4.0/errata03/csd01/
complete/vocabularies/Org.OData.Core.V1.xml">
    <edmx:Include Namespace="Org.OData.Core.V1" Alias="OData"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://docs.oasis-open.org/odata/odata/v4.0/errata03/csd01/
complete/vocabularies/Org.OData.Capabilities.V1.xml">
    <edmx:Include Namespace="Org.OData.Capabilities.V1" Alias="Capabilities"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://redfish.dmtf.org/Schemas/v1/Resource_v1.xml">
    <edmx:Include Namespace="Resource"/>
    <edmx:Include Namespace="Resource.v1_0_0"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://redfish.dmtf.org/Schemas/v1/RedfishExtensions_v1.xml">
    <edmx:Include Namespace="RedfishExtensions.v1_0_0" Alias="Redfish"/>
    <edmx:Include Namespace="Validation.v1_0_0" Alias="Validation"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://redfish.dmtf.org/Schemas/v1/Person_v1.xml">
    <edmx:Include Namespace="Person"/>
  </edmx:Reference>
  <edmx:DataServices>

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

      <EntityType Name="Dog" BaseType="Resource.v1_0_0.Resource" Abstract="true">
        <Annotation Term="OData.Description" String="The Dog schema defines the Dog
Resource.  It represents the properties of a dog."/>
        <Annotation Term="OData.LongDescription" String="This Resource shall be used
```

```
to represent a dog."/>
        <Annotation Term="Capabilities.InsertRestrictions">
          <Record>
            <PropertyValue Property="Insertable" Bool="false"/>
          </Record>
        </Annotation>
        <Annotation Term="Capabilities.UpdateRestrictions">
          <Record>
            <PropertyValue Property="Updatable" Bool="false"/>
          </Record>
        </Annotation>
        <Annotation Term="Capabilities.DeleteRestrictions">
          <Record>
            <PropertyValue Property="Deletable" Bool="false"/>
          </Record>
        </Annotation>
        <Annotation Term="Redfish.Uris">
          <Collection>
            <String>/redfish/v1/Chassis/{ChassisId}/Dogs/{DogId}</String>
            <String>/redfish/v1/Systems/{ComputerSystemId}/Dogs/{DogId}</String>
          </Collection>
        </Annotation>
      </EntityType>

      <Action Name="Bark" IsBound="true">
        <Parameter Name="Dog" Type="Dog.v1_0_0.Actions"/>
        <Parameter Name="BarkCount" Type="Edm.Int64">
          <Annotation Term="OData.Description" String="The number of times to bark."/>
          <Annotation Term="OData.LongDescription" String="This parameter shall
contain the number of times the dog should bark."/>
        </Parameter>
        <Annotation Term="OData.Description" String="This action instructs the dog to
bark loudly."/>
        <Annotation Term="OData.LongDescription" String="This action shall instruct
the dog to bark loudly."/>
      </Action>

      <Action Name="Sit" IsBound="true">
        <Parameter Name="Dog" Type="Dog.v1_0_0.Actions"/>
        <Annotation Term="OData.Description" String="This action instructs the dog to
sit."/>
        <Annotation Term="OData.LongDescription" String="This Action shall instruct
the dog shall sit down."/>
        <Annotation Term="Redfish.Revisions">
          <Collection>
            <Record>
              <PropertyValue Property="Kind" EnumMember="Redfish.RevisionKind/Added"/>
```

```
              <PropertyValue Property="Version" String="v1_1_0"/>
            </Record>
          </Collection>
        </Annotation>
      </Action>

    </Schema>

    <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Dog.v1_0_0">
      <Annotation Term="Redfish.OwningEntity" String="DMTF"/>
      <Annotation Term="Redfish.Release" String="2016.2"/>

      <EntityType Name="Dog" BaseType="Dog.Dog">
        <Property Name="Actions" Type="Dog.v1_0_0.Actions" Nullable="false">
          <Annotation Term="OData.Description" String="The available actions for this
Resource."/>
          <Annotation Term="OData.LongDescription" String="This property shall contain
the available actions for this Resource."/>
        </Property>
        <Property Name="Name" Type="Edm.String">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
          <Annotation Term="Validation.Pattern" String="^[A-Z]([a-z])\*$"/>
          <Annotation Term="OData.Description" String="The name for the dog."/>
          <Annotation Term="OData.LongDescription" String="This property shall be the
name by which the dog is addressed."/>
        </Property>
        <Property Name="PrimaryColor" Type="Dog.v1_0_0.PrimaryColor">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
          <Annotation Term="OData.Description" String="The main color of the dog."/>
          <Annotation Term="OData.LongDescription" String="This property shall be the
main color of the dog."/>
        </Property>
        <Property Name="BirthYear" Type="Edm.Int64">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
          <Annotation Term="Validation.Minimum" Int="0"/>
          <Annotation Term="Validation.Maximum" Int="2100"/>
          <Annotation Term="OData.Description" String="The year when the dog was
born."/>
          <Annotation Term="OData.LongDescription" String="This property shall be the
dog's birth year."/>
        </Property>
        <Property Name="SpayedOrNeutered" Type="Edm.Boolean">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/
ReadWrite"/>
          <Annotation Term="OData.Description" String="An indication of whether the
dog has been spayed or neutered.  If true, the dog has been spayed or neutered."/>
          <Annotation Term="OData.LongDescription" String="This property shall
```

```
indicate whether the dog has been spayed or neutered.  If true, the dog has been
spayed or neutered."/>
        </Property>
        <Property Name="Owner" Type="Edm.String">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/
ReadWrite"/>
          <Annotation Term="OData.Description" String="The dog's owner's name."/>
          <Annotation Term="OData.LongDescription" String="This property shall be the
name of the dog's owner."/>
        </Property>
        <Property Name="Breeds" Type="Collection(Edm.String)">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
          <Annotation Term="OData.Description" String="The dog's breed."/>
          <Annotation Term="OData.LongDescription" String="This property shall be a
name from the international registry of dog breeds."/>
        </Property>
        <Property Name="VaccinationRecord"
Type="Collection(Dog.v1_0_0.VaccinationRecord)" Nullable="false">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/
ReadWrite"/>
          <Annotation Term="OData.Description" String="The vaccination records for
this dog."/>
          <Annotation Term="OData.LongDescription" String="This property shall be an
array of records for the vaccinations applied to this dog."/>
        </Property>
      </EntityType>

      <ComplexType Name="Actions">
        <Annotation Term="OData.AdditionalProperties" Bool="false"/>
        <Annotation Term="OData.Description" String="The available actions for this
Resource."/>
        <Annotation Term="OData.LongDescription" String="This type shall contain the
available actions for this Resource."/>
        <Property Name="Oem" Type="Dog.v1_0_0.OemActions" Nullable="false">
          <Annotation Term="OData.Description" String="This property contains the
available OEM-specific actions for this Resource."/>
          <Annotation Term="OData.LongDescription" String="This property shall contain
any additional OEM actions for this Resource."/>
        </Property>
      </ComplexType>

      <ComplexType Name="OemActions">
        <Annotation Term="OData.AdditionalProperties" Bool="true"/>
        <Annotation Term="OData.Description" String="The available OEM-specific
actions for this Resource."/>
        <Annotation Term="OData.LongDescription" String="This type shall contain any
additional OEM actions for this Resource."/>
```

```
      </ComplexType>

      <EnumType Name="PrimaryColor">
        <Member Name="Black">
          <Annotation Term="OData.Description" String="A black dog."/>
        </Member>
        <Member Name="Brown">
          <Annotation Term="OData.Description" String="A brown dog."/>
        </Member>
        <Member Name="Yellow">
          <Annotation Term="OData.Description" String="A yellow dog."/>
        </Member>
        <Member Name="Red">
          <Annotation Term="OData.Description" String="A red dog."/>
        </Member>
        <Member Name="White">
          <Annotation Term="OData.Description" String="A white dog."/>
        </Member>
      </EnumType>

      <ComplexType Name="VaccinationRecord">
        <Annotation Term="OData.AdditionalProperties" Bool="false"/>
        <Annotation Term="OData.Description" String="The vaccination applied to this
dog."/>
        <Annotation Term="OData.LongDescription" String="This type shall contain the
vaccination applied to this dog."/>
        <Property Name="VaccinationName" Type="Edm.String" Nullable="false">
          <Annotation Term="OData.Description" String="The name of a vaccination
applied to this dog."/>
          <Annotation Term="OData.LongDescription" String="This property shall contain
the name of a vaccination applied to this dog."/>
        </Property>
        <Property Name="YearApplied" Type="Edm.Int64" Nullable="false">
          <Annotation Term="OData.Description" String="The year this dog was
vaccinated."/>
          <Annotation Term="OData.LongDescription" String="This property shall contain
the year this dog was vaccinated."/>
        </Property>
      </ComplexType>
    </Schema>

    <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Dog.v1_1_0">
      <Annotation Term="Redfish.OwningEntity" String="DMTF"/>
      <EntityType Name="Dog" BaseType="Dog.v1_0_0.Dog">
        <Property Name="NickName" Type="Edm.String">
          <Annotation Term="OData.Permissions" EnumMember="OData.Permission/
ReadWrite"/>
```

```
          <Annotation Term="OData.Description" String="The nickname for this dog."/>
          <Annotation Term="OData.LongDescription" String="This property shall contain
the most recently used nickname for this dog."/>
        </Property>
      </EntityType>

      <ComplexType Name="VaccinationRecord" BaseType="Dog.v1_0_0.VaccinationRecord">
        <Property Name="VeterinarianName" Type="Edm.String" Nullable="false">
          <Annotation Term="OData.Description" String="The name of the veterinarian
who applied this vaccination to this dog."/>
          <Annotation Term="OData.LongDescription" String="This property shall contain
the veterinarian's name, in `Last, First` format."/>
        </Property>
      </ComplexType>

    </Schema>

  </edmx:DataServices>
</edmx:Edmx>
```

8. Change log

Version	Date	Description
1.0.0	2019-08-06	Initial release.