OData Common Schema Definition Language (CSDL) JSON Representation Version 4.02

Committee Specification Draft 01

14 July 2023

This stage:

https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/csd01/odata-csdl-json-v4.02-csd01.md (Authoritative)
https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/csd01/odata-csdl-json-v4.02-csd01.html
https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/csd01/odata-csdl-json-v4.02-csd01.pdf

Previous stage:

N/A

Latest stage:

https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/odata-csdl-json-v4.02.md (Authoritative)
https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/odata-csdl-json-v4.02.html
https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/odata-csdl-json-v4.02.pdf

Technical Committee:

OASIS Open Data Protocol (OData) TC

Chairs:

Ralf Handl (ralf.handl@sap.com), SAP SE
Michael Pizzo (mikep@microsoft.com), Microsoft

Editors:

Ralf Handl (ralf.handl@sap.com), SAP SE
Michael Pizzo (mikep@microsoft.com), Microsoft
Heiko Theißen (heiko.theissen@sap.com), SAP SE

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- XML schemas: (list file names or directory name)
- Other parts (list titles and/or file names)
- (Note: Any normative computer language definitions that are part of the Work Product, such as XML instances, schemas and Java(TM) code, including fragments of such, must be (a) well formed and valid, (b) provided in separate plain text files, (c) referenced from the Work Product; and (d) where any definition in these separate files disagrees with the definition found in the specification, the definition in the separate file prevails. Remove this note before submitting for publication.)

Related work:

This specification replaces or supersedes:

- *OData Common Schema Definition Language (CSDL) JSON Representation Version 4.01*. Edited by Michael Pizzo, Ralf Handl, and Martin Zurmuehl. OASIS Standard. Latest stage: https://docs.oasis-open.org/odata/odata-csdl-json/v4.01/odata-csdl-json-v4.01.html.

This specification is related to:

- *OData Version 4.02*. Edited by Michael Pizzo, Ralf Handl, and Heiko Theißen. A multi-part Work Product that includes:
  - *OData Version 4.02 Part 1: Protocol*. Latest stage. https://docs.oasis-open.org/odata/odata/v4.02/odata-v4.02-part1-protocol.html
  - *OData Version 4.02 Part 2: URL Conventions*. Latest stage. https://docs.oasis-open.org/odata/odata/v4.02/odata-v4.02-part2-url-conventions.html

Abstract:

OData services are described by an Entity Model (EDM). The Common Schema Definition Language (CSDL) defines specific representations of the entity data model exposed by an OData service, using XML, JSON, and other formats. This document (OData CSDL JSON Representation) specifically defines the JSON representation of CSDL.

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 stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odata#technical.

TC members should send comments on this specification 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/.

This specification is provided under the RF on RAND Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/odata/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Key words:

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 **[RFC2119]** and **[RFC8174]** when, and only when, they appear in all capitals, as shown here.

Citation format:

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

[OData-CSDL-JSON-v4.02]

OData Common Schema Definition Language (CSDL) JSON Representation Version 4.02. Edited by Ralf Handl, Michael Pizzo, and Heiko Theißen. 14 July 2023. OASIS Committee Specification Draft 01. https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/csd01/odata-csdl-json-v4.02-csd01.html. Latest stage: https://docs.oasis-open.org/odata/odata-csdl-json/v4.02/odata-csdl-json-v4.02.html.

Notices

Table of Contents

1 Introduction

OData services are described in terms of an Entity Model. The Common Schema Definition Language (CSDL) defines a representation of the entity model exposed by an OData service using the JavaScript Object Notation (JSON)[, see[ ]{.apple-converted-space}]{style="color:black"} **[[RFC8259]]**.

This format is based on the OpenUI5 OData V4 Metadata JSON Format, see **[OpenUI5]**, with some extensions and modifications made necessary to fully cover OData CSDL Version 4.01.

1.1 Changes from earlier Versions

1.2 Glossary

1.2.1 Definitions of terms

1.2.2 Acronyms and abbreviations

1.2.3 Document conventions

Keywords defined by this specification use `this monospaced font`.

Some sections of this specification are illustrated with non-normative examples.

Example 1: text describing an example uses this paragraph style

```
Non-normative examples use this paragraph style.
```

All examples in this document are non-normative and informative only. Examples labeled with ⚠ contain advanced concepts or make use of keywords that are defined only later in the text, they can be skipped at first reading.

Representation-specific text is indented and marked with vertical lines.

> **Representation-Specific Headline**
>
> Normative representation-specific text

All other text is normative unless otherwise labeled.

Here is a customized command line which will generate HTML from this markdown file (named `odata-csdl-json-v4.02-csd01.md`). Line breaks are added for readability only:

```
pandoc -f gfm+tex_math_dollars+fenced_divs
       -t html
       -o odata-csdl-json-v4.02-csd01.html
       -c styles/markdown-styles-v1.7.3b.css
       -c styles/odata.css
       -s
       --mathjax
       --eol=lf
       --wrap=none
```



```
       --metadata pagetitle="OData Common Schema Definition Language (CSDL) JSON
Representation Version 4.02"
       odata-csdl-json-v4.02-csd01.md
```

This uses pandoc 3.1.2 from https://github.com/jgm/pandoc/releases/tag/3.1.2.

2 JSON Representation

OData CSDL JSON is a full representation of the OData Common Schema Definition Language in the JavaScript Object Notation (JSON) defined in **[RFC8259]**. It additionally follows the rules for "Internet JSON" (I-JSON) defined in **[RFC7493]** for e.g. objects, numbers, date values, and duration values.

It is an alternative to the CSDL XML representation defined in **[OData-CSDLXML]** and neither adds nor removes features.

2.1 Requesting the JSON Representation

The OData CSDL JSON representation can be requested using the `$format` query option in the request URL with the media type `application/json`, optionally followed by media type parameters, or the case-insensitive abbreviation `json` which MUST NOT be followed by media type parameters.

Alternatively, this representation can be requested using the `Accept` header with the media type `application/json`, optionally followed by media type parameters.

If specified, `$format` overrides any value specified in the `Accept` header.

The response MUST contain the `Content-Type` header with a value of `application/json`, optionally followed by media type parameters.

Possible media type parameters are:

- `IEEE754Compatible`
- `metadata`

The names and values of these parameters are case-insensitive.

2.1.1 Controlling the Representation of Numbers

The `IEEE754Compatible=true` parameter indicates that the service MUST serialize `Edm.Int64` and `Edm.Decimal` numbers as strings. This is in conformance with **[RFC7493]**. If not specified, or specified as `IEEE754Compatible=false`, all numbers MUST be serialized as JSON numbers.

This enables support for JavaScript numbers that are defined to be 64-bit binary format IEEE 754 values ECMAScript (see section 4.3.1.9) resulting in integers losing precision past 15 digits, and decimals losing precision due to the conversion from base 10 to base 2.

Responses that format `Edm.Int64` and `Edm.Decimal` values as strings MUST specify this parameter in the media type returned in the `Content-Type` header.

2.1.2 Controlling the Amount of Control Information

The representation of constant annotation values in CSDL JSON documents closely follows the representation of data defined in **[OData-JSON]**.

A client application can use the `metadata` format parameter in the `Accept` header when requesting a CSDL JSON document to influence how much control information will be included in the response.

Other `Accept` header parameters are orthogonal to the `metadata` parameter and are therefore not mentioned in this section.

2.1.2.1 `metadata=minimal`

The `metadata=minimal` format parameter indicates that the service SHOULD remove computable control information from the payload wherever possible.

This means that the `@type` control information is only included if the type of the containing object or targeted property cannot be heuristically determined, e.g. for

- Terms or term properties with an abstract declared type,
- Terms or term properties with a declared type that has derived types, or
- Dynamic properties of open types.

See **[OData-JSON]** for the exact rules.

2.1.2.2 `metadata=full`

The `metadata=full` format parameter indicates that the service MUST include all control information explicitly in the payload.

This means that the `@type` control information is included in annotation values except for primitive values whose type can be heuristically determined from the representation of the value, see **[OData-JSON]** for the exact rules.

2.1.2.3 `metadata=none`

The `metadata=none` format parameter indicates that the service SHOULD omit all control information.

2.2 Design Considerations

CSDL JSON documents are designed for easy and efficient lookup of model constructs by their name without having to know or guess what kind of model element it is. Thus, all primary model elements (entity types, complex types, type definitions, enumeration types, terms, actions, functions, and the entity container) are direct members of their schema, using the schema-unique name as the member name. Similarly, child elements of primary model elements (properties, navigation properties, enumeration type members, entity sets, singletons, action imports, and function imports) are direct members of the objects describing their parent model element, using their locally unique name as the member name.

To avoid name collisions, all fixed member names are prefixed with a dollar ($) sign and otherwise have the same name and capitalization as their counterparts in the CSDL XML representation

[OData-CSDLXML] (with one exception: the counterpart of the `EntitySet` element's `EntityType` attribute is `$Type`, to harmonize it with all other type references).

Additional fixed members introduced by this specification and without counterpart in **[OData-CSDLXML]** are also prefixed with a dollar (`$`) sign and use upper-camel-case names. One of these is `$Kind` which represents the kind of model element. Its value is the upper-camel-case local name of the XML element representing this kind of model element in **[OData-CSDLXML]**, e.g. `EntityType` or `NavigationProperty`.

While the XML representation of CSDL allows referencing model elements with alias-qualified names as well as with namespace-qualified names, this JSON representation requires the use of alias-qualified names if an alias is specified for an included or document-defined schema. Aliases are usually shorter than namespaces, so this reduces text size of the JSON document. Text size matters even if the actual HTTP messages are sent in compressed form because the decompressed form needs to be reconstructed, and clients not using a streaming JSON parser have to materialize the full JSON document before parsing.

To further reduce size the member `$Kind` is optional for structural properties as these are more common than navigation properties, and the member `$Type` is optional for string properties, parameters, and return types, as this type is more common than other primitive types.

In general, all members that have a default value SHOULD be omitted if they have the default value.

2.3 JSON Schema Definition

The structure of CSDL JSON documents can be verified with the JSON Schema OData-CSDL-Schema provided as an additional artifact of this prose specification. This schema only defines the shape of a well-formed CSDL JSON document but is not descriptive enough to define what a correct CSDL JSON document MUST be in every imaginable use case. This specification document defines additional rules that correct CSDL JSON documents MUST fulfill. In case of doubt on what makes a CSDL JSON document correct the rules defined in this specification document take precedence.

3 Entity Model

An OData service exposes a single entity model. This model may be distributed over several schemas, and these schemas may be distributed over several documents.

A service is defined by a single CSDL document which can be accessed by sending a `GET` request to `<serviceRoot>/$metadata`. This document is called the metadata document. It MAY reference other CSDL documents.

The metadata document contains a single entity container that defines the resources exposed by this service. This entity container MAY extend an entity container defined in a referenced document.

The *model* of the service consists of all CSDL constructs used in its entity containers.

The *scope* of a CSDL document is the document itself and all schemas included from directly referenced documents. All entity types, complex types and other named model elements *in scope* (that is, defined in the document itself or a schema of a directly referenced document) can be

accessed from a referencing document by their qualified names. This includes the built-in primitive and abstract types.

Referencing another document may alter the model defined by the referencing document. For instance, if a referenced document defines an entity type derived from an entity type in the referencing document, then an entity set of the service defined by the referencing document may return entities of the derived type. This is identical to the behavior if the derived type had been defined directly in the referencing document.

Note: referencing documents is not recursive. Only named model elements defined in directly referenced documents can be used within the schema. However, those elements may in turn include or reference model elements defined in schemas referenced by their defining schema.

3.1 Nominal Types

A nominal type has a name that MUST be a simple identifier. Nominal types are referenced using their qualified name. The qualified type name MUST be unique within a model as it facilitates references to the element from other parts of the model.

Names are case-sensitive, but service authors SHOULD NOT choose names that differ only in case.

3.2 Structured Types

Structured types are composed of other model elements. Structured types are common in entity models as the means of representing entities and structured properties in an OData service. Entity types and complex types are both structured types.

Structured Types are composed of zero or more structural properties and navigation properties.

Open entity types and open complex types allow properties to be added dynamically to instances of the open type.

3.3 Primitive Types

Structured types are composed of other structured types and primitive types. OData defines the following primitive types:

Type	Meaning
`Edm.Binary`	Binary data
`Edm.Boolean`	Binary-valued logic
`Edm.Byte`	Unsigned 8-bit integer
`Edm.Date`	Date without a time-zone offset
`Edm.DateTimeOffset`	Date and time with a time-zone offset, no leap seconds
`Edm.Decimal`	Numeric values with decimal representation

Type	Meaning
`Edm.Double`	IEEE 754 binary64 floating-point number (15-17 decimal digits)
`Edm.Duration`	Signed duration in days, hours, minutes, and (sub)seconds
`Edm.Guid`	16-byte (128-bit) unique identifier
`Edm.Int16`	Signed 16-bit integer
`Edm.Int32`	Signed 32-bit integer
`Edm.Int64`	Signed 64-bit integer
`Edm.SByte`	Signed 8-bit integer
`Edm.Single`	IEEE 754 binary32 floating-point number (6-9 decimal digits)
`Edm.Stream`	Binary data stream
`Edm.String`	Sequence of characters
`Edm.TimeOfDay`	Clock time 00:00-23:59:59.999999999999
`Edm.Geography`	Abstract base type for all Geography types
`Edm.GeographyPoint`	A point in a round-earth coordinate system
`Edm.GeographyLineString`	Line string in a round-earth coordinate system
`Edm.GeographyPolygon`	Polygon in a round-earth coordinate system
`Edm.GeographyMultiPoint`	Collection of points in a round-earth coordinate system
`Edm.GeographyMultiLineString`	Collection of line strings in a round-earth coordinate system
`Edm.GeographyMultiPolygon`	Collection of polygons in a round-earth coordinate system
`Edm.GeographyCollection`	Collection of arbitrary Geography values
`Edm.Geometry`	Abstract base type for all Geometry types
`Edm.GeometryPoint`	Point in a flat-earth coordinate system
`Edm.GeometryLineString`	Line string in a flat-earth coordinate system
`Edm.GeometryPolygon`	Polygon in a flat-earth coordinate system
`Edm.GeometryMultiPoint`	Collection of points in a flat-earth coordinate system

Type	Meaning
`Edm.GeometryMultiLineString`	Collection of line strings in a flat-earth coordinate system
`Edm.GeometryMultiPolygon`	Collection of polygons in a flat-earth coordinate system
`Edm.GeometryCollection`	Collection of arbitrary Geometry values

`Edm.Date` and `Edm.DateTimeOffset` follow XML-Schema-2 and use the proleptic Gregorian calendar, allowing the year `0000` (equivalent to 1 BCE) and negative years (year `-0001` being equivalent to 2 BCE etc.). The supported date range is service-specific and typically depends on the underlying persistency layer, e.g. SQL only supports years `0001` to `9999`.

`Edm.Decimal` with a `Scale` value of floating, `Edm.Double`, and `Edm.Single` allow the special numeric values `-INF`, `INF`, and `NaN`.

`Edm.Stream` is a primitive type that can be used as a property of an entity type or complex type, the underlying type for a type definition, or the binding parameter or return type of an action or function. `Edm.Stream`, or a type definition whose underlying type is `Edm.Stream`, cannot be used in collections or for non-binding parameters to functions or actions.

Some of these types allow facets, defined in section "Type Facets".

See rule `primitiveLiteral` in **[OData-ABNF]** for the representation of primitive type values in URLs and **[OData-JSON]** for the representation in requests and responses.

3.4 Built-In Abstract Types

The following built-in abstract types can be used within a model:

- `Edm.PrimitiveType`
- `Edm.ComplexType`
- `Edm.EntityType`
- `Edm.Untyped`

Conceptually, these are the abstract base types for primitive types (including type definitions and enumeration types), complex types, entity types, or any type or collection of types, respectively, and can be used anywhere a corresponding concrete type can be used, except:

- `Edm.EntityType`
  - cannot be used as the type of a singleton in an entity container because it doesn't define a structure, which defeats the purpose of a singleton.
  - cannot be used as the type of an entity set because all entities in an entity set must have the same key fields to uniquely identify them within the set.
  - cannot be the base type of an entity type or complex type.
- `Edm.ComplexType`
  - cannot be the base type of an entity type or complex type.
- `Edm.PrimitiveType`

    - cannot be used as the type of a key property of an entity type or as the underlying type of an enumeration type.
    - cannot be used as the underlying type of a type definition in a CSDL document with a version of `4.0`.
    - can be used as the underlying type of a type definition in a CSDL document with a version of `4.01` or greater.
- `Edm.Untyped`
    - cannot be returned in a payload with an `OData-Version` header of `4.0`. Services should treat untyped properties as dynamic properties in `4.0` payloads.
    - cannot be used as the type of a key property of an entity type.
    - cannot be the base type of an entity type or complex type.
    - cannot be used as the underlying type of a type definition or enumeration type.
- `Collection(Edm.PrimitiveType)`
    - cannot be used as the type of a property or term.
    - cannot be used as the type of a parameter or the return type of an action or function.
- `Collection(Edm.Untyped)`
    - cannot be returned in a payload with an `OData-Version` header of `4.0`. Services should treat untyped properties as dynamic properties in `4.0` payloads.

3.5 Built-In Types for defining Vocabulary Terms

Vocabulary terms can, in addition, use

- `Edm.AnnotationPath`
- `Edm.PropertyPath`
- `Edm.NavigationPropertyPath`
- `Edm.AnyPropertyPath` (`Edm.PropertyPath` or `Edm.NavigationPropertyPath`)
- `Edm.ModelElementPath` (any model element, including `Edm.AnnotationPath`, `Edm.NavigationPropertyPath`, and `Edm.PropertyPath`)

as the type of a primitive term, or the type of a property of a complex type (recursively) that is exclusively used as the type of a term. See section "Path Expressions" for details.

3.6 Annotations

Many parts of the model can be decorated with additional information using annotations. Annotations are identified by their term name and an optional qualifier that allows applying the same term multiple times to the same model element.

A model element MUST NOT specify more than one annotation for a given combination of term and qualifier.

4 CSDL JSON Document

Document Object

A CSDL JSON document consists of a single JSON object. This document object MUST contain the member `$Version`.

The document object MAY contain the member `$Reference` to reference other CSDL documents.

It also MAY contain members for schemas.

If the CSDL JSON document is the metadata document of an OData service, the document object MUST contain the member `$EntityContainer`.

$Version

The value of `$Version` is a string containing either 4.0 or 4.01.

$EntityContainer

The value of `$EntityContainer` is value is the namespace-qualified name of the entity container of that service. This is the only place where a model element MUST be referenced with its namespace-qualified name and use of the alias-qualified name is not allowed.

Example 2:

```
{
  "$Version": "4.01",
  "$EntityContainer": "org.example.DemoService",
  ...
}
```

4.1 Reference

A reference to an external CSDL document allows to bring part of the referenced document's content into the scope of the referencing document.

A reference MUST specify a URI that uniquely identifies the referenced document, so two references MUST NOT specify the same URI. The URI SHOULD be a URL that locates the referenced document. If the URI is not dereferencable it SHOULD identify a well-known schema. The URI MAY be absolute or relative URI; relative URLs are relative to the URL of the document containing the reference, or relative to a base URL specified in a format-specific way.

A reference MAY be annotated.

The `Core.SchemaVersion` annotation, defined in **[OData-VocCore]**, MAY be used to indicate a particular version of the referenced document. If the `Core.SchemaVersion` annotation is present, the `$schemaversion` system query option, defined **[OData-Protocol]**, SHOULD be used when retrieving the referenced schema document.

$Reference

The value of `$Reference` is an object that contains one member per referenced CSDL document. The name of the pair is a URI for the referenced document. The URI MAY be relative to the document containing the `$Reference`. The value of each member is a reference object.

Reference Object

The reference object MAY contain the members `$Include` and `$IncludeAnnotations` as well as annotations.

Example 3: references to other CSDL documents

```
{
  ...
  "$Reference": {
    "http://vocabs.odata.org/capabilities/v1": {
      ...
    },
    "http://vocabs.odata.org/core/v1": {
      ...
    },
    "http://example.org/display/v1": {
      ...
    }
  },
  ...
}
```

4.2 Included Schema

A reference MAY include zero or more schemas from the referenced document.

The included schemas are identified via their namespace. The same namespace MUST NOT be included more than once, even if it is declared in more than one referenced document.

When including a schema, a simple identifier value MAY be specified as an alias for the schema that is used in qualified names instead of the namespace. For example, an alias of `display` might be assigned to the namespace `org.example.vocabularies.display`. An alias-qualified name is resolved to a fully qualified name by examining aliases for included schemas and schemas defined within the document.

> If an included schema specifies an alias, the alias MUST be used in qualified names throughout the document to identify model elements of the included schema. A mixed use of namespace-qualified names and alias-qualified names is not allowed.

Aliases are document-global, so all schemas defined within or included into a document MUST have different aliases, and aliases MUST differ from the namespaces of all schemas defined within or included into a document.

The alias MUST NOT be one of the reserved values `Edm`, `odata`, `System`, or `Transient`.

An alias is only valid within the document in which it is declared; a referencing document may define its own aliases for included schemas.

> **`$Include`**
>
> The value of `$Include` is an array. Array items are objects that MUST contain the member `$Namespace` and MAY contain the member `$Alias`.
>
> The item objects MAY contain annotations.
>
> **`$Namespace`**
>
> The value of `$Namespace` is a string containing the namespace of the included schema.
>
> **`$Alias`**
>
> The value of `$Alias` is a string containing the alias for the included schema.

Example 4: references to entity models containing definitions of vocabulary terms

```
{
  ...
  "$Reference": {
    "http://vocabs.odata.org/capabilities/v1": {
      "$Include": [
        {
          "$Namespace": "Org.OData.Capabilities.V1",
          "$Alias": "Capabilities"
        }
      ]
    },
    "http://vocabs.odata.org/core/v1": {
      "$Include": [
        {
          "$Namespace": "Org.OData.Core.V1",
          "$Alias": "Core",
          "@Core.DefaultNamespace": true
        }
      ]
    },
    "http://example.org/display/v1": {
      "$Include": [
        {
          "$Namespace": "org.example.display",
          "$Alias": "UI"
        }
      ]
    }
  },
```



```
  ...
}
```

4.3 Included Annotations

In addition to including whole schemas with all model constructs defined within that schema, annotations can be included with more flexibility.

Annotations are selectively included by specifying the namespace of the annotations' term. Consumers can opt not to inspect the referenced document if none of the term namespaces is of interest for the consumer.

In addition, the qualifier of annotations to be included MAY be specified. For instance, a service author might want to supply a different set of annotations for various device form factors. If a qualifier is specified, only those annotations from the specified term namespace with the specified qualifier (applied to a model element of the target namespace, if present) SHOULD be included. If no qualifier is specified, all annotations within the referenced document from the specified term namespace (taking into account the target namespace, if present) SHOULD be included.

The qualifier also provides consumers insight about what qualifiers are present in the referenced document. If the consumer is not interested in that particular qualifier, the consumer can opt not to inspect the referenced document.

In addition, the namespace of the annotations' target MAY be specified. If a target namespace is specified, only those annotations which apply a term form the specified term namespace to a model element of the target namespace (with the specified qualifier, if present) SHOULD be included. If no target namespace is specified, all annotations within the referenced document from the specified term namespace (taking into account the qualifier, if present) SHOULD be included.

The target namespace also provides consumers insight about what namespaces are present in the referenced document. If the consumer is not interested in that particular target namespace, the consumer can opt not to inspect the referenced document.

> **`$IncludeAnnotations`**
>
> The value of `$IncludeAnnotations` is an array. Array items are objects that MUST contain the member `$TermNamespace` and MAY contain the members `$Qualifier` and `$TargetNamespace`.
>
> **`$TermNamespace`**
>
> The value of `$TermNamespace` is a namespace.
>
> **`$Qualifier`**
>
> The value of `$Qualifier` is a simple identifier.
>
> **`$TargetNamespace`**
>
> The value of `$TargetNamespace` is a namespace.

Example 5: reference documents that contain annotations


```
{
  ...
  "$Reference": {
    "http://odata.org/ann/b": {
      "$IncludeAnnotations": [
        {
          "$TermNamespace": "org.example.validation"
        },
        {
          "$TermNamespace": "org.example.display",
          "$Qualifier": "Tablet"
        },
        {
          "$TermNamespace": "org.example.hcm",
          "$TargetNamespace": "com.example.Sales"
        },
        {
          "$TermNamespace": "org.example.hcm",
          "$Qualifier": "Tablet",
          "$TargetNamespace": "com.example.Person"
        }
      ]
    }
  },
  ...
}
```

The following annotations from `http://odata.org/ann/b` *are included:*

- *Annotations that use a term from the* `org.example.validation` *namespace, and*
- *Annotations that use a term from the* `org.example.display` *namespace and specify a* `Tablet` *qualifier and*
- *Annotations that apply a term from the* `org.example.hcm` *namespace to an element of the* `com.example.Sales` *namespace and*
- *Annotations that apply a term from the* `org.example.hcm` *namespace to an element of the* `com.example.Person` *namespace and specify a* `Tablet` *qualifier.*

Appendix A. References

This appendix contains the normative and informative references that are used in this document.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

A.1 Normative References

The following documents are referenced in such a way that some or all of their content constitutes requirements of this document.

(Reference sources: For references to IETF RFCs, use the approved citation formats at: https://docs.oasis-open.org/templates/ietf-rfc-list/ietf-rfc-list.html. For references to W3C Recommendations, use the approved citation formats at: https://docs.oasis-open.org/templates/w3c-recommendations-list/w3c-recommendations-list.html. Remove this note before submitting for publication.)

[OData-v4.02]

- *OData Version 4.02*. Edited by Michael Pizzo, Ralf Handl, and Heiko Theißen. A multi-part Work Product that includes:
  - *OData Version 4.02 Part 1: Protocol*. Latest stage. https://docs.oasis-open.org/odata/odata/v4.02/odata-v4.02-part1-protocol.html
  - *OData Version 4.02 Part 2: URL Conventions*. Latest stage. https://docs.oasis-open.org/odata/odata/v4.02/odata-v4.02-part2-url-conventions.html

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.
http://www.rfc-editor.org/info/rfc2119.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.
http://www.rfc-editor.org/info/rfc8174.

[RFC6570]

Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.
http://tools.ietf.org/html/rfc6570.

[RFC7493]

Bray, T., Ed., "The I-JSON Message Format", RFC7493, March 2015.
https://tools.ietf.org/html/rfc7493.

[RFC8259]

Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 8259, December 2017.
http://tools.ietf.org/html/rfc8259.

[ECMAScript]

ECMAScript 2016 Language Specification, 7th Edition. June 2016. Standard ECMA-262.
http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[EPSG]

European Petroleum Survey Group (EPSG). http://www.epsg.org/.

[OData-ABNF]

OData ABNF Construction Rules Version 4.01.
See link in "Additional artifacts" section on cover page.

[OData-CSDL-Schema]

OData CSDL JSON Schema.
See link in "Related work" section on cover page.

[OData-CSDLXML]

OData Common Schema Definition Language (CSDL) XML Representation Version 4.01.
See link in "Related work" section on cover page.

[OData-JSON]

OData JSON Format Version 4.01.
See link in "Related work" section on cover page.

[OData-Protocol]

OData Version 4.01 Part 1: Protocol.
See link in "Related work" section on cover page.

[OData-URL]

OData Version 4.01 Part 2: URL Conventions.
See link in "Related work" section on cover page.

[OData-VocCore]

OData Vocabularies Version 4.0: Core Vocabulary.
See link in "Related work" section on cover page.

[OData-VocMeasures]

OData Vocabularies Version 4.0: Measures Vocabulary.
See link in "Related work" section on cover page.

[OData-VocValidation]

OData Vocabularies Version 4.0: Validation Vocabulary.
See link in "Related work" section on cover page.

[XML-Schema-2]

W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. D. Peterson, S. Gao, C. M. Sperberg-McQueen, H. S. Thompson, P. V. Biron, A. Malhotra, Editors, W3C Recommendation, 5 April 2012.
http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/. Latest version available at http://www.w3.org/TR/xmlschema11-2/.

A.2 Informative References

[RFC3552]

Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003 https://www.rfc-editor.org/info/rfc3552.

[OpenUI5]

OpenUI5 Version 1.40.10 - OData V4 Metadata JSON Format.
https://openui5.hana.ondemand.com/1.40.10/#docs/guide/87aac894a40640f89920d7b2a414499b.html.

Appendix B. Table of JSON Objects and Members

Appendix C. Acknowledgments

Note: A Work Product approved by the TC must include a list of people who participated in the development of the Work Product. This is generally done by collecting the list of names in this appendix. This list shall be initially compiled by the Chair, and any Member of the TC may add or remove their names from the list by request. Remove this note before submitting for publication.

C.1 Special Thanks

Substantial contributions to this document from the following individuals are gratefully acknowledged:

Participant Name, Affiliation or "Individual Member"

C.2 Participants

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

OpenC2 TC Members:

First Name	Last Name	Company
Philippe	Alman	Something Networks
Alex	Amirnovman	Company B
Kris	Anderman	Mini Micro
Darren	Anstman	Big Networks

Appendix D. Revision History

Revision	Date	Editor	Changes Made
specname-v1.0-wd01	yyyy-mm-dd	Editor Name	Initial working draft

Appendix E. Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specification, OASIS Standard, or Approved Errata).

[OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.]

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by

an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.]