Introduction to CSDL

Mike Raineri

Dell Inc.

Redfish

What Is CSDL?

- CSDL: Common Schema Definition Language
- A schema format defined by OData to standardize how data elements are represented in a RESTful service
  - Allows for clients to dynamically scan and adapt to a service's data model
  - Provides documentation for developers when writing purpose built clients
- CSDL files are written in XML
- The XML structures define the JSON properties and objects that a service uses in its payloads, as well as constraints that may apply
- Inline annotations are also used to provide clients and users with more detailed information about a given property or object
- CSDL specification can be found here: http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part3-csdl.html

Schema Layout

- The primary body of a schema file contains namespace definitions
  - A namespace is a unique name for a set of type definitions being declared
  - Multiple namespaces can be defined in a single file
  - Namespaces within the same schema file may reference each other
  - Type definitions are referenced as "*Namespace.TypeDefinition*"
- If a schema file requires references to namespaces defined in other schema files, then a reference to the namespace must be included
  - The reference includes the URI of the schema file being referenced in addition to which namespaces in the schema file to include
  - These references are normally done at the top of the file
  - Primitive types in the "Edm" namespace do not need to be included

Schema File Example

```
<edmx:Edmx xmlns:edms="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">

  <edmx:Reference Uri="http://contoso.org/schemas/ExternalSchema.xml">
    <edmx:Include Namespace="ExternalNamespace"/>
    <edmx:Include Namespace="Other.Namespace"/>
  </edmx:Reference>

  <edmx:DataServices>

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

      <ComplexType Name="MyDataType">
        <Property Name="MyProperty" Type="ExternalNamespace.ReferencedDataType"/>
        <Property Name="MyProperty2" Type="Other.Namespace.OtherDataType"/>
        <Property Name="MyProperty3" Type="Edm.Int64"/>
      </ComplexType>

    </Schema>

  </edmx:DataServices>

</edmx:Edmx>
```

Schema Elements: Property

- Defines a property inside of a JSON object; provides the name of the property and data type of the property

- CSDL example

```
<Property Name="SerialNumber" Type="Edm.String"/>
```

- JSON format from a service

```
{
      ...
      "SerialNumber": "123456789",
      ...
}
```

Schema Elements: NavigationProperty

- Defines a property inside of a JSON object that links to another resource in the service; provides the name of the property and the data type of the resource it links

- CSDL example

```
<NavigationProperty Name="Thermal" Type="Thermal.Thermal"/>
```

- JSON format from a service

```
{
    ...
    "Thermal": {
        "@odata.id": "/redfish/v1/Chassis/1/Thermal"
    },
    ...
}
```

Schema Elements: EnumType

- Defines a set of strings that show the valid values of a given property; each member defined is a valid value for the given enum

- Defining an EnumType

```
<EnumType Name="IndicatorLED">
  <Member Name="Lit"/>
  <Member Name="Blinking"/>
  <Member Name="Off"/>
</EnumType>
```

- Using an EnumType

```
<Property Name="IndicatorLED" Type="Resource.v1_1_0.IndiatorLED"/>
```

Schema Elements: ComplexType

- Defines an object with a set of properties; can contain any number of properties or navigation properties

- Defining a ComplexType

```
<ComplexType Name="ProcessorId">
  <Property Name="VendorId" Type="Edm.String"/>
  <Property Name="IdentificationRegisters" Type="Edm.String"/>
  <Property Name="EffectiveFamily" Type="Edm.String"/>
  <Property Name="EffectiveModel" Type="Edm.String"/>
  <Property Name="Step" Type="Edm.String"/>
  <Property Name="MicrocodeInfo" Type="Edm.String"/>
</ComplexType>
```

- Using a ComplexType

```
<Property Name="ProcessorId" Type="Processor.v1_0_0.ProcessorId"/>
```

Schema Elements: ComplexType (cont)

- JSON format from a service

```
{
    ...
    "ProcessorId": {
        "VendorId": "GenuineIntel",
        "IdentificationRegisters": "0x34AC34DC8901274A",
        "EffectiveFamily": "0x42",
        "EffectiveModel": "0x61",
        "Step": "0x1",
        "MicrocodeInfo": "0x429943"
    },
    ...
}
```

Schema Elements: EntityType

- Defines an object with a set of properties and a uniquely identifiable key; can contain any number of properties or navigation properties
- Used to define payloads for an entire resource

- Defining an EntityType

```
<EntityType Name="Processor">
  <Key>
    <PropertyRef Name="Id"/>
  </Key>
  <Property Name="Id" Type="Edm.String"/>
  <Property Name="Name" Type="Edm.String"/>
  <Property Name="MaxSpeedMhz" Type="Edm.Int64"/>
  <Property Name="TotalCores" Type="Edm.Int64"/>
</EntityType>
```

Schema Elements: EntityType (cont)

- JSON format from a service

```
{
    "Id": "CPU0",
    "Name": "Processor in Socket 0",
    "MaxSpeedMhz": 2000,
    "TotalCores": 16
}
```

Schema Elements: Action

- Defines an operation that a client can perform by submitting a POST request to the Action URI
- A service advertises the supported actions as part of the response to a GET request

- Defining an Action

```
<Action Name="Reset" IsBound="true">
  <Parameter Name="Manager" Type="Manager.v1_0_0.Actions"/>
  <Parameter Name="ResetType" Type="Resource.ResetType"/>
</Action>
```

Schema Elements: Action (cont)

- JSON format from a service

```
{
    ...
    "Actions": {
        "#Manager.Reset": {
            "target": "/redfish/v1/Managers/1/Actions/Manager.Reset",
            "ResetType@Redfish.AllowableValues": [
                "On",
                "ForceOff",
                "GracefulShutdown",
                "GracefulRestart",
                "ForceRestart",
                "ForceOn"
            ]
        }
    },
    ...
}
```

Schema Elements: Annotation

- Used to provide inline documentation for anything defined in the schema
- Annotations give guidance and/or express conformance rules for clients or services

- CSDL example

```
<Property Name="UserName" Type="Edm.String">
  <Annotation Term="Redfish.RequiredOnCreate"/>
  <Annotation Term="OData.Permissions"
      EnumMember="OData.Permissions/ReadWrite"/>
  <Annotation Term="OData.Description"
      String="This property contains the user name for the account."/>
</Property>
```

Inheritance

- Both EntityType and ComplexType definitions can define a BaseType
- The value for the BaseType is the name of the EntityType or ComplexType in which the new type is referencing
- All properties defined by the BaseType become available to the newly defined type

- Defining a BaseType

```
<ComplexType Name="Protocol">
  <Property Name="ProtocolEnabled" Type="Boolean"/>
  <Property Name="Port" Type="Edm.Int64"/>
</ComplexType>
```

Inheritance (cont)

- Referencing the BaseType

```
<ComplexType Name="SSDProtocol"
      BaseType="ManagerNetworkProtocol.v1_0_0.Protocol">
  <Property Name="NotifyMulticastIntervalSeconds"
      Type="Edm.Int64"/>
  <Property Name="NotifyTTL" Type="Edm.Int64"/>
</ComplexType>
```

Thank you for watching!

- Redfish Standards
  - Schemas, Specs, Mockups, White Papers, FAQ, Educational Material & more
  - http://www.dmtf.org/standards/redfish
- Redfish Developer Hub
  - Redfish Interactive Explorer, Hosted Schema at Namespace & other links
  - http://redfish.dmtf.org
- SPMF (WG that defines Redfish)
  - Companies involved, Upcoming Schedules & Future work, Charter, Information on joining.
  - http://www.dmtf.org/standards/spmf