This is a Non-Standards Track Work Product. The patent provisions of the OASIS IPR Policy do not apply.

OData to OpenAPI Mapping Version 1.0

Committee Note Draft 01 / Public Review Draft 01

15 December 2016

Specification URIs

This version:

http://docs.oasis-open.org/odata/odata-openapi/v1.0/cnprd01/odata-openapi-v1.0-cnprd01.docx (Authoritative)

http://docs.oasis-open.org/odata/odata-openapi/v1.0/cnprd01/odata-openapi-v1.0-cnprd01.html

http://docs.oasis-open.org/odata/odata-openapi/v1.0/cnprd01/odata-openapi-v1.0-cnprd01.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/odata/odata-openapi/v1.0/odata-openapi-v1.0.docx (Authoritative)

http://docs.oasis-open.org/odata/odata-openapi/v1.0/odata-openapi-v1.0.html

http://docs.oasis-open.org/odata/odata-openapi/v1.0/odata-openapi-v1.0.pdf

Technical Committee:

OASIS Open Data Protocol (OData) TC

Chairs:

Ralf Handl (ralf.handl@sap.com), SAP SE

Mike Pizzo (mikep@microsoft.com), Microsoft

Editors:

Ralf Handl (ralf.handl@sap.com), SAP SE

Hubert Heijkers (hubert.heijkers@nl.ibm.com), IBM

Mike Pizzo (mikep@microsoft.com), Microsoft

Martin Zurmuehl (martin.zurmuehl@sap.com), SAP SE

Related work:

This document is related to:

- *OData Version 4.0*, a multi-part Work Product which includes:

  - *OData Version 4.0 Part 1: Protocol*. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/part1-protocol/odata-v4.0-os-part1-protocol.html.
  - *OData Version 4.0 Part 2: URL Conventions*. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/part2-url-conventions/odata-v4.0-os-part2-url-conventions.html.
  - *OData Version 4.0 Part 3: Common Schema Definition Language (CSDL)*. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/part3-csdl/odata-v4.0-os-part3-csdl.html.
  - Vocabulary components: *OData Core Vocabulary, OData Measures Vocabulary* and *OData Capabilities Vocabulary*. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/
- *OData JSON Format Version 4.0*. OASIS Standard. 24 February 2014. http://docs.oasis-open.org/odata/odata-json-format/v4.0/os/odata-json-format-v4.0-os.html.

Abstract:

The Open Data Protocol (OData) is an open protocol for creating and consuming queryable and interoperable RESTful APIs in a simple and standard way. OData services are described by an entity-relationship model, and the model description is an integral part of each OData service.

The OpenAPI Specification (OAS) is a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service. This document describes a possible mapping of OData service descriptions to OAS documents.

Status:

This document was last revised or approved by the OASIS Open Data Protocol (OData) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document.

Technical Committee (TC) members should send comments on this document to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/odata/.

Citation format:

When referencing this document the following citation format should be used:

[OData-OpenAPI-v1.0]

OData to OpenAPI Mapping Version 1.0. Edited by Ralf Handl, Hubert Heijkers, Mike Pizzo, and Martin Zurmuehl. 15 December 2016. OASIS Committee Note Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/odata/odata-openapi/v1.0/cnprd01/odata-openapi-v1.0-cnprd01.html. Latest version: http://docs.oasis-open.org/odata/odata-openapi/v1.0/odata-openapi-v1.0.html.

Table of Contents

1 Introduction

OData services are described in terms of an Entity Data Model (EDM). **[OData-CSDL]** defines an XML representation of the entity data model exposed by an OData service.

The OpenAPI Specification (OAS, formerly known as Swagger RESTful API Documentation Specification) is a project used to describe and document RESTful APIs. It defines a set of JSON or YAML files required to describe such an API. These files can then be used by various tools to display the API, test the API, or generate clients in various programming languages.

This document describes a possible mapping of OData service descriptions to OAS documents which allows OpenAPI tools to be used for interacting with OData services.

OData is based on a powerful set of concepts and conventions which allow rich interaction with OData services. OpenAPI on the other hand does not assume or rely on any conventions and requires explicit and – from an OData perspective – relatively low-level and repetitive description of each service feature. As a consequence this mapping only translates the basic features of an OData service into OpenAPI terms to allow an easy "first contact" by exploring it e.g. with the Swagger UI **[Swagger UI]**, rather than trying to capture all features of an OData service in an unmanageably long OAS document.

1.1 References (non-normative)

[OData-CSDL] *OData Version 4.0 Part 3: Common Schema Definition Language (CSDL)*.
See link in "Related work" section on cover page.

[OData-JSON] *OData JSON Format Version 4.0.*
See link in "Related work" section on cover page.

[OData-OpenAPI] *odata-openapi OASIS TC GitHub repository*
https://github.com/oasis-tcs/odata-openapi.

[OData-Protocol] *OData Version 4.0 Part 1: Protocol*.
See link in "Additional artifacts" section on cover page.

[OData-URL] *OData Version 4.0 Part 2: URL Conventions*.
See link in "Related work" section on cover page.

[OData-VocCore] *OData Core Vocabulary.*
See link in "Related work" section on cover page.

[OData-VocCap] *OData Capabilities Vocabulary.*
See link in "Related work" section on cover page.

[OpenAPI] *OpenAPI Specification Version 2.0*,
https://openapis.org/specification, specifically
https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014. http://tools.ietf.org/html/rfc7159.

[Swagger UI] https://github.com/swagger-api/swagger-ui

[YAML] YAML Ain't Markup Language (YAML™), Version 1.2, 3rd Edition, Patched at 2001-10-01. Copyright © 2001-2009 Oren Ben-Kiki, Clark Evans, Ingy döt Net. http://www.yaml.org/spec/1.2/spec.html

2 Design Principles

Given the different goals of and levels of abstractions used by OData and OpenAPI, this mapping of OData metadata documents into OAS documents is intentionally lossy and only tries to preserve the main features of an OData service:

- The entity container is translated into an OpenAPI Paths Object with a path templates and operation objects for all top-level resources described by the entity container
- Structure-describing CSDL elements (structured types, type definitions, enumerations) are translated into OpenAPI Schema Objects within the OpenAPI Definitions Object
- CSDL constructs that don't have an OpenAPI counterpart are omitted

3 Providing OAS Documents for an OData Service

OAS documents describing an OData service can be provided in several ways, and the examples given here are by no means exhaustive or mutually exclusive.

Typical provisioning is as a static resource, e.g. as part of a Service Catalog or API Hub.

A more OData-ish way is to provide the OAS document as part of the service. Following the OpenAPI convention, this would be a resource `<service-root>/swagger.json` at the service root, next to `<service-root>/$metadata`.

The OAS document could also be seen as an alternative representation of the service document, so it could be served at `<service-root>/?$format=swagger` or `<service-root>/?$format=openapi2`.

A more sophisticated way would be to provide it wrapped in an exploration tool, e.g. `<service-root>/?$format=html` could start **[Swagger UI]** and pre-load the OAS document of that service. This could even be the default response when accessing the service document without the `$format` query option from a browser because `text/html` is among the most acceptable media types in the `Accept` header sent by the most common browsers.

4 OAS Document Structure

OAS documents are represented as JSON objects and conform to **[RFC7159]**. **[YAML]**, being a superset of JSON, can be used as well to represent an OAS document.

An OAS document consists of a single Swagger Object, see **[OpenAPI]**. It is represented as a JSON object. How to construct each of its name/value pairs ("fields" in OpenAPI terminology) is described in the following sections.

EXAMPLE 1: STRUCTURE OF AN OAS DOCUMENT

```
{
  "swagger":"2.0",
  "info": …,
  "schemes": …,
  "host": …,
  "basepath": …,
  "consumes": …,
  "produces": …,
  "tags": …,
  "paths": …,
  "definitions": …,
  "parameters": …,
  "responses": …
}
```

4.1 Field `swagger`

The value of `swagger` is the string `"2.0"`.

4.2 Field `info`

The value of `info` is an Info Object, see **[OpenAPI]**. It contains the fields `title` and `version`, and it contains the keyword `description`.

EXAMPLE 2: INFO OBJECT – NOTE THAT DESCRIPTION ACCEPTS MARKDOWN FORMAT

```
"info":{
  "title":"OData Service for namespace ODataDemo",
  "version":"0.1.0",
  "description":"This OData service is located at
http://localhost/service-root/\n\n## References\n-
[Org.OData.Core.V1](http://localhost/swagger-
ui/?url=http://localhost/examples/Org.OData.Core.V1.openapi.json)\n-
[Org.OData.Measures.V1](http://localhost/swagger-
ui/?url=http://localhost/examples/Org.OData.Measures.V1.openapi.json)"
}
```

4.2.1 Field `title`

The value of `title` is the value of the unqualified annotation `Core.Description` (see**[OData-VocCore]**) of the main schema or the entity container of the OData service.

If no `Core.Description` is present, a default title has to be provided as this is a required OpenAPI field.

4.2.2 Field `version`

The value of `version` is the value of the annotation `Core.SchemaVersion` (see **[OData-VocCore]**) of the main schema.

If no `Core.SchemaVersion` is present, a default version has to be provided as this is a required OpenAPI field.

4.2.3 Field `description`

The value of `description` is the value of the annotation `Core.LongDescription` (see **[OData-VocCore]**) of the main schema or the entity container.

While this field is optional, it prominently appears in OpenAPI exploration tools, so a default description should be provided if no `Core.LongDescription` annotation is present.

4.3 Fields `schemes`, `host`, and `basePath`

The value of `schemes` is an array with one string containing the scheme component of the service root URL.

The value of `host` is a string containing the authority component of the service root URL.

The value of `basePath` is a string containing the path component of the service root URL without a trailing forward-slash.

EXAMPLE 3: SERVICE ROOT URL

```
"schemes": [
  "http"
],
"host":"localhost",
"basePath":"/service-root"
```

4.4 Fields `consumes` and `produces`

The values of `consumes` and `produces` are arrays of strings. If present they contain an item for each media type listed in the `Capabilities.SupportedFormats` annotation (see **[OData-VocCap]**) on the entity container.

EXAMPLE 4: SUPPORTED FORMATS

```
"consumes":[
  "application/json"
],
"produces":[
  "application/json"
]
```


4.5 Field `tags`

The value of `tags` is an array of Tag Objects, see **[OpenAPI]**. Tags are used for logical grouping of operations. For an OData service the natural groups are entity sets and singletons, so the `tags` array contains one Tag Object per entity set and singleton in the entity container.

A Tag Object has to contain the field `name`, whose value is the name of the entity set or singleton, and it optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the entity set or singleton.

The `tags` array can contain additional Tag Objects for other logical groups, e.g. for action imports or function imports that are not associated with an entity set.

EXAMPLE 5: TAGS WITH OPTIONAL DESCRIPTIONS

```
"tags":[
  {
    "name": "Products"
  },
  {
    "name": "Categories",
    "description": "Product Categories"
  },
  {
    "name": "Suppliers"
  },
  {
    "name": "MainSupplier",
    "description": "Primary Supplier"
  },
  {
    "name": "Countries"
  }
]
```

4.6 Field `paths`

The value of `paths` is a Paths Object, see **[OpenAPI]**. It is the main source of information on how to use the described API. It consists of name/value pairs whose name is a path template relative to the service root URL, and whose value is a Path Item Object, see **[OpenAPI]**.

Due to the power and flexibility of OData a full representation of all service capabilities in the Paths Object is typically not feasible, so this mapping only describes the minimum information desired in the Paths Object. Implementations are allowed – and in fact encouraged – to add additional information that is deemed useful for the intended target audience of the OpenAPI description of that service, leveraging the documentation features of the OpenAPI Specification, especially and not limited to human-readable descriptions.

The minimum information to be included in the Paths Object is described in the remainder of this section. The Paths Object reflects the top-level resources and capabilities of the service as closely as possible, i.e. only list supported operations and query options.

EXAMPLE 6: PATHS FOR ENTITY SETS, INDIVIDUAL ENTITIES, SINGLETONS, ACTION IMPORTS, AND FUNCTION IMPORTS


```
"paths": {
  "/Products": …,
  "/Products('{ID}')": …,
  "/Categories": …,
  "/Categories({ID})": …,
  "/Suppliers": …,
  "/Suppliers('{ID}')": …,
  "/MainSupplier": …,
  "/Countries": …,
  "/Countries('{Code}')": …,
  "/ProductsByRating(Rating={Rating})": …
}
```

4.6.1 Paths for Entity Sets

Each entity set is represented as a name/value pair whose name is the service-relative resource path of the entity set prepended with a forward slash, and whose value is a Path Item Object, see **[OpenAPI].**

EXAMPLE 7: PATH TEMPLATE OPERATION FOR AN ENTITY SET

```
"/Products": …
```

Each entity set that is indexable by key is additionally represented as a name/value pair whose name is the path template for key access, with path parameters for the key values, and whose value is a Path Item Object describing the allowed operations on individual entities of this set.

EXAMPLE 8: PATH TEMPLATE FOR AN INDIVIDUAL ENTITY WITHIN AN ENTITY SET – SINGLE-PART KEY

```
"/Products('{ID}')": …
```

EXAMPLE 9: PATH TEMPLATE FOR AN INDIVIDUAL ENTITY WITHIN AN ENTITY SET – MULTI-PART KEY

```
"/OrderItems(OrderID={OrderID},ItemID={ItemID})": …
```

If the service defines bound actions or functions applicable to the entity set or its entities, these are represented as additional name/value pairs with corresponding path templates for the action/function invocation.

EXAMPLE 10: PATH TEMPLATE FOR A BOUND ACTION

```
"/LeaveRequests({ID})/OData.Demo.Approval": …
```

4.6.1.1 Query a Collection of Entities

The Path Item Object for the entity set contains the keyword `get` with an Operation Object as value that describes the capabilities for querying the entity set. The `tags` array of the Operation Object – as well as all other Operation Objects described in this section – includes the entity set name.

EXAMPLE 11: `GET` OPERATION FOR AN ENTITY SET – `SUMMARY` AND `TAGS`

```
"/Products": {
  "get": {
    "summary": "Get entities from Products",
    "tags": [
      "Products"
```



```
    ],
```

The `parameters` array contains Parameter Objects for system query options allowed for this entity set, and it does not list system query options not allowed for this entity set.

EXAMPLE 12: GET OPERATION FOR AN ENTITY SET - PARAMETERS

```
    "parameters": [
      {
        "$ref": "#/parameters/top"
      },
      {
        "$ref": "#/parameters/skip"
      },
      {
        "$ref": "#/parameters/search"
      },
      {
        "$ref": "#/parameters/filter"
      },
      {
        "$ref": "#/parameters/count"
      },
```

Note: the syntax of the system query options `$expand`, `$select`, and `$orderby` is too flexible to be formally described with OpenAPI Specification means, yet the typical use cases of just providing a comma-separated list of properties can be expressed via an array-valued parameter with an `enum` constraint, as shown in the following example. This makes it easy to try out these system query options in OpenAPI tools.

EXAMPLE 13: GET OPERATION FOR AN ENTITY SET – MORE SPECIFIC PARAMETERS

```
      {
        "name": "$expand",
        "in": "query",
        "description": "Expand related entities, see [OData
Expand](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374621)",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string"
        },
        "enum": [
          "*",
          "Category",
          "Supplier"
        ]
      },
      {
        "name": "$select",
        "in": "query",
        "description": "Select properties to be returned, see [OData
Select](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374620)",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string"
        },
        "enum": [
```



```
          "ID",
          "Description",
          "ReleaseDate",
          "DiscontinuedDate",
          "Rating",
          "Price",
          "Currency"
        ]
      },
      {
        "name": "$orderby",
        "in": "query",
        "description": "Order items by property values, see [OData
Sorting](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374629)",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string"
        },
        "enum": [
          "ID",
          "ID desc",
          "Description",
          "Description desc",
          "ReleaseDate",
          "ReleaseDate desc",
          "DiscontinuedDate",
          "DiscontinuedDate desc",
          "Rating",
          "Rating desc",
          "Price",
          "Price desc",
          "Currency",
          "Currency desc"
        ]
      }
    ],
```

The value of `responses` is a Responses Object, see **[OpenAPI]**. It contains a name/value pair for the success case (HTTP response code `200`) describing the structure of a successful response referencing the schema of the entity set's entity type in the global `definitions`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 14: GET OPERATION FOR AN ENTITY SET - RESPONSES

```
    "responses": {
      "200": {
        "description": "Retrieved entities",
        "schema": {
          "type": "object",
          "title": "Collection of Product",
          "properties": {
            "value": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/ODataDemo.Product"
              }
            }
          }
```



```
        }
      },
      "default": {
        "$ref": "#/responses/error"
        }
      }
    }
  },
```

4.6.1.2 Create an Entity

If the entity set allows inserts, the Path Item Object contains the keyword `post` with an Operation Object as value that describes the capabilities for creating new entities. The `tags` array of the Operation Object includes the entity set name.

The `parameters` array contains a Parameter Objects for the request body that references the schema of the entity set's entity type in the global `definitions`.

The `responses` object contains a name/value pair for the success case (HTTP response code `201`) describing the structure of the success response referencing the schema of the entity set's entity type in the global `definitions`. If the service supports the preference `return=minimal`, it contains a name/value pair for the HTTP response code `204`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 15: POST OPERATION FOR AN ENTITY SET

```
  "post": {
    "summary": "Add new entity to Products",
    "tags": [
      "Products"
    ],
    "parameters": [
      {
        "name": "Product",
        "in": "body",
        "description": "New entity",
        "schema": {
          "$ref": "#/definitions/ODataDemo.Product"
        }
      }
    ],
    "responses": {
      "201": {
        "description": "Created entity",
        "schema": {
          "$ref": "#/definitions/ODataDemo.Product"
        }
      },
      "default": {
        "$ref": "#/responses/error"
        }
      }
    }
  }
}
```

4.6.1.3 Retrieve an Entity

The Path Item Object for individual entities in the entity set contains the keyword `get` with an Operation Object as value that describes the capabilities for retrieving a single entity. The `tags` array of the Operation Object includes the entity set name.

EXAMPLE 16: GET OPERATION FOR AN INDIVIDUAL ENTITY

```
"/Products('{ID}')": {
  "get": {
    "summary": "Get entity from Products by key",
    "tags": [
      "Products"
    ],
```

The `parameters` array contains a Parameter Object for each key property, and it contains specific Parameter Objects for the system query options `$select` and `$expand` if these are allowed.

The Parameter Objects describing the allowed key values uses the same type mapping as described for primitive properties in section 4.7.1.1, with the exception that for key properties of type `Edm.Decimal` the `type` keyword has the value `"number"`.

They optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the key property.

EXAMPLE 17: GET OPERATION FOR AN INDIVIDUAL ENTITY - PARAMETERS

```
    "parameters": [
      {
        "name": "ID",
        "in": "path",
        "required": true,
        "description": "key: ID",
        "type": "string"
      },
      {
        "name": "$select",
        "in": "query",
        "description": "Select properties to be returned, see [OData
Select](http://docs.oasis-
open.org/odata/odata/v4.0/errata02/os/complete/part1-protocol/odata-
v4.0-errata02-os-part1-protocol-complete.html#_Toc406398297)",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string"
        },
        "enum": [
          "ID",
          "Description",
          "ReleaseDate",
          "DiscontinuedDate",
          "Rating",
          "Price",
          "Currency"
        ]
      },
      {
```



```
        "name": "$expand",
        "in": "query",
        "description": "Expand related entities, see [OData
Expand](http://docs.oasis-
open.org/odata/odata/v4.0/errata02/os/complete/part1-protocol/odata-
v4.0-errata02-os-part1-protocol-complete.html#_Toc406398298)",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string"
        },
        "enum": [
          "*",
          "Category",
          "Supplier"
        ]
      }
    ],
```

The `responses` object contains a name/value pair for the success case (HTTP response code `201`) describing the structure of the success response referencing the schema of the entity set's entity type in the global `definitions`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 18: GET OPERATION FOR AN INDIVIDUAL ENTITY - RESPONSES

```
    "responses": {
      "200": {
        "description": "Retrieved entity",
        "schema": {
          "$ref": "#/definitions/ODataDemo.Product"
        }
      },
      "default": {
        "$ref": "#/responses/error"
        }
      }
    }
  },
```

4.6.1.4 Update an Entity

If the entity set allows updates, the Path Item Object for individual entities in the entity set contains the keyword `patch` with an Operation Object as value that describes the capabilities for updating entities. The `tags` array of the Operation Object includes the entity set name.

The `parameters` array contains a Parameter Object for each key property, using the same type mapping as described for primitive properties in section 4.7.1.1, with the exception that for key properties of type `Edm.Decimal` the `type` keyword has the value `"number"`.

They optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the key property.

The `responses` object contains a name/value pair for the success case (HTTP response code `204`). If the service supports the preference `return=representation`, it contains a name/value pair for the HTTP response code `200` describing the structure of the success

response referencing the schema of the entity set's entity type in the global `definitions`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 19: PATCH OPERATION FOR AN INDIVIDUAL ENTITY

```
"patch": {
  "summary": "Update entity in Products",
  "tags": [
    "Products"
  ],
  "parameters": [
    {
      "name": "ID",
      "in": "path",
      "required": true,
      "description": "key: ID",
      "type": "string"
    },
    {
      "name": "Product",
      "in": "body",
      "description": "New property values",
      "schema": {
        "$ref": "#/definitions/ODataDemo.Product"
      }
    }
  ],
  "responses": {
    "204": {
      "description": "Success"
    },
    "default": {
      "$ref": "#/responses/error"
      }
    }
  }
},
```

4.6.1.5 Delete an Entity

If the entity set allows deletion of entities, the Path Item Object for individual entities in the entity set contains the keyword `delete` with an Operation Object as value that describes the capabilities for deleting entities. The `tags` array of the Operation Object includes the entity set name.

The `parameters` array contains a Parameter Object for each key property, using the same type mapping as described for primitive properties in section 4.7.1.1, with the exception that for key properties of type `Edm.Decimal` the `type` keyword has the value `"number"`.

They optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the key property.

The `responses` object contains a name/value pair for the success case (HTTP response code `204`). In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 20: DELETE OPERATION FOR AN INDIVIDUAL ENTITY

```
  "delete": {
    "summary": "Delete entity from Products",
    "tags": [
      "Products"
    ],
    "parameters": [
      {
        "name": "ID",
        "in": "path",
        "required": true,
        "description": "key: ID",
        "type": "string"
      },
      {
        "name": "If-Match",
        "in": "header",
        "description": "ETag",
        "type": "string"
      }
    ],
    "responses": {
      "204": {
        "description": "Success"
      },
      "default": {
        "$ref": "#/responses/error"
        }
      }
    }
  }
}
```

4.6.1.6 Invoke Bound Actions and Bound Functions

The Path Item Object for a bound action contains the keyword `post`, the Path Item Object for a bound function contains the keyword `get`. The value of the operation keyword is an Operation Object that describes how to invoke the action or function. The `tags` array of the Operation Object includes the entity set name.

EXAMPLE 21: ACTION BOUND TO ENTITY WITHIN A SET – SUMMARY AND TAGS

```
"/LeaveRequests({ID})/OData.Demo.Rejection": {
  "post": {
    "summary": "Invoke action Rejection",
    "tags": [
      "LeaveRequests"
    ],
```

For actions and functions bound to a single entity within an entity set the `parameters` array contains a Parameter Object for each key property, using the same type mapping as described for primitive properties in section 4.7.1.1, with the exception that for key properties of type `Edm.Decimal` the `type` keyword has the value `"number"`.

They optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the key property.

For bound actions the `parameters` array contains a Parameter Object describing the structure of the request body. Its `schema` value follows the rules for Schema Objects for complex types described in section 4.7.1, with one property per non-binding action parameter.

For bound functions the `parameters` array contains a Parameter Object for each non-binding parameter. Primitive parameters use the same type mapping as described for primitive properties in section 4.7.1.1, with the exception that for parameters of type `Edm.Decimal` the `type` keyword has the value `"number"`. Structured or collection-valued parameters are represented as a parameter alias in the path template and the `parameters` array contains a Parameter Object for the parameter alias as a query option of type `string`. The parameter `description` describes the format this URL-encoded JSON object or array, and/or reference to **[OData-URL]**, section 5.1.1.11.2.

These Parameter Objects optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the parameter.

Depending on the result type of the bound action or function the `parameters` array contains specific Parameter Objects for the allowed system query options.

EXAMPLE 22: ACTION BOUND TO ENTITY WITHIN A SET – PARAMETERS

```
"parameters": [
  {
    "name": "ID",
    "in": "path",
    "required": true,
    "description": "key: ID",
    "type": "integer",
    "format": "int32"
  },
  {
    "name": "body",
    "in": "body",
    "description": "Action parameters",
    "schema": {
      "type": "object",
      "properties": {
        "Reason": {
          "type": [
            "string",
            "null"
          ]
        }
      }
    }
  }
],
```

The `responses` object contains a name/value pair for the success case (HTTP response code `204`). In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 23: ACTION BOUND TO ENTITY WITHIN A SET – RESPONSES

```
"responses": {
  "204": {
```



```
        "description": "Success"
      },
      "default": {
        "$ref": "#/responses/error"
      }
    }
  }
}
```

4.6.2 Paths for Singletons

Each singleton is represented as a name/value pair whose name is the service-relative resource path of the singleton prepended with a forward slash, and whose value is Path Item Object describing the allowed operations on this singleton.

If the service defines bound actions or functions applicable to singleton, these are described as additional name/value pairs.

All operations for a singleton are tagged with the name of the entity set for consistent grouping in OpenAPI tools.

If the service defines bound actions or functions applicable to the singleton, these are described as additional name/value pairs with corresponding path templates for action/function invocation.

4.6.2.1 Retrieve a Singleton

The Path Item Object for a singleton contains the keyword `get` with an Operation Object as value that describes the capabilities for retrieving the singleton, unless the singleton is write-only. The `tags` array of the Operation Object includes the singleton's name.

The `parameters` array contains specific Parameter Objects for the system query options `$select` and `$expand` if these are allowed.

The `responses` object contains a name/value pair for the success case (HTTP response code `200`) describing the structure of the success response referencing the schema of the singleton's entity type in the global `definitions`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 24: GET OPERATION FOR A SINGLETON

```
"/MainSupplier": {
  "get": {
    "summary": "Get MainSupplier",
    "tags": [
      "MainSupplier"
    ],
    "parameters": [
      {
        "name": "$select",
        "in": "query",
        "description": "Select properties to be returned, see [OData
Select](http://docs.oasis-
open.org/odata/odata/v4.0/errata02/os/complete/part1-protocol/odata-
v4.0-errata02-os-part1-protocol-complete.html#_Toc406398297)",
```



```
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string"
        },
        "enum": [
          "ID",
          "Name",
          "Address",
          "Concurrency"
        ]
      },
      {
        "name": "$expand",
        "in": "query",
        "description": "Expand related entities, see [OData
Expand](http://docs.oasis-
open.org/odata/odata/v4.0/errata02/os/complete/part1-protocol/odata-
v4.0-errata02-os-part1-protocol-complete.html#_Toc406398298)",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string"
        },
        "enum": [
          "*",
          "Products"
        ]
      }
    ],
    "responses": {
      "200": {
        "description": "Retrieved entity",
        "schema": {
          "$ref": "#/definitions/ODataDemo.Supplier"
        }
      },
      "default": {
        "$ref": "#/responses/error"
      }
    }
  },
```

4.6.2.2 Update a Singleton

The Path Item Object for a singleton contains the keyword `patch` with an Operation Object as value that describes the capabilities for updating the singleton, unless the singleton is read-only. The `tags` array of the Operation Object includes the singleton's name.

The `responses` object contains a name/value pair for the success case (HTTP response code `204`). If the service supports the preference `return=representation`, it contains a name/value pair for the HTTP response code `200` describing the structure of the success response referencing the schema of the singleton's entity type in the global `definitions`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 25: PATCH OPERATION FOR A SINGLETON


```
  "patch": {
    "summary": "Update MainSupplier",
    "tags": [
      "MainSupplier"
    ],
    "parameters": [
      {
        "name": "Supplier",
        "in": "body",
        "description": "New property values",
        "schema": {
          "$ref": "#/definitions/ODataDemo.Supplier"
        }
      }
    ],
    "responses": {
      "204": {
        "description": "Success"
      },
      "default": {
        "$ref": "#/responses/error"
      }
    }
  }
}
```

4.6.3 Paths for Action Imports

Each action import is represented as a name/value pair whose name is the service-relative resource path of the action import prepended with a forward slash, and whose value is a Path Item Object containing the keyword `post` with an Operation Object as value that describes how to invoke the action import.

If the action import specifies the `EntitySet` attribute, the `tags` array of the Operation Object includes the entity set name.

The `parameters` array contains a Parameter Object describing the structure of the request body. Its `schema` value follows the rules for Schema Objects for complex types described in section 4.7.1, with one property per action parameter.

The `responses` object contains a name/value pair for the success case (HTTP response code `200`) describing the structure of the success response by referencing an appropriate schema in the global `definitions`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 26: ACTION IMPORT

```
"/IncreaseSalaries": {
  "post": {
    "summary": "Invoke action IncreaseSalaries",
    "tags": [
      "Service Operations"
    ],
    "parameters": [
      {
        "name": "body",
        "in": "body",
```



```
        "description": "Action parameters",
        "schema": {
          "type": "object",
          "properties": {
            "percentage": {
              "type": [
                "number",
                "string"
              ],
              "format": "decimal"
            }
          }
        }
      }
    ],
    "responses": {
      "204": {
        "description": "Success"
      },
      "default": {
        "$ref": "#/responses/error"
      }
    }
  }
}
```

4.6.4 Paths for Function Imports

Each function import is represented as one name/value pair per unbound function overload whose name is the service-relative resource path template of the function overload, and whose value is a Path Item Object containing the keyword `get` with an Operation Object as value that describes how to invoke the function overload.

If the function import specifies the `EntitySet` attribute, the `tags` array of the Operation Object includes the entity set name.

The `parameters` array contains a Parameter Object for each parameter of the function overload, and it contains specific Parameter Objects for the allowed system query options.

The Parameter Objects for primitive parameters use the same type mapping as described for primitive properties in section 4.7.1.1, with the exception that for parameters of type `Edm.Decimal` the `type` keyword has the value `"number"`.

Structured or collection-valued parameters are represented as a parameter alias in the path template and the `parameters` array contains a Parameter Object for the parameter alias as a query option of type `string`. The parameter `description` describes the format of this URL-encoded JSON object or array, and/or reference to **[OData-URL]**, section 5.1.1.11.2.

The `responses` object contains a name/value pair for the success case (HTTP response code `200`) describing the structure of the success response by referencing an appropriate schema in the global `definitions`. In addition it contains a `default` name/value pair for the OData error response referencing the global `responses`.

EXAMPLE 27: FUNCTION IMPORT


```
"/ProductsByRating(Rating={Rating})": {
  "get": {
    "summary": "Invoke function ProductsByRating",
    "tags": [
      "Products"
    ],
    "parameters": [
      {
        "name": "Rating",
        "in": "path",
        "required": true,
        "type": "integer",
        "format": "int32"
      }
    ],
    "responses": {
      "200": {
        "description": "Success",
        "schema": {
          "title": "Result",
          "type": "object",
          "properties": {
            "value": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/ODataDemo.Product"
              }
            }
          }
        }
      },
      "default": {
        "$ref": "#/responses/error"
        }
      }
    }
  }
}
```

4.7 Field `definitions`

The value of `definitions` is a Definitions Object, see **[OpenAPI]**. Each entity type, complex type, enumeration type, and type definition directly or indirectly used in the `paths` field is represented as a name/value pair of the Definitions Object.

The name of each pair is the namespace-qualified name of the type. It uses the namespace instead of the alias because these definitions can be reused by other CSDL documents, and aliases are document-local, so they are meaningless for referencing documents.

The value of each pair is a Schema Object, see **[OpenAPI]**.

EXAMPLE 28: DEFINITIONS

```
"definitions":{
  "ODataDemo.Product": …,
  "ODataDemo.Category": …,
  "ODataDemo.Supplier": …,
  "ODataDemo.Country": …,
  "ODataDemo.Address": …,
```



```
  "org.example.Employee": …,
  "org.example.Manager": …
}
```

4.7.1 Definitions for Entity Types and Complex Types

A structured type without a base type is represented as a Schema Object of type `object`.

A structured type with a base type is represented as a Schema Object that contains the keyword `allOf` whose value is an array with two items: a JSON Reference to the definition of the base type, and a Schema Object describing the derived type.

The Schema Object describing the (derived) type contains the standard OpenAPI Specification keywords appropriate for type `object`. It does not contain the `additionalProperties` keyword in order to allow additional properties beyond the declared properties. This is necessary for inheritance as well as instance annotations and dynamic properties.

It optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the structured type.

EXAMPLE 29: PRODUCT ENTITY TYPE

```
"ODataDemo.Product":{
  "type":"object",
  "properties": …,
  …
}
```

EXAMPLE 30: MANAGER ENTITY TYPE INHERITING FROM EMPLOYEE

```
"org.example.Manager":{
  "type":"object",
  "allOf":[
    {
      "$ref":"#/definitions/org.example.Employee"
    },
    {
      …
    }
  ]
}
```

4.7.1 Properties

Each structural property and navigation property is represented as a name/value pair of the standard OpenAPI `properties` object. The name is the property name, the value is a Schema Object describing the allowed values of the property.

The Schema Object for a property optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the property.

EXAMPLE 31: STRUCTURAL AND NAVIGATION PROPERTIES OF SUPPLIER ENTITY TYPE

```
"ODataDemo.Supplier":{
  …,
  "properties":{
```



```
    "ID":…,
    "Name":…,
    "Address":…,
    "Concurrency":…,
    "Products":…
  },
  …
}
```

4.7.1.1 Primitive Properties

Primitive properties of type `Edm.PrimitiveType`, `Edm.Stream`, and any of the `Edm.Geo*` types are represented as Schema Objects that are JSON References to definitions in the Definitions Object or a reusable definitions file such as odata-definitions.json in the examples folder of the **[OData-OpenAPI]** OASIS GitHub repository.

All other primitive properties are represented as a Schema Object with the following OpenAPI Specification types, formats, and validation keywords:

OData Primitive Type	OpenAPI Specification			Comment
	Type	Format	Keywords	
`Edm.Binary`	`string`	`base64url`	`maxLength`	OData-specific format `maxLength` is the maximum length of the base64-encoded string representation, i.e. 4*ceil(`MaxLength`/3)
`Edm.Boolean`	`boolean`			
`Edm.Byte`	`integer`	`uint8`		OData-specific format
`Edm.Date`	`string`	`date`		OpenAPI format
`Edm.DateTimeOffset`	`string`	`date-time`		OpenAPI format
`Edm.Decimal`	`number`, `string`	`decimal`	`minimum` `maximum` `multipleOf`	OData-specific format `string` is needed for IEEE754Compatible mode
`Edm.Double`	`number` [,`string`]	`double`		OpenAPI format with extended meaning `string` is needed for `-INF`, `INF`, and `NaN`
`Edm.Duration`	`string`	`duration`		OData-specific format

OData Primitive Type	OpenAPI Specification			Comment
	Type	Format	Keywords	
Edm.Guid	string	uuid		OData-specific format
Edm.Int16	integer	int16		OData-specific format
Edm.Int32	integer	int32		OpenAPI format
Edm.Int64	integer, string	int64		OpenAPI format string is needed for IEEE754Compatible mode
Edm.SByte	integer	int8		OData-specific format
Edm.Single	number [,string]	float		OpenAPI format with extended meaning string is needed for -INF, INF, and NaN
Edm.String	string		maxLength	Sequence of UTF-8 characters
Edm.TimeOfDay	string	time		OData-specific format

Properties of type Edm.Decimal and Edm.Int64 are represented as JSON strings if the format option IEEE754Compatible=true is specified, so they have to be declared with both number and string.

The scale of properties of type Edm.Decimal are represented with the OpenAPI Specification keyword multipleOf and a value of 10^{-scale}. The precision is represented with the maximum and minimum keywords and a value of $\pm(10^{precision-scale} - 10^{-scale})$ if the scale facet has a numeric value, and $\pm(10^{precision} - 1)$ if the scale is variable).

Properties of type Edm.Double and Edm.Single have special values for -INF, INF, and NaN that are represented as JSON strings, so they also have to be declared with both number and string. Services that do not support the special values -INF, INF, and NaN omit the string keyword.

The default value of a property is represented with the OpenAPI Specification keyword default.

EXAMPLE 32: NON-NULLABLE BOOLEAN PROPERTY WITH DEFAULT VALUE

```
"BooleanValue":{
  "type":"boolean",
  "default":false
}
```

EXAMPLE 33: NON-NULLABLE BINARY PROPERTY WITH BOTH MAXLENGTH AND BYTELENGTH


```
"BinaryValue":{
  "type":"string",
  "format":"base64url",
  "maxLength":44,
  "default":"T0RhdGE"
}
```

EXAMPLE 34: NON-NULLABLE INTEGER PROPERTY

```
"IntegerValue":{
  "type":"integer",
  "format":"int32",
  "default":-128
}
```

EXAMPLE 35: NON-NULLABLE FLOATING-POINT PROPERTIES: STRING REPRESENTATION FOR -INF, INF, AND NAN,

```
"DoubleValue":{
  "type":[
    "number",
    "string"
  ],
  "format":"double",
  "default":3.1415926535897931
},
"SingleValue":{
  "type":[
    "number",
    "string"
  ],
  "format":"float"
}
```

EXAMPLE 36: NON-NULLABLE DECIMAL PROPERTY WITH UNSPECIFIED PRECISION: NO MINIMUM AND MAXIMUM

```
"DecimalValue":{
  "type":[
    "number",
    "string"
  ],
  "format":"decimal",
  "default":34.95
}
```

EXAMPLE 37: NON-NULLABLE DECIMAL PROPERTY WITH PRECISION 15 AND SCALE 2

```
"FixedDecimalValue":{
  "type":[
    "number",
    "string"
  ],
  "format":"decimal",
  "multipleOf":0.01,
  "minimum":-999999999.99,
  "maximum":999999999.99
}
```

EXAMPLE 38: NULLABLE DECIMAL PROPERTY WITH PRECISION 15 AND SCALE 3

```
"NullableDecimalValue":{
  "type":["number","string","null"],
  "format":"decimal",
  "multipleOf":1e-3,
  "minimum":-999999999999.999,
```



```
  "maximum":999999999999.999
}
```

EXAMPLE 39: NON-NULLABLE STRING PROPERTY WITH MAXIMUM LENGTH OF 40 CHARACTERS

```
"StringValue":{
  "type":"string",
  "maxLength":40
  "default":"Say \"Hello\",\nthen go"
}
```

EXAMPLE 40: NON-NULLABLE DATE PROPERTY

```
"DateValue":{
  "type":"string",
  "format":"date",
  "default":"2012-12-03"
}
```

EXAMPLE 41: NON-NULLABLE TIMESTAMP PROPERTY WITH 7 FRACTIONAL DIGITS PRECISION

```
"DateTimeOffsetValue":{
  "type":"string",
  "format":"date-time",
  "default":"2012-12-03T07:16:23:00.0000000Z"
}
```

EXAMPLE 42: NULLABLE TIMESTAMP PROPERTY

```
"NullableDateTimeOffsetValue":{
  "type":["string","null"],
  "format":"date-time"
}
```

EXAMPLE 43: NON-NULLABLE TIMESTAMP PROPERTY WITH 12 FRACTIONAL DIGITS PRECISION

```
"DurationValue":{
  "type":"string",
  "format":"duration",
  "default":"P12DT23H59M59.999999999999S"
}
```

EXAMPLE 44: NON-NULLABLE TIME PROPERTY WITH 3 FRACTIONAL DIGITS PRECISION

```
"TimeOfDayValue":{
  "type":"string",
  "format":"time",
  "default":"07:59:59.999"
}
```

EXAMPLE 45: NON-NULLABLE GUID PROPERTY WITH DEFAULT VALUE

```
"GuidValue":{
  "type":"string",
  "format":"uuid",
  "default":"1234567-89ab-cdef-0123-456789abcdef"
}
```

EXAMPLE 46: NON-NULLABLE 8-BYTE INTEGER PROPERTY, ALLOWING FOR STRING REPRESENTATION IN IEEE754COMPATIBLE MODE

```
"Int64Value":{
  "type":[
    "integer",
    "string"
  ],
```



```
  "format":"int64",
  "default":0
}
```

EXAMPLE 47: NON-NULLABLE ENUMERATION PROPERTY

```
"ColorEnumValue":{
  "$ref":"#/definitions/Model1.Color",
  "default":"yellow"
},
```

EXAMPLE 48: NON-NULLABLE GEOGRAPHY-POINT PROPERTY

```
"GeographyPoint":{
  "$ref":"https://raw.githubusercontent.com/oasis-tcs/odata-
openapi/master/examples/odata-
definitions.json#/definitions/Edm.GeographyPoint",
  "default":{
    "type":"Point",
    "coordinates":[
      142.1,
      64.1
    ]
  }
}
```

EXAMPLE 49: NON-NULLABLE STREAM PROPERTY: NOT PART OF PAYLOAD IN VERSION 4.0

```
"StreamValue":{
  "$ref":"https://raw.githubusercontent.com/oasis-tcs/odata-
openapi/master/examples/odata-definitions.json#/definitions/Edm.Stream"
}
```

EXAMPLE 50: NON-NULLABLE PROPERTY TYPED WITH A TYPE DEFINITION

```
"TypeDefValue":{
  "$ref":"#/definitions/Model1.IntegerDecimal",
  "default":42
}
```

EXAMPLE 51: NON-NULLABLE PRIMITIVE PROPERTY WITH ABSTRACT TYPE, E.G. IN TERM DEFINITION

```
"PrimitiveValue":{
  "$ref":"https://raw.githubusercontent.com/oasis-tcs/odata-
openapi/master/examples/odata-
definitions.json#/definitions/Edm.PrimitiveType"
}
```

4.7.1.2 Complex Properties

Complex properties are represented as JSON References to the definition of the complex type, either as local references for types directly defined in the CSDL document, or as external references for types defined in referenced CSDL documents.

EXAMPLE 52: COMPLEX PROPERTY ADDRESS

```
"Address":{
  "$ref":"#/definitions/ODataDemo.Address"
},
```


4.7.1.3 Collection-Valued Structural Properties

Collection-valued structural and navigation properties are represented as Schema Objects of type `array`. The value of the `items` keyword is a Schema Object specifying the type of the items.

EXAMPLE 53: COLLECTION-VALUED COMPLEX PROPERTY TAGS

```
"Adresses":{
  "type":"array",
  "items":{
    "$ref":"#/definitions/ODataDemo.Address"
  }
}
```

EXAMPLE 54: NULLABLE COLLECTION-VALUED PRIMITIVE PROPERTY

```
"Dates":{
  "type":"array",
  "items":{
    "type":["string","null"],
    "format":"date"
  }
},
```

4.7.1.4 Navigation Properties

Navigation properties are represented similar to complex properties so that a standard OpenAPI Specification validator can validate the expanded representation of the navigation property.

EXAMPLE 55: COLLECTION-VALUED NAVIGATION PROPERTY PRODUCTS

```
"Products":{
  "type":"array",
  "items":{
    "$ref":"#/definitions/ODataDemo.Product"
  }
}
```

EXAMPLE 56: SINGLE-VALUED NAVIGATION PROPERTY CATEGORY

```
"Category":{
  "$ref":"#/definitions/ODataDemo.Category"
}
```

4.7.2 Definitions for Enumeration Types

An enumeration type is represented as a Schema Object of type `string` containing the OpenAPI Specification `enum` keyword. Its value is an array that contains a string with the member name for each enumeration member.

It optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the enumeration type.

EXAMPLE 57: ENUMERATION TYPE

```
"org.example.ShippingMethod":{
  "type":"string",
```



```
  "enum":[
    "FirstClass",
    "TwoDay",
    "Overnight"
  ]
}
```

4.7.3 Definitions for Type Definitions

A type definition is represented as a Schema Object describing the allowed values of the type definition using the same rules as described for primitive properties in section 4.7.1.1.

It optionally can contain the field `description`, whose value is the value of the unqualified annotation `Core.Description` of the type definition.

EXAMPLE 58: TYPE DEFINITIONS BASED ON EDM.STRING, EDM.DECIMAL AND EDM.DATETIMEOFFSET

```
"Model1.Text50":{
  "type":"string",
  "maxLength":50
},
"Model1.VariableDecimal":{
  "type":"number",
  "description":"A type definition"
},
"Model1.ExactTimestamp":{
  "type":"string",
  "format":"date-time"
}
```

4.8 Field `parameters`

The value of `parameters` is a Parameters Definitions Object, see **[OpenAPI]**. It allows defining query options and headers that can be reused across operations of the service.

It contains one name/value pair per OData system query option supported by the service.

EXAMPLE 59: REUSABLE QUERY OPTIONS

```
"parameters": {
  "top": {
    "name": "$top",
    "type": "integer",
    "in": "query",
    "description": "Show only the first n items, see [OData Paging -
Top](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374630))"
  },
  "skip": {
    "name": "$skip",
    "type": "integer",
    "in": "query",
    "description": "Skip the first n items, see [OData Paging - Skip](
http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374631)"
  },
  "count": {
    "name": "$count",
    "type": "boolean",
```



```
    "in": "query",
    "description": "Include count of items, see [OData Count](
http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374632)"
  },
  "filter": {
    "name": "$filter",
    "type": "string",
    "in": "query",
    "description": "Filter items by property values, see [OData
Filtering]( http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-
part1-protocol.html#_Toc445374625)"
  },
  "search": {
    "name": "$search",
    "type": "string",
    "in": "query",
    "description": "Search items by search phrases, see [OData
Searching]( http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-
part1-protocol.html#_Toc445374633)"
  }
}
```

4.9 Field `responses`

The value of `responses` is a Responses Definitions Object, see **[OpenAPI]**. It allows defining responses that can be reused across operations of the service.

It contains one name/value pair for the standard OData error response that is referenced from all operations of the service. The reusable error response in turn references a Schema Object in the Definitions Object or a reusable definitions file such as odata-definitions.json in the examples folder of the **[OData-OpenAPI]** OASIS GitHub repository.

EXAMPLE 60: REUSABLE ERROR RESPONSE

```
"responses": {
  "error": {
    "description": "Error",
    "schema": {
      "$ref": "#/definitions/odata.error"
    }
  }
}
```

5 Example

This is the shortened OAS document for the Products and Categories example metadata document in section 16.1 of **[OData-CSDL]**, listing only one entity set in full length.

EXAMPLE 61: PRODUCTS AND CATEGORIES EXAMPLE

```
{
  "swagger": "2.0",
  "info": {
    "title": "OData Service for namespace ODataDemo",
    "version": "",
```



```
    "description": "This OData service is located at
http://localhost/service-root/\n\n## Entity Data Model\n![ER
Diagram](http://yuml.me/diagram/class/[Product],[Product]-
0..1>[Category],[Product]-0..1>[Supplier],[Category],[Category]-
*>[Product],[Supplier],[Supplier]-*>[Product],[Country])\n\n##
References\n- [Org.OData.Core.V1](http://localhost/swagger-
ui/?url=http://localhost/examples/Org.OData.Core.V1.openapi.json)\n-
[Org.OData.Measures.V1](http://localhost/swagger-
ui/?url=http://localhost/examples/Org.OData.Measures.V1.openapi.json)"
  },
  "schemes": [
    "http"
  ],
  "host": "localhost",
  "basePath": "/service-root",
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "tags": [
    {
      "name": "Products"
    },
    {
      "name": "Categories",
      "description": "Product Categories"
    },
    {
      "name": "Suppliers"
    },
    {
      "name": "MainSupplier",
      "description": "Primary Supplier"
    },
    {
      "name": "Countries"
    }
  ],
  "definitions": {
    "ODataDemo.Product": {
      "type": "object",
      "properties": {
        "ID": {
          "type": "string"
        },
        "Description": {
          "type": [
            "string",
            "null"
          ]
        },
        "ReleaseDate": {
          "type": [
            "string",
            "null"
          ],
          "format": "date"
        },
        "DiscontinuedDate": {
          "type": [
            "string",
```



```
            "null"
          ],
          "format": "date"
        },
        "Rating": {
          "type": [
            "integer",
            "null"
          ],
          "format": "int32"
        },
        "Price": {
          "type": [
            "number",
            "string",
            "null"
          ],
          "format": "decimal",
          "multipleOf": 1
        },
        "Currency": {
          "type": [
            "string",
            "null"
          ],
          "maxLength": 3
        },
        "Category": {
          "$ref": "#/definitions/ODataDemo.Category"
        },
        "Supplier": {
          "$ref": "#/definitions/ODataDemo.Supplier"
        }
      },
      "title": "Product"
    },
    "ODataDemo.Category": {
      "type": "object",
      "properties": {
        "ID": {
          "type": "integer",
          "format": "int32"
        },
        "Name": {
          "type": "string"
        },
        "Products": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/ODataDemo.Product"
          }
        }
      },
      "title": "Category"
    },
    "ODataDemo.Supplier": {
      "type": "object",
      "properties": {
        "ID": {
          "type": "string"
        },
        "Name": {
          "type": [
```



```
            "string",
            "null"
          ]
        },
        "Address": {
          "$ref": "#/definitions/ODataDemo.Address"
        },
        "Concurrency": {
          "type": "integer",
          "format": "int32"
        },
        "Products": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/ODataDemo.Product"
          }
        }
      },
      "title": "Supplier"
    },
    "ODataDemo.Country": {
      "type": "object",
      "properties": {
        "Code": {
          "type": "string",
          "maxLength": 2
        },
        "Name": {
          "type": [
            "string",
            "null"
          ]
        }
      },
      "title": "Country"
    },
    "ODataDemo.Address": {
      "type": "object",
      "properties": {
        "Street": {
          "type": [
            "string",
            "null"
          ]
        },
        "City": {
          "type": [
            "string",
            "null"
          ]
        },
        "State": {
          "type": [
            "string",
            "null"
          ]
        },
        "ZipCode": {
          "type": [
            "string",
            "null"
          ]
        },
```



```
        "CountryName": {
          "type": [
            "string",
            "null"
          ]
        },
        "Country": {
          "$ref": "#/definitions/ODataDemo.Country"
        }
      },
      "title": "Address"
    }
  },
  "paths": {
    "/Products": {
      "get": {
        "summary": "Get entities from Products",
        "tags": [
          "Products"
        ],
        "parameters": [
          {
            "$ref": "#/parameters/top"
          },
          {
            "$ref": "#/parameters/skip"
          },
          {
            "$ref": "#/parameters/search"
          },
          {
            "$ref": "#/parameters/filter"
          },
          {
            "$ref": "#/parameters/count"
          },
          {
            "name": "$orderby",
            "in": "query",
            "description": "Order items by property values, see [OData
Sorting](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374629)",
            "type": "array",
            "uniqueItems": true,
            "items": {
              "type": "string"
            },
            "enum": [
              "ID",
              "ID desc",
              "Description",
              "Description desc",
              "ReleaseDate",
              "ReleaseDate desc",
              "DiscontinuedDate",
              "DiscontinuedDate desc",
              "Rating",
              "Rating desc",
              "Price",
              "Price desc",
              "Currency",
              "Currency desc"
            ]
```



```
          },
          {
            "name": "$select",
            "in": "query",
            "description": "Select properties to be returned, see
[OData Select](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-
part1-protocol.html#_Toc445374620)",
            "type": "array",
            "uniqueItems": true,
            "items": {
              "type": "string"
            },
            "enum": [
              "ID",
              "Description",
              "ReleaseDate",
              "DiscontinuedDate",
              "Rating",
              "Price",
              "Currency"
            ]
          },
          {
            "name": "$expand",
            "in": "query",
            "description": "Expand related entities, see [OData
Expand](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374621)",
            "type": "array",
            "uniqueItems": true,
            "items": {
              "type": "string"
            },
            "enum": [
              "*",
              "Category",
              "Supplier"
            ]
          }
        ],
        "responses": {
          "200": {
            "description": "Retrieved entities",
            "schema": {
              "type": "object",
              "title": "Collection of Product",
              "properties": {
                "value": {
                  "type": "array",
                  "items": {
                    "$ref": "#/definitions/ODataDemo.Product"
                  }
                }
              }
            }
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      },
      "post": {
        "summary": "Add new entity to Products",
```



```
        "tags": [
          "Products"
        ],
        "parameters": [
          {
            "name": "Product",
            "in": "body",
            "description": "New entity",
            "schema": {
              "$ref": "#/definitions/ODataDemo.Product"
            }
          }
        ],
        "responses": {
          "201": {
            "description": "Created entity",
            "schema": {
              "$ref": "#/definitions/ODataDemo.Product"
            }
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      }
    },
    "/Products('{ID}')": {
      "get": {
        "summary": "Get entity from Products by key",
        "tags": [
          "Products"
        ],
        "parameters": [
          {
            "name": "ID",
            "in": "path",
            "required": true,
            "description": "key: ID",
            "type": "string"
          },
          {
            "name": "$select",
            "in": "query",
            "description": "Select properties to be returned, see
[OData Select](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-
part1-protocol.html#_Toc445374620)",
            "type": "array",
            "uniqueItems": true,
            "items": {
              "type": "string"
            },
            "enum": [
              "ID",
              "Description",
              "ReleaseDate",
              "DiscontinuedDate",
              "Rating",
              "Price",
              "Currency"
            ]
          },
          {
            "name": "$expand",
```



```
            "in": "query",
            "description": "Expand related entities, see [OData
Expand](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374621)",
            "type": "array",
            "uniqueItems": true,
            "items": {
              "type": "string"
            },
            "enum": [
              "*",
              "Category",
              "Supplier"
            ]
          }
        ],
        "responses": {
          "200": {
            "description": "Retrieved entity",
            "schema": {
              "$ref": "#/definitions/ODataDemo.Product"
            }
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      },
      "patch": {
        "summary": "Update entity in Products",
        "tags": [
          "Products"
        ],
        "parameters": [
          {
            "name": "ID",
            "in": "path",
            "required": true,
            "description": "key: ID",
            "type": "string"
          },
          {
            "name": "Product",
            "in": "body",
            "description": "New property values",
            "schema": {
              "$ref": "#/definitions/ODataDemo.Product"
            }
          }
        ],
        "responses": {
          "204": {
            "description": "Success"
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      },
      "delete": {
        "summary": "Delete entity from Products",
        "tags": [
          "Products"
```



```
        ],
        "parameters": [
          {
            "name": "ID",
            "in": "path",
            "required": true,
            "description": "key: ID",
            "type": "string"
          },
          {
            "name": "If-Match",
            "in": "header",
            "description": "ETag",
            "type": "string"
          }
        ],
        "responses": {
          "204": {
            "description": "Success"
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      }
    },
    "/Categories": {
      …
    },
    "/Categories({ID})": {
      …
    },
    "/Suppliers": {
      …
    },
    "/Suppliers('{ID}')": {
      …
    },
    "/MainSupplier": {
      "get": {
        "summary": "Get MainSupplier",
        "tags": [
          "MainSupplier"
        ],
        "parameters": [
          {
            "name": "$select",
            "in": "query",
            "description": "Select properties to be returned, see
[OData Select](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-
part1-protocol.html#_Toc445374620)",
            "type": "array",
            "uniqueItems": true,
            "items": {
              "type": "string"
            },
            "enum": [
              "ID",
              "Name",
              "Address",
              "Concurrency"
            ]
          },
```



```
          {
            "name": "$expand",
            "in": "query",
            "description": "Expand related entities, see [OData
Expand](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374621)",
            "type": "array",
            "uniqueItems": true,
            "items": {
              "type": "string"
            },
            "enum": [
              "*",
              "Products"
            ]
          }
        ],
        "responses": {
          "200": {
            "description": "Retrieved entity",
            "schema": {
              "$ref": "#/definitions/ODataDemo.Supplier"
            }
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      },
      "patch": {
        "summary": "Update MainSupplier",
        "tags": [
          "MainSupplier"
        ],
        "parameters": [
          {
            "name": "Supplier",
            "in": "body",
            "description": "New property values",
            "schema": {
              "$ref": "#/definitions/ODataDemo.Supplier"
            }
          }
        ],
        "responses": {
          "204": {
            "description": "Success"
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      }
    },
    "/Countries": {
      …
    },
    "/ProductsByRating(Rating={Rating})": {
      "get": {
        "summary": "Invoke function ProductsByRating",
        "tags": [
          "Products"
        ],
```



```
        "parameters": [
          {
            "name": "Rating",
            "in": "path",
            "required": true,
            "type": "integer",
            "format": "int32"
          }
        ],
        "responses": {
          "200": {
            "description": "Success",
            "schema": {
              "title": "Result",
              "type": "object",
              "properties": {
                "value": {
                  "type": "array",
                  "items": {
                    "$ref": "#/definitions/ODataDemo.Product"
                  }
                }
              }
            }
          },
          "default": {
            "$ref": "#/responses/error"
          }
        }
      }
    }
  },
  "parameters": {
    "top": {
      "name": "$top",
      "in": "query",
      "description": "Show only the first n items, see [OData Paging -
Top](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374630)",
      "type": "integer"
    },
    "skip": {
      "name": "$skip",
      "in": "query",
      "description": "Skip the first n items, see [OData Paging -
Skip](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374631)",
      "type": "integer"
    },
    "count": {
      "name": "$count",
      "in": "query",
      "description": "Include count of items, see [OData
Count](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-
protocol.html#_Toc445374632)",
      "type": "boolean"
    },
    "filter": {
      "name": "$filter",
      "in": "query",
      "description": "Filter items by property values, see [OData
Filtering](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-
part1-protocol.html#_Toc445374625)",
```



```
      "type": "string"
    },
    "search": {
      "name": "$search",
      "in": "query",
      "description": "Search items by search phrases, see [OData
Searching](http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-
part1-protocol.html#_Toc445374633)",
      "type": "string"
    }
  },
  "responses": {
    "error": {
      "description": "Error",
      "schema": {
        "$ref": "https://raw.githubusercontent.com/oasis-tcs/odata-
openapi/master/examples/odata-
definitions.json#/definitions/odata.error"
      }
    }
  }
}
```


Appendix A. Acknowledgments

The contributions of the OASIS OData Technical Committee members, enumerated in **[OData-Protocol]**, are gratefully acknowledged.

Appendix B. Revision History

Revision	Date	Editor	Changes Made
Committee Note Draft 01	2016-11-28	Ralf Handl	Initial version

