

OData JSON Format Version 4.01

Committee Specification Draft 01 /
Public Review Draft 01

08 December 2016

Specification URIs

This version:

<http://docs.oasis-open.org/odata/odata-json-format/v4.01/csprd01/odata-json-format-v4.01-csprd01.docx> (Authoritative)

<http://docs.oasis-open.org/odata/odata-json-format/v4.01/csprd01/odata-json-format-v4.01-csprd01.html>

<http://docs.oasis-open.org/odata/odata-json-format/v4.01/csprd01/odata-json-format-v4.01-csprd01.pdf>

Previous version:

N/A

Latest version:

<http://docs.oasis-open.org/odata/odata-json-format/v4.01/odata-json-format-v4.01.docx> (Authoritative)

<http://docs.oasis-open.org/odata/odata-json-format/v4.01/odata-json-format-v4.01.html>

<http://docs.oasis-open.org/odata/odata-json-format/v4.01/odata-json-format-v4.01.pdf>

Technical Committee:

[OASIS Open Data Protocol (OData) TC](https://www.oasis-open.org/committees/odata/)

Chairs:

Ralf Handl (ralf.handl@sap.com), [SAP SE](http://www.sap.com/)

Ram Jeyaraman (Ram.Jeyaraman@microsoft.com), [Microsoft](http://www.microsoft.com/)

Editors:

Michael Pizzo (mikep@microsoft.com), [Microsoft](http://www.microsoft.com/)

Ralf Handl (ralf.handl@sap.com), [SAP SE](http://www.sap.com/)

Mark Biamonte (mark.biamonte@progress.com), [Progress Software](http://www.progress.com/)

Related work:

This specification replaces or supersedes:

* *OData JSON Format Version 4.0*. Edited by Ralf Handl, Michael Pizzo, and Mark Biamonte. 24 February 2014. OASIS Standard. <http://docs.oasis-open.org/odata/odata-json-format/v4.0/os/odata-json-format-v4.0-os.html>. Latest version: <http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.html>.

This specification is related to:

* *OData Version 4.01*. Edited by Michael Pizzo, Ralf Handl, and Martin Zurmuehl. A multi-part Work Product which includes:
	+ *OData Version 4.01. Part 1: Protocol*. Latest version: <http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html>.
	+ *OData Version 4.01. Part 2: URL Conventions*. Latest version: <http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part2-url-conventions.html>.
	+ *OData Version 4.01. Part 3: Common Schema Definition Language (CSDL)*. Latest version: <http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part3-csdl.html>.
	+ OData ABNF Construction Rules Version 4.01 and OData ABNF Test Cases Version 4.01. <http://docs.oasis-open.org/odata/odata/v4.01/csprd01/abnf/>.
* *OData Vocabularies Version 4.0.* Edited by Mike Pizzo, Ralf Handl, and Ram Jeyaraman. Latest version: <http://docs.oasis-open.org/odata/odata-vocabularies/v4.0/odata-vocabularies-v4.0.html>.
* *OData Common Schema Definition Language (CSDL) XML Representation Version 4.01*. Edited by Michael Pizzo, Ralf Handl, and Martin Zurmuehl. Latest version: <http://docs.oasis-open.org/odata/odata-csdl-xml/v4.01/odata-csdl-xml-v4.01.html>.
* *OData Atom Format Version 4.0*. Edited by Martin Zurmuehl, Michael Pizzo, and Ralf Handl. Latest version. <http://docs.oasis-open.org/odata/odata-atom-format/v4.0/odata-atom-format-v4.0.html>.

Abstract:

The Open Data Protocol (OData) for representing and interacting with structured content is comprised of a set of specifications. The core specification for the protocol is in OData Version 4.01 Part 1: Protocol. This document extends the core specification by defining representations for OData requests and responses using a JSON format.

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. 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](https://www.oasis-open.org/committees/comments/index.php?wg_abbrev=odata)” button on the TC’s web page at <https://www.oasis-open.org/committees/odata/>.

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 Technical Committee web page (<https://www.oasis-open.org/committees/odata/ipr.php>).

Citation format:

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

[OData-JSON-Format-v4.01]

*OData JSON Format Version 4.01*. Edited by Ralf Handl, Michael Pizzo, and Mark Biamonte. 08 December 2016. OASIS Committee Specification Draft 01 / Public Review Draft 01. <http://docs.oasis-open.org/odata/odata-json-format/v4.01/csprd01/odata-json-format-v4.01-csprd01.html>. Latest version: <http://docs.oasis-open.org/odata/odata-json-format/v4.01/odata-json-format-v4.01.html>.

Notices

Copyright © OASIS Open 2016. All Rights Reserved.

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](https://www.oasis-open.org/policies-guidelines/ipr) 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.

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 Committee Specification or OASIS Standard, 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 specification.

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 specification 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 specification. 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 document 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 Committee Specification or OASIS Standard, 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.

The name "OASIS" is a trademark of [OASIS](https://www.oasis-open.org/), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

Table of Contents

[1 Introduction 6](#_Toc471396250)

[1.1 Terminology 6](#_Toc471396251)

[1.2 Normative References 6](#_Toc471396252)

[1.3 Typographical Conventions 7](#_Toc471396253)

[2 JSON Format Design 8](#_Toc471396254)

[3 Requesting the JSON Format 9](#_Toc471396255)

[3.1 Controlling the Amount of Control Information in Responses 9](#_Toc471396256)

[3.1.1 metadata=minimal (odata.metadata=minimal) 9](#_Toc471396257)

[3.1.2 metadata=full (odata.metadata=full) 10](#_Toc471396258)

[3.1.3 metadata=none (odata.metadata=none) 10](#_Toc471396259)

[3.2 Controlling the Representation of Numbers 10](#_Toc471396260)

[4 Common Characteristics 12](#_Toc471396261)

[4.1 Header Content-Type 12](#_Toc471396262)

[4.2 Message Body 12](#_Toc471396263)

[4.3 Relative URLs 12](#_Toc471396264)

[4.4 Payload Ordering Constraints 13](#_Toc471396265)

[4.5 Control Information 14](#_Toc471396266)

[4.5.1 Annotation context (odata.context) 14](#_Toc471396267)

[4.5.2 Annotation metadataEtag (odata.metadataEtag) 14](#_Toc471396268)

[4.5.3 Annotation type (odata.type) 14](#_Toc471396269)

[4.5.4 Annotation count (odata.count) 16](#_Toc471396270)

[4.5.5 Annotation nextLink (odata.nextLink) 16](#_Toc471396271)

[4.5.6 Annotation delta (odata.delta) 16](#_Toc471396272)

[4.5.7 Annotation deltaLink (odata.deltaLink) 16](#_Toc471396273)

[4.5.8 Annotation id (odata.id) 16](#_Toc471396274)

[4.5.9 Annotation editLink and readLink (odata.editLink and odata.readLink) 16](#_Toc471396275)

[4.5.10 Annotation etag (odata.etag) 17](#_Toc471396276)

[4.5.11 Annotation navigationLink and associationLink (odata.navigationLink and odata.associationLink) 17](#_Toc471396277)

[4.5.12 Annotation media\* (odata.media\*) 17](#_Toc471396278)

[5 Service Document 19](#_Toc471396279)

[6 Entity 21](#_Toc471396280)

[7 Structural Property 22](#_Toc471396281)

[7.1 Primitive Value 22](#_Toc471396282)

[7.2 Complex Value 23](#_Toc471396283)

[7.3 Collection of Primitive Values 23](#_Toc471396284)

[7.4 Collection of Complex Values 23](#_Toc471396285)

[7.5 Untyped Value 24](#_Toc471396286)

[8 Navigation Property 25](#_Toc471396287)

[8.1 Navigation Link 25](#_Toc471396288)

[8.2 Association Link 25](#_Toc471396289)

[8.3 Expanded Navigation Property 25](#_Toc471396290)

[8.4 Deep Insert 26](#_Toc471396291)

[8.5 Bind Operation 26](#_Toc471396292)

[9 Stream Property 28](#_Toc471396293)

[10 Media Entity 29](#_Toc471396294)

[11 Individual Property or Operation Response 30](#_Toc471396295)

[12 Collection of Entities 31](#_Toc471396296)

[13 Entity Reference 32](#_Toc471396297)

[14 Delta Payload 33](#_Toc471396298)

[14.1 Added/Changed Entity 34](#_Toc471396299)

[14.2 Deleted Entity 35](#_Toc471396300)

[14.3 Added Link 36](#_Toc471396301)

[14.4 Deleted Link 36](#_Toc471396302)

[15 Bound Function 38](#_Toc471396303)

[16 Bound Action 40](#_Toc471396304)

[17 Action Invocation 42](#_Toc471396305)

[18 Instance Annotations 43](#_Toc471396306)

[18.1 Annotate a JSON Object 43](#_Toc471396307)

[18.2 Annotate a JSON Array or Primitive 43](#_Toc471396308)

[19 Error Response 44](#_Toc471396309)

[20 Extensibility 45](#_Toc471396310)

[21 Security Considerations 46](#_Toc471396311)

[22 Conformance 47](#_Toc471396312)

[Appendix A. Acknowledgments 50](#_Toc471396313)

[Appendix B. Revision History 51](#_Toc471396314)

# Introduction

The OData protocol is comprised of a set of specifications for representing and interacting with structured content. The core specification for the protocol is in [**[OData-Protocol]**](#odata); this document is an extension of the core protocol. This document defines representations for the OData requests and responses using the JavaScript Object Notation (JSON), see [**[RFC7159]**](#rfc7159).

An OData JSON payload may represent:

* a [single primitive value](#_Primitive_Value)
* a [collection of primitive values](#CollectionOfPrimitive)
* a [single complex type value](#ComplexValue)
* a [collection of complex type values](#_Collection_of_Complex)
* a [single entity](#_Entity) or [entity reference](#ResourceReference)
* a [collection of entities](#CollectionOfEntities) or [entity references](#_Entity_Reference)
* a [collection of changes](#DeltaResponse)
* a [service document](#_Representing_the_Service) describing the top-level resources exposed by the service
* an [error](#_Representing_Errors_in).

## Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [**[RFC2119]**](#rfc2119).

## Normative References

[GeoJSON] Howard Butler, Martin Daly, Alan Doyle, Sean Gillies, Stefan Hagen and Tim Schaub, "The GeoJSON Format", RFC 7946, August 2016.
<http://tools.ietf.org/html/rfc7946>.

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

[OData-ABNF] *OData ABNF Construction Rules Version 4.0*.
See link in “Related work” section on cover page.

[OData-CSDL] OData Version 4.0 Part 3: Common Schema Definition Language (CSDL).
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-Protocol] *OData Version 4.0 Part 1: Protocol*.
See link in “Related work” section on cover page.

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

**[****OData-VocCap]** *OData Vocabularies Version 4.0: Capabilities Vocabulary.*
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

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. <https://tools.ietf.org/html/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, IETF RFC3986, January 2005. <https://tools.ietf.org/html/rfc3986>.

[RFC3987] Duerst, M. and, M. Suignard, “Internationalized Resource Identifiers (IRIs)”, RFC 3987, January 2005. <https://tools.ietf.org/html/rfc3987>.

**[****RFC4648]** Josefsson, S,, “The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006. <https://tools.ietf.org/html/rfc4648>.

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

[RFC5646] Phillips, A., Ed., and M. Davis, Ed., “Tags for Identifying Languages”, BCP 47, RFC 5646, September 2009. <http://tools.ietf.org/html/rfc5646>.

[ECMAScript] *ECMAScript Language Specification Edition 5,1*. June 2011. Standard ECMA-262. <http://www.ecma-international.org/publications/standards/Ecma-262.htm>.

## Typographical Conventions

Keywords defined by this specification use this monospaced font.

Normative source code uses this paragraph style.

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

Example : 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.

All other text is normative unless otherwise labeled.

# JSON Format Design

JSON, as described in [**[RFC7159]**](#rfc7159), defines a text format for serializing structured data. Objects are serialized as an unordered collection of name/value pairs.

JSON does not define any semantics around the name/value pairs that make up an object, nor does it define an extensibility mechanism for adding control information to a payload.

OData’s JSON format extends JSON by defining general conventions for name/value pairs that annotate a JSON object, property or array. OData defines a set of canonical annotations for control information such as ids, types, and links, and custom annotations MAY be used to add domain-specific information to the payload.

A key feature of OData’s JSON format is to allow omitting predictable parts of the wire format from the actual payload. To reconstitute this data on the receiving end, expressions are used to compute missing links, type information, and other control data. These expressions (together with the data on the wire) can be used by the client to compute predictable payload pieces as if they had been included on the wire directly.

Annotations are used in JSON to capture control information that cannot be predicted (e.g., the next link of a collection) as well as a mechanism to provide values where a computed value would be wrong (e.g., if the media read link of one particular entity does not follow the standard URL conventions). Computing values from metadata expressions is compute intensive and some clients might opt for a larger payload size to avoid computational complexity; to accommodate for this the Accept header allows the client to control the amount of control information added to the response.

To optimize streaming scenarios, there are a few restrictions that MAY be imposed on the sequence in which name/value pairs appear within JSON objects. For details on the ordering requirements see [Payload Ordering Constraints](#_Payload_Ordering_Constraints_1).

# Requesting the JSON Format

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

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

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

Possible format parameters are:

* [ExponentialDecimals](#_Controlling_the_Representation)
* [IEEE754Compatible](#_Controlling_the_Representation)
* [metadata](#_Controlling_the_Amount) (odata.metadata)
* [streaming](#_Payload_Ordering_Constraints_1) (odata.streaming)

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

Services SHOULD advertise the supported MIME types by annotating the entity container with the term Capabilities.SupportedFormats defined in [**[OData-VocCap]**](#VocCapabilities), listing all available formats and combinations of supported format parameters.

## Controlling the Amount of Control Information in Responses

The amount of [control information](#_Control_Information) needed (or desired) in the payload depends on the client application and device. The metadata parameter can be applied to the Accept header of an OData request to influence how much control information will be included in the response.

Other Accept header parameters (e.g., streaming) are orthogonal to the metadata parameter and are therefore not mentioned in this section.

If a client prefers a very small wire size and is intelligent enough to compute data using metadata expressions, the Accept header should include [metadata=minimal](#_odata=minimalmetadata). If computation is more critical than wire size or the client is incapable of computing control information, [metadata=full](#_odata.metadata=full) directs the service to inline the control information that normally would be computed from metadata expressions in the payload. [metadata=none](#_odata=nometadata) is an option for clients that have out-of-band knowledge or don't require control information.

In addition the client may use the include-annotations preference in the Prefer header to request additional control information. Services supporting this MUST NOT omit control information required by the chosen metadata parameter, and services MUST NOT exclude the [nextLink](#odataNext), [deltaLink](#odataDelta), and [count](#odataCount) if they are required by the response type.

If the client includes the OData-MaxVersion header in a request and does not specify the metadata format parameter in either the Accept header or $format query option, the service MUST return at least the [minimal control information](#_odata=minimalmetadata).

Note that in OData 4.0 the metadata format parameter was prefixed with “odata.”. Payloads with an OData-Version header equal to 4.0 MUST include the “odata.” prefix. Payloads with an OData-Version header equal to 4.01 or greater SHOULD NOT include the “odata.” prefix.

### metadata=minimal (odata.metadata=minimal)

The metadata=minimal format parameter indicates that the service SHOULD remove computable control information from the payload wherever possible. The response payload MUST contain at least the following common [annotations](#_Instance_Annotations):

* [context](#_The_odata.metadata_Annotation_1): the root context URL of the payload and the context URL for any deleted entries or added or deleted links in a delta response, or for entities or entity collections whose set cannot be determined from the root context URL
* [etag](#odataEtag): the ETag of the entity, as appropriate
* [count](#odataCount): the total count of a collection of entities or collection of entity references, if requested
* [nextLink](#odataNext): the next link of a collection with partial results
* [deltaLink](#odataDelta): the delta link for obtaining changes to the result, if requested

In addition, odata annotations MUST appear in the payload for cases where actual values are not the same as the computed values and MAY appear otherwise. When odata annotations appear in the payload, they are treated as exceptions to the computed values.

Media entities and stream properties MAY in addition contain the following annotations:

* [mediaEtag](#_Annotation_odata.media*_1): the ETag of the stream, as appropriate
* [mediaContentType](#_Annotation_odata.media*_1): the content type of the stream

### metadata=full (odata.metadata=full)

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

The full list of annotations that may appear in an metadata=full response is as follows:

* [context](#_The_odata.metadata_Annotation_1): the context URL for a collection, entity, primitive value, or service document.
* [count](#odataCount): the total count of a collection of entities or collection of entity references, if requested.
* [nextLink](#odataNext): the next link of a collection with partial results
* [deltaLink](#odataDelta): the delta link for obtaining changes to the result, if requested
* [id](#_The_odata.id_Annotation_1): the ID of the entity
* [etag](#odataEtag): the ETag of the entity
* [readLink](#_Annotation_odata.editLink_and): the link used to read the entity, if the edit link cannot be used to read the entity
* [editLink](#_Annotation_odata.editLink_and): the link used to edit/update the entity, if the entity is updatable and the id does not represent a URL that can be used to edit the entity
* [navigationLink](#_Annotation_odata.navigationLink_and_1): the link used to retrieve the values of a navigation property
* [associationLink](#_Annotation_odata.navigationLink_and_1): the link used to describe the relationship between this entity and related entities
* [type](#odataType): the type of the containing object or targeted property if the type of the object or targeted property cannot be heuristically determined

Media entities and stream properties may in addition contain the following annotations:

* [mediaReadLink](#_Annotation_odata.media*_1): the link used to read the stream
* [mediaEditLink](#_Annotation_odata.media*_1): the link used to edit/update the stream
* [mediaEtag](#_Annotation_odata.media*_1): the ETag of the stream, as appropriate
* [mediaContentType](#_Annotation_odata.media*_1): the content type of the stream

### metadata=none (odata.metadata=none)

The metadata=none format parameter indicates that the service SHOULD omit control information other than [nextLink](#odataNext) and [count](#odataCount). These annotations MUST continue to be included, as applicable, even in the metadata=none case.

It is not valid to specify metadata=none on a [delta request](#_Delta_Response).

## Controlling the Representation of Numbers

The IEEE754Compatible=true format parameter indicates that the service MUST serialize Edm.Int64 and Edm.Decimal numbers (including the [count](#_Annotation_odata.count), if requested) as strings. This is in conformance with [**[I-JSON]**](#ijson).

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]**](#ECMAScript) (see [section 4.3.1.9](http://www.ecma-international.org/ecma-262/5.1/#sec-4.3.19)) resulting in integers losing precision past 15 digits, and decimals losing precision due to the conversion from base 10 to base 2.

OData JSON payloads that format Edm.Int64 and Edm.Decimal values as strings MUST specify this format parameter in the media type returned in the [Content-Type](#_Header_Content-Type) header.

For payloads with an OData-Version header equal to 4.0 the ExponentialDecimals=true format parameter indicates that the service MAY serialize Edm.Decimal numbers (including the [count](#_Annotation_odata.count), if requested) in exponential notation (e.g. 1e-6 instead of 0.000001).

The sender of a request MUST specify ExponentialDecimals=true in the Content-Type header if the request body contains Edm.Decimal values in exponential notation.

If not specified, or specified as ExponentialDecimals=false, all Edm.Decimal values MUST be serialized in long notation, using only an optional sign, digits, and an optional decimal point followed by digits.

Payloads with an OData-Version header equal to 4.01 or greater always allow exponential notation for numbers and the ExponentialNotation format parameter is not needed or used.

# Common Characteristics

This section describes common characteristics of the representation for OData values in JSON. A request or response body consists of several parts. It contains OData values as part of a larger document. Requests and responses are structured almost identical; the few existing differences will be explicitly called out in the respective subsections.

## Header Content-Type

Requests and responses with a JSON message body MUST have a Content-Type header value of application/json.

Requests MAY add the charset parameter to the content type. Allowed values are UTF-8, UTF-16, and UTF-32. If no charset parameter is present, UTF-8 MUST be assumed.

Responses MUST include the [metadata](#_Controlling_the_Amount) parameter to specify the amount of metadata included in the response.

Responses MUST include the [IEEE754Compatible](#_Controlling_the_Representation) parameter if Edm.Int64 and Edm.Decimal numbers are represented as strings.

Requests and responses MAY add the streaming parameter with a value of true or false, see section [Payload Ordering Constraints](#_Payload_Ordering_Constraints_1).

## Message Body

Each message body is represented as a single JSON object. This object is either the representation of an [entity](#Entitiy), an [entity reference](#ResourceReference) or a [complex type instance](#_Representing_a_Complex), or it contains a name/value pair whose name MUST be value and whose value is the correct representation for a [primitive value](#_Primitive_Value), a [collection of primitive values](#CollectionOfPrimitive), a [collection of complex values](#_Collection_of_Complex), a [collection of entities](#CollectionOfEntities), or a collection of objects that represent [changes to a previous result](#DeltaResponse).

Client libraries MUST retain the order of objects within an array in JSON responses.

## Relative URLs

URLs present in a payload (whether request or response) MAY be represented as relative URLs.

Relative URLs, other than those in [type](#_Annotation_odata.type), are relative to their base URL, which is

* the [context URL](#_Annotation_odata.metadata) of the same JSON object, if one exists, otherwise
* the context URL of the enclosing object, if one exists, otherwise
* the context URL of the next enclosing object, if one exists, etc. until the document root, otherwise
* the request URL.

For context URLs these rules apply starting with the second bullet point.

Within the [type](#_Annotation_odata.type) annotation, relative URLs are relative to the base type URL, which is

* the type of the enclosing object, if one exists, otherwise
* the type of the next enclosing object, if one exists, etc. until the document root, otherwise
* the context URL of the document root, if one exists, otherwise
* the request URL.

Processors expanding the URLs MUST use normal URL expansion rules as defined in RFC3986. This means that if the base URL is a context URL, the part starting with $metadata# is ignored when resolving the relative URL.

Clients that receive relative URLs in response payloads SHOULD use the same relative URLs, where appropriate, in request payloads (such as [bind operations](#_Bind_Operation) and batch requests) and in system query options (such as $id).

Example :

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 ...

 "@editLink": "Customers('ALFKI')",

 ...

 "Orders@navigationLink": "Customers('ALFKI')/Orders",

 ...

}

The resulting absolute URLs are http://host/service/Customers('ALFKI') and http://host/service/Customers('ALFKI')/Orders.

## Payload Ordering Constraints

Ordering constraints MAY be imposed on the JSON payload in order to support streaming scenarios. These ordering constraints MUST only be assumed if explicitly specified as some clients (and services) might not be able to control, or might not care about, the order of the JSON properties in the payload.

Clients can request that a JSON response conform to these ordering constraints by specifying a media type of application/json with the streaming=true parameter in the Accept header or $format query option. Services MUST return 406 Not Acceptable if the client only requests streaming and the service does not support it.

Clients may specify the streaming=true parameter in the Content-Type header of requests to indicate that the request body follows the payload ordering constraints. In the absence of this parameter, the service must assume that the JSON properties in the request are unordered.

Processors MUST only assume streaming support if it is explicitly indicated in the Content-Type header via the streaming=true parameter.

Example : a payload with

Content-Type: application/json;metadata=minimal;streaming=true

can be assumed to support streaming, whereas a payload with

Content-Type: application/json;metadata=minimal

cannot be assumed to support streaming.

JSON producers are encouraged to follow the payload ordering constraints whenever possible (and include the streaming=true content type parameter) to support the maximum set of client scenarios.

To support streaming scenarios the following payload ordering constraints have to be met:

* If present, the context annotation MUST be the first property in the JSON object.
* The type annotation, if present, MUST appear next in the JSON object.
* The id and etag annotations MUST appear before any property or property annotation.
* All annotations for a structural or navigation property MUST appear as a group immediately before the property they annotate. The one exception is the nextLink annotation of an collection which MAY appear after the collection it annotates.
* All other odata annotations can appear anywhere in the payload as long as they do not violate any of the above rules.
* For 4.0 payloads, annotations for navigation properties MUST appear after all structural properties. 4.01 clients MUST NOT assume this ordering.

Note that, in OData 4.0, the streaming format parameter was prefixed with “odata.”. Payloads with an OData-Version header equal to 4.0 MUST include the “odata.” prefix. Payloads with an OData-Version header equal to 4.01 or greater SHOULD NOT include the “odata.” prefix.

## Control Information

In addition to the “pure data” a message body MAY contain control information that is represented as [annotations](#_Instance_Annotations) whose names start with odata followed by a dot.

In requests and responses that do not contain the OData-Version header with a value of 4.0, the “odata.” prefix SHOULD be omitted.

In some cases control information is required in request payloads; this is called out in the following subsections.

Receivers that encounter unknown annotations in any namespace, including the odata namespace, MUST NOT stop processing and MUST NOT signal an error.

### Annotation context (odata.context)

The context annotation returns the context URL (see [**[OData-Protocol]**](#odata)) for the payload. This URL can be absolute or [relative](#_Relative_URLs).

The context annotation is not returned if [metadata=none](#_odata.metadata=none) is requested. Otherwise it MUST be the first property of any JSON response.

The context annotation MUST also be included in requests and responses for entities whose entity set cannot be determined from the context URL of the collection.

If the metadata document referenced by the context annotation is versioned, then the response MUST also include the Core.SchemaVersion annotation, defined in [**[OData-VocCore]**](#VocCore), to indicate the version of the schema used to generate the response. For [streamed JSON responses](#_Payload_Ordering_Constraints_1), this annotation MUST immediately follow the context annotation. If the Core.SchemaVersion annotation is present, the SchemaVersion header, defined in [**[OData-Protocol]**](#odata), SHOULD be used when retrieving the referenced metadata document.

For more information on the format of the context URL, see [**[OData-Protocol]**](#odata).

Request payloads MAY include a context URL as a base URL for [relative URLs](#_Relative_URLs) in the request payload.

Example :

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 "@metadataEtag": "W/\"A1FF3E230954908F\"",

 "@Core.SchemaVersion": "2.0.1",

 ...

}

### Annotation metadataEtag (odata.metadataEtag)

The metadataEtag annotation MAY appear in a response in order to specify the entity tag (ETag) that can be used to determine the version of the metadata of the response. If an ETag is returned when requesting the metadata document, then the service SHOULD set the metadataEtag annotation to the metadata document's ETag in all responses when using [metadata=minimal](#_odata=minimalmetadata) or [metadata=full](#_odata.metadata=full). If no ETag is returned when requesting the metadata document, then the service SHOULD NOT set the metadataEtag annotation in any responses.

For details on how ETags are used, see [**[OData-Protocol]**](#odata).

### Annotation type (odata.type)

The type annotation specifies the type of a JSON object or name/value pair. Its value is a URI that identifies the type of the property or object. For built-in primitive types the value is the unqualified name of the primitive type. For payloads described by an OData-Version header with a value of 4.0, this name MUST be prefixed with the hash symbol (#); for non-OData 4.0 payloads, built-in primitive type values SHOULD be represented without the hash symbol, but consumers of 4.01 or greater payloads MUST support values with or without the hash symbol.. For all other types, the URI may be absolute or relative to the type of the containing object. The root type may be absolute or relative to the root [context URL](#_Annotation_odata.context).

If the URI references a metadata document (that is, it’s not just a fragment), and refers to a specific version of that metadata, then the object or name/value pair MUST also be annotated with the Core.SchemaVersion annotation, defined in [**[OData-VocCore]**](#VocCore), to indicate the version of the metadata document containing the corresponding version of the type. For [streamed JSON responses](#_Payload_Ordering_Constraints_1), this annotation MUST immediately follow the type annotation. If the Core.SchemaVersion annotation is present, the Core.SchemaVersion header, defined in [**[OData-Protocol]**](#odata), SHOULD be used when retrieving the referenced metadata document.

For non-built in primitive types, the URI contains the namespace-qualified or alias-qualified type, specified as a URI fragment. For properties that represent a collection of values, the fragment is the namespace-qualified or alias-qualified element type enclosed in parentheses and prefixed with Collection. The namespace or alias MUST be defined or the namespace referenced in the metadata document of the service, see [**[OData-CSDLXML]**](#ODataCSDLXMLRef).

The type annotation MUST appear in requests and in responses with [minimal](#_odata=minimalmetadata) or [full](#_odata.metadata=full) metadata, if the type cannot be heuristically determined, as described below, and one of the following is true:

* The type is derived from the type specified for the (collection of) entities or (collection of) complex type instances, or
* The type is for a property whose type is not declared in $metadata.

The following heuristics are used to determine the primitive type of a dynamic property in the absence of the type annotation:

* Boolean values have a first-class representation in JSON and do not need any additional annotations.
* Numeric values have a first-class representation in JSON but are not further distinguished, so they include a [type](#_Annotation_odata.type) annotation unless their type is Double.
* The special floating-point values NaN, INF, and -INF are serialized as strings. In 4.0 responses these strings are serialized as the value of the structural property and MUST have a [type](#_Annotation_odata.type) annotation to specify the numeric type of the property.
* String values do have a first class representation in JSON, but there is an obvious collision: OData also encodes a number of other primitive types as strings, e.g. DateTimeOffset, Int64 in the presence of the [IEEE754Compatible](#_Controlling_the_Representation) format parameter etc. If a property appears in JSON string format, it should be treated as a string value unless the property is known (from the metadata document) to have a different type.

For more information on namespace- and alias-qualified names, see [**[OData-CSDLXML]**](#ODataCSDLXMLRef)..

Example : entity of type Model.VipCustomer defined in the metadata document of the same service with a dynamic property of type Edm.Date

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 "@type": "#Model.VipCustomer",

 "ID": 2,

 "DynamicValue@type": "Date",

 "DynamicValue": "2016-09-22",

 ...

}

Example : entity of type Model.VipCustomer defined in the metadata document of a different service

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 "@type": "http://host/alternate/$metadata#Model.VipCustomer",

 "ID": 2,

 ...

}

### Annotation count (odata.count)

The count annotation occurs only in responses and can annotate any collection, see [**[OData-Protocol]**](#odata)section 11.2.5.5 System Query Option $count. Its value is an Edm.Int64 value corresponding to the total count of members in the collection represented by the request.

### Annotation nextLink (odata.nextLink)

The nextLink annotation indicates that a response is only a subset of the requested collection. It contains a URL that allows retrieving the next subset of the requested collection.

This annotation can also be applied to [expanded to-many navigation properties](#_Expanded_Navigation_Property).

### Annotation delta (odata.delta)

The delta annotation is applied to a collection-valued navigation property within an [added/changed entity](#_Representing_Added/Changed_Entities_1) in a delta payload to represent changes in membership or value of nested entities.

### Annotation deltaLink (odata.deltaLink)

The deltaLink annotation contains a URL that can be used to retrieve changes to the current set of results. The deltaLink annotation MUST only appear on the last page of results. A page of results MUST NOT have both a deltaLink annotation and a nextLink annotation.

### Annotation id (odata.id)

The id annotation contains the entity-id, see [**[OData-Protocol]**](#odata). By convention the entity-id is identical to the canonical URL of the entity, as defined in [**[OData-URL]**](#ODataURLRef).

The id annotation MUST appear in responses if [metadata=full](#_odata.metadata=full) is requested, or if [metadata=minimal](#_odata=minimalmetadata) is requested and any of the entity's key fields are omitted from the response *or* the entity-id is not identical to the canonical URL of the entity after

* IRI-to-URI conversion as defined in [**[RFC3987]**](#RFC3987),
* relative resolution as defined in section 5.2 of [**[RFC3986]**](#RFC3986), and
* percent-encoding normalization as defined in section 6 of [**[RFC3986]**](#RFC3986).

Note that the entity-id MUST be invariant across languages, so if key values are language dependent then the id MUST be included if it does not match convention for the localized key values. If the id is represented, it MAY be a [relative URL](#_Relative_URLs).

If the entity is transient (i.e. cannot be read or updated), the id annotation MUST appear and have the null value.

The id annotation MUST NOT appear for a collection. Its meaning in this context is reserved for future versions of this specification.

Entities with id equal to null cannot be compared to other entities, reread, or updated. If [metadata=minimal](#_odata=minimalmetadata) is specified and the id is not present in the entity then the canonical URL MUST be used as the entity-id.

### Annotation editLink and readLink (odata.editLink and odata.readLink)

The editLink annotation contains the edit URL of the entity; see [**[OData-Protocol]**](#odata).

The readLink annotation contains the read URL of the entity or collection; see [**[OData-Protocol]**](#odata).

The editLink and readLink annotations are ignored in request payloads and not written in responses if [metadata=none](#_odata=nometadata) is requested.

The default value of both the edit URL and read URL is the entity's [entity-id](#_The_odata.id_Annotation_1) appended with a cast segment to the type of the entity if its type is derived from the declared type of the entity set. If neither the editLink nor the readLink annotation is present in an entity, the client uses this default value for the edit URL.

For updatable entities:

* The editLink annotation is written if [metadata=full](#_odata.metadata=full) is requested or if [metadata=minimal](#_odata=minimalmetadata) is requested and the edit URL differs from the default value of the edit URL.
* The readLink annotation is written if the read URL is different from the edit URL. If no readLink annotation is present, the read URL is identical to the edit URL.

For read-only entities:

* The readLink annotation is written if [metadata=full](#_odata.metadata=full) is requested or if [metadata=minimal](#_odata=minimalmetadata) is requested and its value differs from the default value of the read URL.
* The readLink annotation may also be written if [metadata=minimal](#_odata=minimalmetadata) is specified in order to signal that an individual entity is read-only.

For collections:

* The readLink annotation, if written, MUST be the request URL that produced the collection.
* The editLink annotation MUST NOT be written as its meaning in this context is reserved for future versions of this specification.

### Annotation etag (odata.etag)

The etag annotation MAY be applied to an [entity](#_Entity) in a response. The value of the annotation is an entity tag (ETag) which is an opaque string value that can be used in a subsequent request to determine if the value of the entity has changed.

For details on how ETags are used, see [**[OData-Protocol]**](#odata).

The etag annotation is ignored in request payloads and not written in responses if [metadata=none](#_odata=nometadata) is requested.

### Annotation navigationLink and associationLink (odata.navigationLink and odata.associationLink)

The navigationLink annotation in a response contains a *navigation URL* that can be used to retrieve an entity or collection of entities related to the current entity via a [navigation property](#_Navigation_Property).

The *default computed value of a navigation URL* is the value of the [read URL](#_Annotation_odata.editLink_and) appended with a segment containing the name of the navigation property. The service MAY omit the navigationLink annotation if [metadata=minimal](#_odata=minimalmetadata) has been specified on the request and the navigation link matches this computed value.

The associationLink annotation in a response contains an *association URL* that can be used to retrieve a reference to an entity or a collection of references to entities related to the current entity via a navigation property.

The *default computed value of an association URL* is the value of the navigation URL appended with /$ref. The service MAY omit the associationLink annotation if the association link matches this computed value.

The navigationLink and associationLink annotations are ignored in request payloads and not written in responses if [metadata=none](#_odata=nometadata) is requested.

### Annotation media\* (odata.media\*)

For [media entities](#_Media_Entity) and [stream properties](#_Stream_Property) at least one of the annotations mediaEditLink and mediaReadLink MUST be included in responses if they don't follow standard URL conventions as defined in [**OData-URL**](#ODataURLRef) or if [metadata=full](#_odata.metadata=full) is requested.

The mediaEditLink annotation contains a URL that can be used to update the binary stream associated with the media entity or stream property. It MUST be included for updatable media entities if it differs from the value of the id, and for updatable stream properties if it differs from standard URL conventions.

The mediaReadLink annotation contains a URL that can be used to read the binary stream associated with the media entity or stream property. It MUST be included if its value differs from the value of the associated mediaEditLink, if present, or the value of the id for media entities if the associated mediaEditLink is not present.

The mediaContentType annotation MAY be included; its value SHOULD match the content type of the binary stream represented by the mediaReadLink URL. This is only a hint; the actual content type will be included in a header when the resource is requested.

The mediaEtag annotation MAY be included; its value is the ETag of the binary stream represented by this media entity or stream property.

The media\* annotations are ignored in request payloads and not written in responses if [metadata=none](#_odata=nometadata) is requested.

Example :

{

 "@context": "http://host/service/$metadata#Employees/$entity",

 "@mediaReadLink": "Employees(1)/$value",

 "@mediaContentType": "image/jpeg",

 "ID": 1,

 ...

}

# Service Document

A service document in JSON is represented as a single JSON object with at least the [context](#_Annotation_context_(odata.context)) annotation and a property value.

The value of the [context](#_The_odata.metadata_Annotation_1) annotation MUST be the URL of the metadata document, without any fragment part.

The value of the value property MUST be a JSON array containing one element for each entity set and function import with an explicit or default value of true for the attribute IncludeInServiceDocument and each singleton exposed by the service, see[[**[OData-CSDLXML]**](#ODataCSDLXMLRef).](#odataCSDL).

Each element MUST be a JSON object with at least two name/value pairs, one with name name containing the name of the entity set, function import, or singleton, and one with name url containing the URL of the entity set, which may be an absolute or a [relative URL](#_Relative_URLs). It MAY contain a name/value pair with name title containing a human-readable, language-dependent title for the object.

JSON objects representing an entity set MAY contain an additional name/value pair with name kind and a value of EntitySet. If the kind name/value pair is not present, the object MUST represent an entity set.

JSON objects representing a function import MUST contain the kind name/value pair with a value of FunctionImport.

JSON objects representing a singleton MUST contain the kind name/value pair with a value of Singleton.

JSON objects representing a related service document MUST contain the kind name/value pair with a value of ServiceDocument.

Clients that encounter unknown values of the kind name/value pair not defined in this version of the specification MUST NOT stop processing and MUST NOT signal an error.

Service documents MAY contain [annotations](#_Instance_Annotations) in any of its JSON objects. Services MUST NOT produce name/value pairs other than the ones explicitly defined in this section, and clients MUST ignore unknown name/value pairs.

Example :

{

 "@context": "http://host/service/$metadata",

 "value": [

 {

 "name": "Orders",

 "kind": "EntitySet",

 "url": "Orders"

 },

 {

 "name": "OrderItems",

 "title": "Order Details",

 "url": "OrderItems"

 },

 {

 "name": "TopProducts",

 "title": "Best-Selling Products",

 "kind": "FunctionImport",

 "url": "TopProducts"

 },

 {

 "name": "MainSupplier",

 "title": "Main Supplier",

 "kind": "Singleton",

 "url": "MainSupplier"

 },

 {

 "name": "Human Resources",

 "kind": "ServiceDocument",

 "url": "http://host/HR/"

 }

 ]

}

# Entity

An entity is serialized as a JSON object.

Each [property](#_Structural_Property) to be transmitted is represented as a name/value pair within the object. The order properties appear within the object is considered insignificant.

An entity in a payload may be a complete entity, a projected entity (see *System Query Option* $select [**[OData-Protocol]**](#odata)), or a partial entity update (see*Update an Entity* in [**[OData-Protocol]**](#odata)).

An entity representation can be (modified and) round-tripped to the service directly. The [context URL](#_The_odata.metadata_Annotation_1) is used in requests only as a base for [relative URLs](#_Relative_URLs).

Example : entity with metadata=minimal

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 "ID": "ALFKI",

 "CompanyName": "Alfreds Futterkiste",

 "ContactName": "Maria Anders",

 "ContactTitle": "Sales Representative",

 "Phone": "030-0074321",

 "Fax": "030-0076545",

 "Address": {

 "Street": "Obere Str. 57",

 "City": "Berlin",

 "Region": null,

 "PostalCode": "D-12209"

 }

}

Example : entity with metadata=full

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 "@id": "Customers('ALFKI')",

 "@etag": "W/\"MjAxMy0wNS0yN1QxMTo1OFo=\"",

 "@editLink": "Customers('ALFKI')",

 "ID": "ALFKI",

 "CompanyName": "Alfreds Futterkiste",

 "ContactName": "Maria Anders",

 "ContactTitle": "Sales Representative",

 "Phone": "030-0074321",

 "Fax": "030-0076545",

 "Address": {

 "Street": "Obere Str. 57",

 "City": "Berlin",

 "Region": null,

 "PostalCode": "D-12209",

 "Country@associationLink":"Customers('ALFKI')/Address/Country/$ref",

 "Country@navigationLink": "Customers('ALFKI')/Address/Country"

 },

 "Orders@associationLink": "Customers('ALFKI')/Orders/$ref",

 "Orders@navigationLink": "Customers('ALFKI')/Orders"

}

# Structural Property

A property within an entity or complex type instance is represented as a name/value pair. The name MUST be the name of the property; the value is represented depending on its type as a [primitive value](#_Primitive_Value), a [complex value](#_Complex_Value), a [collection of primitive values](#CollectionOfPrimitive), or a [collection of complex values](#_Collection_of_Complex).

## Primitive Value

Primitive values are represented following the rules of [**[RFC7159]**](#rfc7159).

Null values are represented as the JSON literal null.

Values of type Edm.Boolean are represented as the JSON literals true and false

Values of types Edm.Byte, Edm.SByte, Edm.Int16, Edm.Int32, Edm.Int64, Edm.Single, Edm.Double, and Edm.Decimal are represented as JSON numbers, except for NaN, INF, and –INF which are represented as strings in OData 4.0. In OData 4.01, they are represented as properties annotated with the Core.NumericValueException annotation term as defined in [**[OData-VocCore]**](#VocCore).

Values of type Edm.String are represented as JSON strings, using the JSON string escaping rules.

Values of type Edm.Binary, Edm.Date, Edm.DateTimeOffset, Edm.Duration, Edm.Guid, and Edm.TimeOfDay are represented as JSON strings whose content satisfies the rules binaryValue, dateValue, dateTimeOffsetValue, durationValue, guidValue, and timeOfDayValue respectively, in [**[OData-ABNF]**](#abnf).

Enumeration values are represented as JSON strings whose content satisfies the rule enumValue in [**[OData-ABNF]**](#abnf). The preferred representation is the enumerationMember. If no enumerationMember (or combination of named enumeration members) is available, the enumMemberValue representation may be used.

Geography and geometry values are represented as geometry types as defined in [**[GeoJSON]**](#GeoJSON), with the following modifications:

* Keys SHOULD be ordered with type first, then coordinates, then any other keys
* The coordinates member of a LineString can have zero or more positions
* If the optional CRS object is present, it MUST be of type name, where the value of the name member of the contained properties object is an EPSG SRID legacy identifier.

Geography and geometry types have the same representation in a JSON payload. Whether the value represents a geography type or geometry type is inferred from its usage or specified using the [type](#_Annotation_odata.type) annotation.

Example :

{

 "NullValue": null,

 "TrueValue": true,

 "FalseValue": false,

 "BinaryValue": "T0RhdGE",

 "IntegerValue": -128,

 "DoubleValue": 3.1415926535897931,

 "SingleValue@Core.NumericValueException": "INF",

 "DecimalValue": 34.95,

 "StringValue": "Say \"Hello\",\nthen go",

 "DateValue": "2012-12-03",

 "DateTimeOffsetValue": "2012-12-03T07:16:23Z",

 "DurationValue": "P12DT23H59M59.999999999999S",

 "TimeOfDayValue": "07:59:59.999",

 "GuidValue": "01234567-89ab-cdef-0123-456789abcdef",

 "Int64Value": 0,

 "ColorEnumValue": "Yellow",

 "GeographyPoint": {"type": "Point","coordinates":[142.1,64.1]}

}

## Complex Value

A complex value is represented as a single JSON object containing one name/value pair for each property that makes up the complex type. Each property value is formatted as appropriate for the type of the property.

It MAY have name/value pairs for instance annotations, including odata annotations.

Example :

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 ...

 "Address": {

 "Street": "Obere Str. 57",

 "City": "Berlin",

 "Region": null,

 "PostalCode": "D-12209"

 }

}

A complex value with no selected properties, or no defined properties (such as an empty open complex type or complex type with no structural properties) is represented as an empty JSON object.

## Collection of Primitive Values

A collection of primitive values is represented as a JSON array; each element in the array is the representation of a [primitive value](#_Primitive_Value). A JSON literal null represents a null value within the collection. An empty collection is represented as an empty array.

Example : partial collection of strings with next link

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 ...

 "EmailAddresses": [

 "Julie@Swansworth.com",

 "Julie.Swansworth@work.com"

 ],

 "EmailAddresses@nextLink": "..."

}

## Collection of Complex Values

A collection of complex values is represented as a JSON array; each element in the array is the representation of a [complex value](#_Representing_a_Complex). A JSON literal null represents a null value within the collection. An empty collection is represented as an empty array.

Example : partial collection of complex values with next link

{

 "PhoneNumbers": [

 {

 "Number": "425-555-1212",

 "Type": "Home"

 },

 {

 "@type": "#Model.CellPhoneNumber",

 "Number": "425-555-0178",

 "Type": "Cell",

 "Carrier": "Sprint"

 }

 ],

 "PhoneNumbers@nextLink": "..."

}

## Untyped Value

OData 4.01 adds the built-in abstract types Edm.Untyped and Collection(Edm.Untyped)that services can use to advertise in metadata that there is a property of a particular name present, but there is no type to describe the structure of the property’s values.

The value of an Edm.Untyped property MAY be a primitive value, a structural value, or a collection. If a collection, it may contain any combination of primitive values, structural values, and collections.

The value of a property of type Collection(Edm.Untyped)MUST be a collection, and it MAY contain any combination of primitive values, structural values, and collections.

Untyped values are the only place where a collection can directly contain a collection, or a collection can contain a mix of primitive values, structural values, and collections.

All children of an untyped property are assumed to be untyped unless they are annotated with the [type](#odataType) annotation, in which case they MUST conform to the type described by the annotation.

# Navigation Property

A navigation property is a reference from a source entity to zero or more related entities.

## Navigation Link

The navigation link for a navigation property is represented as a [navigationLink](#_Annotation_odata.navigationLink_and_1) annotation on the navigation property. Its value is an absolute or [relative URL](#_Relative_URLs) that allows retrieving the related entity or collection of entities.

The navigation link for a navigation property is only represented if the client requests metadata=full or the navigation link cannot be computed, e.g. if it is within a collection of complex type instances. If it is represented it MUST immediately precede the expanded navigation property if the latter is represented.

Example :

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 ...

 "Orders@navigationLink": "Customers('ALFKI')/Orders",

 ...

}

## Association Link

The association link for a navigation property is represented as an [associationLink](#_Annotation_odata.navigationLink_and_1) annotation on the navigation property. Its value is an absolute or [relative URL](#_Relative_URLs) that can be used to retrieve the reference or collection of references to the related entity or entities.

The association link for a navigation property is only represented if the client requests metadata=full or the association link cannot be computed by appending /$ref to the navigation link. If it is represented, it MUST immediately precede the navigation link if the latter is represented, otherwise it MUST immediately precede the expanded navigation property if it is represented.

Example :

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 ...

 "Orders@associationLink": "Customers('ALFKI')/Orders/$ref",

 ...

}

## Expanded Navigation Property

An expanded navigation property is represented as a name/value pair where the name is the name of the navigation property, and the value is the representation of the related entity or collection of entities.

If at most one entity can be related, the value is the representation of the related entity, or null if no entity is currently related.

If a collection of entities can be related, it is represented as a JSON array. Each element is the [representation of an entity](#_Entity) or the [representation of an entity reference](#ResourceReference). An empty collection of entities (one that contains no entities) is represented as an empty JSON array. The navigation property MAY be annotated with [context](#_The_odata.metadata_Annotation_1), [count or nextLink](#_Annotation_odata.count). If a navigation property is expanded with the suffix /$count, only the [count annotation is represented.](#_Annotation_odata.count)

Example :

{

 "@context": "http://host/service/$metadata#Customers/$entity",

 ...

 "Orders@count": 42,

 "Orders": [ ... ],

 "Orders@nextLink": "...",

 ...

}

## Deep Insert

When inserting a new entity with a POST request, related new entities MAY be specified using the same representation as for an [expanded navigation property](#_Expanded_Navigation_Property).

Deep inserts are not allowed in update operations using PUT or PATCH requests.

Example : inserting a new order for a new customer with order items related to existing products:

{

 "ID": 11643,

 "Amount": 100,

 ...,

 "Customer": {

 "ID": "ANEWONE",

 ...

 },

 "Items": [

 {

 "Product": { "@id": "Products(28)" },

 "Quantity": 1,

 ...

 },

 {

 "Product": { "@id": "Products(39)" },

 "Quantity": 5,

 ...

 }

 ]

}

## Bind Operation

When inserting or updating an entity, relationships of navigation properties MAY be inserted or updated via bind operations.

For requests containing an OData-Version header with a value of 4.0, a bind operation is encoded as a property annotation odata.bind on the navigation property it belongs to and has a single value for single-valued navigation properties or an array of values for collection navigation properties. For nullable single-valued navigation properties the value null may be used to remove the relationship.

Example : assign an existing product to an existing category with a partial update request against the product

PATCH http://host/service/Products(42) HTTP/1.1

{

 "Category@odata.bind": "Categories(6)"

}

The values are the [ids](#_Annotation_odata.id) of the related entities. They MAY be absolute or [relative URLs](#_Relative_URLs).

For requests containing an OData-Version header with a value of 4.01, a relationship is bound to an existing entity using the same representation as for an [expanded entity reference](#_Entity_Reference).

Example 20: assign an existing product to an existing category with a partial update request against the product

PATCH http://host/service/Products(42) HTTP/1.1

{

 "Category": {"@id": "Categories(6)"}

}

Example 21: submit a partial update request to:

* modify the name of an existing category
* assign an existing product with the id 42 to the category
* assign an existing product 57 to the category and update its name
* create a new product named “Wedges” and assign it to the category

*at the end of the request, the updated category contains exactly the three specified products.*

PATCH http://host/service/Categories(6) HTTP/1.1

{

 "Name": "UpdatedCategory",

 "Products": [

 {

 "@id": "Products(42)"

 },

 {

 "@id": "Products(57)",

 "Name": "Widgets"

 },

 {

 "Name": "Wedges"

 }

 ]

}

OData 4.01 services MUST support both the OData 4.0 representation, for requests containing an OData-Version header with a value of 4.0, and the OData 4.01 representation, for requests containing an OData-Version header with a value of 4.01. Clients MUST NOT use @odata.bind in requests with an OData-Version header with a value of 4.01.

For insert operations collection navigation property bind operations and deep insert operations can be combined. For OData 4.0 requests, the bind operations MUST appear before the deep insert operations in the payload.

For update operations a bind operation on a collection navigation property adds additional relationships, it does not replace existing relationships, while bind operations on an entity navigation property update the relationship.

# Stream Property

An entity or complex type instance can have one or more stream properties.

The actual stream data is not usually contained in the representation. Instead stream property data is generally read and edited via URLs.

Depending on the [metadata level](#_Controlling_the_Amount), the stream property MAY be annotated to provide the read link, edit link, content type, and ETag of the media stream through a set of [media\*](#_Annotation_odata.media*) annotations. If the actual stream data is included inline, it is represented as a base64url-encoded string value, see **[RFC4648]**, section 5.

Example :

{

 "@context": "http://host/service/$metadata#Products/$entity",

 ...

 "Thumbnail@mediaReadLink": "http://server/Thumbnail546.jpg",

 "Thumbnail@mediaEditLink": "http://server/uploads/Thumbnail546.jpg",

 "Thumbnail@mediaContentType": "image/jpeg",

 "Thumbnail@mediaEtag": "W/\"####\"",

 "Thumbnail": "...base64url encoded value...",

 ...

}

# Media Entity

Media entities are entities that describe a media resource, for example a photo. They are represented as entities that contain additional [media\*](#_Annotation_odata.media*) annotations.

If the actual stream data for the media entity is included, it is represented as property named $value whose string value is the base64url-encoded value of the media stream, see **[RFC4648]**, section 5.

Example :

{

 "@context": "http://host/service/$metadata#Employees/$entity",

 "@mediaReadLink": "Employees(1)/$value",

 "@mediaContentType": "image/jpeg",

 "$value": "...base64url encoded value...",

 "ID": 1,

 ...

}

# Individual Property or Operation Response

An individual property or operation response is represented as a JSON object.

A single-valued property or operation response that has the null value does not have a representation; see [**[OData-Protocol]**](#odata).

A property or operation response that is of a primitive type is represented as an object with a single name/value pair, whose name is value and whose value is a [primitive value](#_Primitive_Value).

A property or operation response that is of complex type is represented as a [complex value](#_Representing_a_Complex).

A property or operation response that is of a collection type is represented as an object with a single name/value pair whose name is value. Its value is the JSON representation of a [collection of complex type values](#_Collection_of_Complex) or [collection of primitive values](#_Collection_of_Primitive).

Example : primitive value

{

 "@context": "http://host/service/$metadata#Edm.String",

 "value": "Pilar Ackerman"

}

Example : collection of primitive values

{

 "@context": "http://host/service/$metadata#Collection(Edm.String)",

 "value": ["small", "medium", "extra large"]

}

Example : empty collection of primitive values

{

 "@context": "http://host/service/$metadata#Collection(Edm.String)",

 "value": []

}

Example : complex value

{

 "@context": "http://host/service/$metadata#Model.Address",

 "Street": "12345 Grant Street",

 "City": "Taft",

 "Region": "Ohio",

 "PostalCode": "OH 98052",

 "Country@navigationLink": "Countries('US')"

}

Example : empty collection of complex values

{
 "@context":"http://host/service/$metadata#Collection(Model.Address)",
 "value": []

}

Note: the context URL is optional in requests.

# Collection of Entities

A collection of entities is represented as a JSON object containing a name/value pair named value. It MAY contain [context](#_The_odata.metadata_Annotation_1), [count](#_Annotation_odata.count), [nextLink](#odataNext), or [deltaLink](#odataDelta) annotations.

If present, the context annotation MUST be the first name/value pair in the response.

The count name/value pair represents the number of entities in the collection. If present and the streaming=true content type parameter is set, it MUST come before the value name/value pair. If the response represents a partial result, the count name/value pair MUST appear in the first partial response, and it MAY appear in subsequent partial responses (in which case it may vary from response to response).

The value of the value name/value pair is a JSON array where each element is [representation of an entity](#_Entity) or a representation of an entity reference. An empty collection is represented as an empty JSON array.

Functions or actions that are bound to this collection of entities are advertised in the “wrapper object” in the same way as [functions](#_Bound_Function) or [actions](#_Bound_Action) are advertised in the object representing a single entity.

The [nextLink](#_Annotation_odata.nextLink) annotation MUST be included in a response that represents a partial result.

Example :

{

 "@context": "...",

 "@count": 37,

 "value": [

 { ... },

 { ... },

 { ... }

 ],

 "@nextLink": "...?$skiptoken=342r89"

}

# Entity Reference

An entity reference (see [**[OData-Protocol]**](#odata)) MAY take the place of an entity instance in a JSON payload, based on the client request. It is serialized as a JSON object that MUST contain the [id](#_The_odata.id_Annotation_1) of the referenced entity and MAY contain the [type](#odataType) annotation and other custom annotations.

A collection of entity references is represented as a [collection of entities](#_Collection_of_Entities), with entity reference representations instead of entity representations as items in the array value of the value name/value pair.

The outermost JSON object in a response MUST contain a [context](#_The_odata.metadata_Annotation_1) annotation and MAY contain [count](#_Annotation_odata.count), [nextLink](#odataNext), or [deltaLink](#odataDelta) annotations.

Example : entity reference to order 10643

{

 "@context": "http://host/service/$metadata#$ref",

 "@id": "Orders(10643)"

}

Example : collection of entity references

{

 "@context": "http://host/service/$metadata#Collection($ref)",

 "value": [

 { "@id": "Orders(10643)" },

 { "@id": "Orders(10759)" }

 ]

}

# Delta Payload

The non-format specific aspects of the delta handling are described in the section “Requesting Changes” in [**[OData-Protocol]**](#odata).

Responses from a delta request are returned as a JSON object. The JSON object MUST contain an array-valued property named value containing all [added](#_Added/Changed_Entity), [changed](#_Representing_Added/Changed_Entities_1), or [deleted](#_Deleted_Entity) entities, as well as [added](#_Added_Link) or [deleted](#_Deleted_Link) links between entities, and MAY contain additional, unchanged entities.

If the delta response contains a partial list of changes, it MUST include a [next link](#_Annotation_odata.nextLink) for the client to retrieve the next set of changes.

The last page of a delta response SHOULD contain a [delta link](#_Annotation_odata.deltaLink) for retrieving subsequent changes once the current set of changes has been applied to the initial set.

If the response from the delta link contains a count annotation, the returned number MUST include all added, changed, or deleted entities to be returned, as well as added or deleted links.

Example : a 4.01 delta response with five changes, in order of occurrence

1. ContactName for customer 'BOTTM' was changed to "Susan Halvenstern"
2. Order 10643 was removed from customer 'ALFKI'
3. Order 10645 was added to customer 'BOTTM'
4. The shipping information for order 10643 was updated
5. Customer 'ANTON' was deleted

{

 "@context":"http://host/service/$metadata#Customers/$delta",

 "@count":5,

 "value":

 [

 {

 "@id":"Customers('BOTTM')",

 "ContactName":"Susan Halvenstern"

 },

 {

 "@context":"#Customers/$deletedLink",

 "source":"Customers('ALFKI')",

 "relationship":"Orders",

 "target":"Orders(10643)"

 },

 {

 "@context":"#Customers/$link",

 "source":"Customers('BOTTM')",

 "relationship":"Orders",

 "target":"Orders(10645)"

 },

 {

 "@context":"#Orders/$entity",

 "@id":"Orders(10643)",

 "ShippingAddress":{

 "Street":"23 Tsawassen Blvd.",

 "City":"Tsawassen",

 "Region":"BC",

 "PostalCode":"T2F 8M4"

 },

 },

 {

 "@context":"#Customers/$deletedEntity",

 "@removed": {

 "reason":"deleted"

 },

 "@id":"Customers('ANTON')"

 }

 ],

 "@deltaLink": "Customers?$expand=Orders&$deltatoken=8015"

}

## Added/Changed Entity

Added or changed entities within a delta response are represented as [entities](#_Entity).

Added entities MUST include all available selected properties and MAY include additional, unselected properties. Collection-valued properties are treated as atomic values; any collection-valued properties returned from a delta request MUST contain all current values for that collection.

Changed entities MUST include all available selected properties that have changed and MAY include additional properties.

If a property of an entity is dependent upon the property of another entity within the expanded set of entities being tracked, then both the change to the dependent property as well as the change to the principle property or [added](#_Added_Link_1)/[deleted](#_Representing_Deleted_Links_1) link corresponding to the change to the dependent property are returned in the delta response.

Entities that are not part of the entity set specified by the context URL MUST include the [context](#_Annotation_odata.metadata) annotation to specify the entity set of the entity, regardless of the specified [metadata](#_Controlling_the_Amount) value.

Entities include annotations for selected navigation links based on [metadata](#_Controlling_the_Amount).

OData 4.0 payloads MUST NOT include expanded navigation properties inline; all changes MUST be represented as a flat array of added, deleted, or changed entities, along with added or deleted links.

OData 4.01 delta payloads MAY include expanded navigation properties inline. Related single entities are represented as either an [added/changed](#_Representing_Added/Changed_Entities_1) entity, an [entity reference](#_Entity_Reference), or the null value (if no entity is related as the outcome of the change). Collection-valued navigation properties are represented either as a delta representation or as a full representation of the collection.

If the expanded navigation property represents a delta, it MUST be represented as an array-valued annotation [delta](#_Annotation_delta) on the navigation property. [Added/changed](#_Representing_Added/Changed_Entities_1) entities or [entity references](#_Entity_Reference) are added to the collection. Deleted entities MAY be specified in a nested delta representation to represent entities no longer part of the collection. If the deleted entity specifies a reason as deleted, then the entity is both removed from the collection and deleted, otherwise it is removed from the collection and only deleted if the navigation property is a containment navigation property. The array MUST NOT contain [added](#_Added_Link_1) or [deleted links](#_Representing_Deleted_Links_1).

Example : 4.01 delta response customers with expanded orders represented inline as a delta

1. Customer 'BOTTM':
	1. ContactName was changed to "Susan Halvenstern"
	2. Order 10645 was added
2. Customer 'ALFKI':
	1. Order 10643 was removed
3. Customer 'ANTON' was deleted

{

 "@context":"http://host/service/$metadata#Customers/$delta",

 "@count":3,

 "value":

 [

 {

 "@id":"Customers('BOTTM')",

 "ContactName":"Susan Halvenstern",

 "Orders@delta":[

 {

 "@id":"Orders(10645)"

 }

 ]

 },

 {

 "@id":"Customers('ALFKI')",

 "Orders@delta":[

 {

 "@context":"#Orders/$deletedEntity",

 "@removed": {

 "reason": "changed"

 },

 "@id":"Orders(10643)"

 }

 ]

 },

 {

 "@context":"#Customers/$deletedEntity",

 "@removed": {

 "reason": "deleted"

 },

 "@id":"Customers('ANTON')"

 }

 ],

 "@deltaLink": "Customers?$expand=Orders&$deltatoken=8015"

}

If the expanded navigation property is a full representation of the collection, it MUST be represented as an expanded navigation property, and its array value represents the full set of entities related according to that relationship and satisfying any specified expand options. Members of the array MUST be represented as [added/changed](#_Representing_Added/Changed_Entities_1) entities or [entity references and MUST NOT include added links, deleted links, or deleted entities.](#_Representing_Deleted_Entities_1) Any entity not represented in the collection has either been removed, deleted, or changed such that it no longer satisfies the expand options in the defining query. In any case, clients SHOULD NOT receive additional notifications for such removed entities.

## Deleted Entity

Deleted entities in JSON are returned as deleted-entity objects. Delta responses MUST contain a deleted-entity object for each deleted entity, including deleted expanded entities that are not related through a containment navigation property. The service MAY additionally include expanded entities related through a containment navigation property in which case it MUST include those in any returned count of enumerated changes.

The representation of deleted-entity objects differs between OData 4.0 and OData 4.01.

In OData 4.0 payloads the deleted-entity object MUST include the following properties, regardless of the specified [metadata](#_Controlling_the_Amount) value:

* Annotation [context](#_The_odata.metadata_Annotation_1) – the context URL fragment MUST be #{entity-set}/$deletedEntity, where {entity-set} is the entity set of the deleted entity
* id – The [id](#_The_odata.id_Annotation_1) of the deleted entity (same as the [id](#_The_odata.id_Annotation_1) returned or computed when calling GET on resource), which may be absolute or [relative](#_Relative_URLs)

In OData 4.0 payloads the deleted-entity object MAY include the following optional property, regardless of the specified [metadata](#_Controlling_the_Amount) value, and MAY include annotations:

* reason – either deleted, if the entity was deleted (destroyed), or changed if the entity was removed from membership in the result (i.e., due to a data change).

Example : deleted entity in OData 4.0 response – note that id is a property, not an annotation

{

 "@context":"#Customers/$deletedEntity",

 "reason":"deleted",

 "id":"Customers('ANTON')"

}

In OData 4.01 payloads the deleted-entity object MUST include the following properties, regardless of the specified [metadata](#_Controlling_the_Amount) value:

* Annotation removed, whose value is an object that MAY contain a property named reason. If present, the value of reason MUST be either deleted if the entity was deleted (destroyed), or changed if the entity was removed from membership in the result either due to change in value such that the entity no longer matches the defining query or because the entity was removed from the collection. The object MAY include annotations, and clients SHOULD NOT error due to the presence of additional properties that MAY be defined by future versions of this specification. For [ordered payloads](#_Payload_Ordering_Constraints_1), the annotation removed MUST immediately follow the [context](#_The_odata.metadata_Annotation_1) annotation, if present, otherwise it MUST be the first property in the deleted entity.
* Annotation [id](#_Annotation_id_(odata.id)) or all of the entity’s key fields. The id annotation MUST appear if any of the entity's key fields are omitted from the response *or* the entity-id is not identical to the canonical URL of the entity.

For full metadata the [context](#_The_odata.metadata_Annotation_1) annotation MUST be included. It also MUST be included if the entity set of the deleted entity cannot be determined from the surrounding context.

The deleted-entity object MAY include additional properties of the entity as well as annotations.

Example : deleted entity in OData 4.01 response with id annotation (prefixed with an @)

{

 "@context":"#Customers/$deletedEntity",

 "@removed":{

 "reason":"deleted",

 "@myannoation.deletedBy":"Mario"

 },

 "@id":"Customers('ANTON')"

}

Example : entity removed OData 4.01 response without id annotation and instead all key fields (ID is the single key field of Customer)

{

 "@removed":{},

 "ID":"ANTON"

}

## Added Link

Links within a delta response are represented as link objects.

Delta responses MUST contain a link object for each added link that corresponds to a $expand path in the initial request.

The link object MUST include the following properties, regardless of the specified [metadata](#_Controlling_the_Amount) value, and MAY include annotations:

* [context](#_The_odata.metadata_Annotation_1) – the context URL fragment MUST be #{entity-set}/$link, where {entity-set} is the entity set containing the source entity
* source – The [id](#_The_odata.id_Annotation_1) of the entity from which the relationship is defined, which may be absolute or [relative](#_Relative_URLs)
* relationship – The path from the source object to the navigation property which MAY traverse one or more complex properties, type cast segments, or members of ordered collections
* target – The [id](#_The_odata.id_Annotation_1) of the related entity, which may be absolute or [relative](#_Relative_URLs)

## Deleted Link

Deleted links within a delta response are represented as deleted-link objects.

Delta responses MUST contain a deleted-link object for each deleted link that corresponds to a $expand path in the initial request, unless either of the following is true:

* The source or target entity has been deleted
* The maximum cardinality of the related entity is one and there is a subsequent [link object](#_Added_Link) that specifies the same source and relationship.

The deleted-link object MUST include the following properties, regardless of the specified [metadata](#_Controlling_the_Amount) value, and MAY include annotations:

* [context](#_The_odata.metadata_Annotation_1) – the context URL fragment MUST be #{entity-set}/$deletedLink, where {entity-set} is the entity set containing the source entity
* source – The [id](#_The_odata.id_Annotation_1) of the entity from which the relationship is defined, which may be absolute or [relative](#_Relative_URLs)
* relationship – The path from the source object to the navigation property which MAY traverse one or more complex properties, type cast segments, or members of ordered collections
* target – The [id](#_The_odata.id_Annotation_1) of the related entity for multi-valued navigation properties, which may be absolute or [relative](#_Relative_URLs). For delta payloads that do not specify an OData-Version header value of 4.0, the target MAY be omitted for single-valued navigation properties.

# Bound Function

A bound function is advertised via a name/value pair where the name is a hash (#) character followed by the namespace- or alias-qualified name of the function. The namespace or alias MUST be defined or the namespace referenced in the metadata document of the service, see [[**[OData-CSDLXML]**](#ODataCSDLXMLRef).](#odataCSDL). A specific function overload can be advertised by appending the parentheses-enclosed, comma-separated list of non-binding parameter names to the qualified function name, see rule qualifiedFunctionName in [**[OData-ABNF]**](#abnf).

A function that is bound to a single structured type MAY be advertised within the JSON object representing that structured type.

Functions that are bound to a collection MAY be advertised within the JSON object containing the collection. If the collection is the top-level response, the function advertisement name/value pair is placed next to the value name/value pair representing the collection. If the collection is nested within an instance of a structured type, then in 4.01 payloads the name of the function advertisement is prepended with the name of the collection-valued property and is placed next to the collection-valued property, [expanded navigation property](#_Expanded_Navigation_Property), or [navigationLink](#_Annotation_odata.navigationLink_and_1) annotation, if present. 4.0 payloads MUST NOT advertise functions prefixed with property names.

If the function is available, the value of the advertisement is an object. OData 4.01 services MAY advertise the non-availability of the function with the value null.

If [metadata=full](#_odata.metadata=full) is requested, each value object MUST have at least the two name/value pairs title and target. It MAY contain [annotations](#_Instance_Annotations). The order of the name/value pairs MUST be considered insignificant.

The target name/value pair contains a URL. Clients MUST be able to invoke the function or the specific function overload by passing the parameter values via query options for [parameter aliases](http://docs.oasis-open.org/odata/odata/v4.0/errata03/os/complete/part1-protocol/odata-v4.0-errata03-os-part1-protocol-complete.html#_Parameter_Aliases_1) that are identical to the parameter name preceded by an at (@) sign. Clients MUST check if the obtained URL already contains a query part and appropriately precede the parameters either with an ampersand (&) or a question mark (?).

The title name/value pair contains the function or action title as a string.

If [metadata=minimal](#_odata=minimalmetadata) is requested, the target name/value pair MUST be included if its value differs from the canonical function or action URL.

Example : minimal representation of a function where all overloads are applicable

{

 "@context": "http://host/service/$metadata#Employees/$entity",

 "#Model.RemainingVacation": {},

 ...

}

Example : full representation of a specific overload with parameter alias for the Year parameter

{

 "@context": "http://host/service/$metadata#Employees/$entity",

 "#Model.RemainingVacation(Year)": {

 "title": "Remaining vacation from year.",

 "target": "Employees(2)/RemainingVacation(Year=@Year)"

 },

 ...

}

Example : full representation in a collection

{

 "@context": "http://host/service/$metadata#Employees",

 "#Model.RemainingVacation": {

 "title": "Remaining Vacation",

 "target": "Managers(22)/Employees/RemainingVacation"

 },

 "value": [ ... ]

}

Example : full representation in a nested collection

{

 "@context": "http://host/service/$metadata#Employees/$entity",

 "@type": "Model.Manager",

 "ID":22,

 ...

 "Employees#RemainingVacation": {

 "title": "RemainingVacation",

 "target": "Managers(22)/Employees/RemainingVacation"

 }

}

# Bound Action

A bound action is advertised via a name/value pair where the name is a hash (#) character followed by the namespace- or alias-qualified name of the action. The namespace or alias MUST be defined or the namespace referenced in the metadata document of the service, see [[**[OData-CSDLXML]**](#ODataCSDLXMLRef).](#odataCSDL).

An action that is bound to a single structured type is advertised within the JSON object representing that structured type.

Actions that are bound to a collection MAY be advertised within the JSON object containing the collection. If the collection is the top-level response, the action advertisement name/value pair is placed next to the value name/value pair representing the collection. If the collection is nested within an instance of a structured type, then in 4.01 payloads the name of the action advertisement is prepended with the name of the collection-valued property and is placed next to the name/value pair representing the collection-valued property, [expanded navigation property](#_Expanded_Navigation_Property), or [navigationLink](#_Annotation_odata.navigationLink_and_1) annotation, if present. 4.0 payloads MUST NOT advertise actions prefixed with property names.

If the action is available, the value of the advertisement is an object. OData 4.01 services MAY advertise the non-availability of the action with the value null.

If [metadata=full](#_odata.metadata=full) is requested, each value object MUST have at least the two name/value pairs title and target. It MAY contain [annotations](#_Instance_Annotations). The order of these name/value pairs MUST be considered insignificant.

The target name/value pair contains a bound function or action URL.

The title name/value pair contains the function or action title as a string.

If [metadata=minimal](#_odata=minimalmetadata) is requested, the target name/value pair MUST be included if its value differs from the canonical function or action URL.

Example : minimal representation in an entity

{

 "@context": "http://host/service/$metadata#LeaveRequests/$entity",

 "#Model.Approve": {},

 ...

}

Example : full representation in an entity:

{

 "@context": "http://host/service/$metadata#LeaveRequests/$entity",

 "#Model.Approve": {

 "title": "Approve Leave Request",

 "target": "LeaveRequests(2)/Approve"

 },

 ...

}

Example : full representation in a collection

{

 "@context": "http://host/service/$metadata#LeaveRequests",

 "#Model.Approve": {

 "title": "Approve All Leave Requests",

 "target": "Employees(22)/Model.Manager/LeaveRequests/Approve"

 },

 "value": [ ... ]

}

Example : full representation in a nested collection

{

 "@context": "http://host/service/$metadata#Employees/$entity",

 "@type": "Model.Manager",

 "ID":22,

 ...

 "LeaveRequests#Model.Approve": {

 "title": "Approve All Leave Requests",

 "target": "Employees(22)/Model.Manager/LeaveRequests/Approve"

 }

}

# Action Invocation

Action parameter values are encoded in a single JSON object in the request body.

Each non-binding parameter value is encoded as a separate name/value pair in this JSON object. The name is the name of the parameter. The value is the parameter value in the JSON representation appropriate for its type.

Any parameter values not specified in the JSON object are assumed to have the null value.

Example :

{

 "param1": 42,

 "param2": {

 "Street": "One Microsoft Way",

 "Zip": 98052

 },

 "param3": [ 1, 42, 99 ],

 "param4": null

}

In order to invoke an action with no non-binding parameters, the client passes an empty JSON object in the body of the request. 4.01 Services MUST also support clients passing an empty request body for this case.

# Instance Annotations

Annotations are an extensibility mechanism that allows services and clients to include information other than the raw data in the request or response. Annotations are used to include control information in many payloads.

Annotations are name/value pairs that have an at (@) and a dot (.) as part of the name. The part after the "at" sign (@) is the *annotation* *identifier*. It consists of the namespace or alias of the schema that defines the term, followed by a dot (.), followed by the name of the term, optionally followed by a hash (#) and a qualifier. The namespace or alias MUST be defined in the metadata document, see [[**[OData-CSDLXML]**](#ODataCSDLXMLRef).](#odataCSDL).

The namespace or alias odata is reserved for future extensions of the protocol and format. Custom annotations are annotations that have a namespace or alias that is different from odata.

Annotations can be applied to any name/value pair in a JSON payload that represents a value of any type from the entity data model (see[[**[OData-CSDLXML]**](#ODataCSDLXMLRef).](#odataCSDL)). Clients should never error due to an unexpected annotation in a JSON payload.

Annotations are always expressed as name/value pairs. For entity data model constructs represented as JSON objects the annotation name/value pairs are placed within the object; for constructs represented as JSON arrays or primitives they are placed next to the annotated model construct. When annotating a payload that represents a [single primitive or collection value](#_Individual_Property_or), the annotations for the value appear next to the value property and are not prefixed with a property name.

Example :

{

 "@context": "http://host/service/$metadata#Customers",

 "@com.example.customer.setkind": "VIPs",

 "value": [

 {

 "@com.example.display.highlight": true,

 "ID": "ALFKI",

 "CompanyName@com.example.display.style": { "title": true, "order": 1 },

 "CompanyName": "Alfreds Futterkiste",

 "Orders@com.example.display.style#simple": { "order": 2 }

 }

 ]

}

## Annotate a JSON Object

When annotating a name/value pair for which the value is represented as a JSON object, each annotation is placed within the object and represented as a single name/value pair.

The name always starts with the "at" sign (@), followed by the annotation identifier.

The value MUST be an appropriate value for the annotation.

## Annotate a JSON Array or Primitive

When annotating a name/value pair for which the value is represented as a JSON array or primitive value, each annotation that applies to this name/value pair MUST be represented as a single name/value pair and placed immediately prior to the annotated name/value pair, with the exception of the [nextLink](#_Annotation_nextLink_(odata.nextLink) annotation which can appear immediately before or after the collection it annotates.

The name is the same as the name of the property or name/value pair being annotated, followed by the “at” sign (@), followed by the annotation identifier.

The value MUST be an appropriate value for the annotation.

# Error Response

The error response MUST be a single JSON object. This object MUST have a single name/value pair named error. The value must be a JSON object.

This object MUST contain name/value pairs with the names code and message, and it MAY contain name/value pairs with the names target, details and innererror.

The value for the code name/value pair is a language-independent string. Its value is a service-defined error code. This code serves as a sub-status for the HTTP error code specified in the response.

The value for the message name/value pair MUST be a human-readable, language-dependent representation of the error. The Content-Language header MUST contain the language code from [**[RFC5646]**](#rfc5646) corresponding to the language in which the value for message is written.

The value for the target name/value pair is the target of the particular error (for example, the name of the property in error).

The value for the details name/value pair MUST be an array of JSON objects that MUST contain name/value pairs for code and message, and MAY contain a name/value pair for target, as described above.

The value for the innererror name/value pair MUST be an object. The contents of this object are service-defined. Usually this object contains information that will help debug the service. The innererror name/value pair SHOULD only be used in development environments in order to guard against potential security concerns around information disclosure.

Error responses MAY contain [annotations](#_Instance_Annotations) in any of its JSON objects.

Example :

{

 "error": {

 "code": "501",

 "message": "Unsupported functionality",

 "target": "query",

 "details": [

 {

 "code": "301",

 "target": "$search",

 "message": "$search query option not supported"

 }

 ],

 "innererror": {

 "trace": [...],

 "context": {...}

 }

 }

}

# Extensibility

Implementations can add [custom annotations](#_Instance_Annotations) of the form @namespace.termname or property@namespace.termname to any JSON object, where property MAY or MAY NOT match the name of a name/value pair within the JSON object. However, the namespace MUST NOT start with odata and SHOULD NOT be required to be understood by the receiving party in order to correctly interpret the rest of the payload as the receiving party MUST ignore unknown annotations not defined in this version of the OData JSON Specification.

# Security Considerations

This specification raises no security issues.

This section is provided as a service to the application developers, information providers, and users of OData version 4.0 giving some references to starting points for securing OData services as specified. OData is a REST-full multi-format service that depends on other services and thus inherits both sides of the coin, security enhancements and concerns alike from the latter.

For JSON-relevant security implications please cf. at least the relevant subsections of [**[RFC7159]**](#rfc7159) as starting point.

# Conformance

Conforming clients MUST be prepared to consume a service that uses any or all of the constructs defined in this specification. The exception to this are the constructs defined in [Delta Response](#_Delta_Response), which are only required for clients that request changes.

In order to be a conforming *consumer* of the OData JSON format, a client or service:

1. MUST either:
	1. understand metadata=minimal (section 3.1.1) or
	2. explicitly specify metadata=none (section 3.1.3) or metadata=full (section 3.1.2) in the request (client)
2. MUST be prepared to consume a response with full metadata
3. MUST be prepared to receive all data types (section 7.1)
	1. defined in this specification (client)
	2. exposed by the service (service)
4. MUST interpret all odata annotations defined according to the OData-Version header of the payload (section 0)
5. MUST be prepared to receive any annotations, including custom annotations and odata annotations not defined in the OData-Version header of the payload (section 20)
6. MUST NOT require streaming=true in the Content-Type header (section 4.4)
7. MUST be a conforming consumer of the OData 4.0 JSON format, for payloads with an OData-Version header value of 4.0.
	1. MUST accept the odata. prefix, where defined, on [format parameters](#_Controlling_the_Amount) and [control information](#_Control_Information_1)
	2. MUST accept the # prefix in [@odata.type](#_Annotation_odata.type_(type)) values
	3. MUST be prepared to handle binding through the use of the [@odata.bind](#_Bind_Operation) property in payloads to a PATCH, PUT, or POST request
	4. MUST accept TargetId within in a [deleted link](#_Representing_Deleted_Links_1) for a relationship with a maximum cardinality of one
	5. MUST accept the string values INF, -INF, and Nan for numeric value exceptions for single and double values
	6. MUST support property annotations that appear immediately before or after the property they annotate
8. MAY be a conforming consumer of the OData 4.01 JSON format, for payloads with an OData-Version header value of 4.01.
	1. MUST be prepared to interpret [control information](#_Control_Information_1) with or without the odata. prefix
	2. MUST be prepared for [@odata.type](#_Annotation_odata.type_(type)) primitive values with or without the # prefix
	3. MUST be prepared to handle binding through inclusion of an entity reference within a collection-valued navigation property in the body of a PATCH, PUT, or POST request
	4. MUST be prepared for TargetId to be included or omitted in a [deleted link](#_Representing_Deleted_Links_1) for a relationship with a maximum cardinality of one
	5. MUST be prepared to interpret the Core.[NumericValueException](#_Primitive_Value) annotation, defined in [**[OData-VocCore]**](#VocCore), for numeric value exceptions
	6. MUST be prepared to handle related entities inline within a [delta payload](#_Representing_Added/Changed_Entities_1) as well as a nested delta representation for the collection
	7. MUST be prepared to handle decimal values written in exponential notation

In order to be a conforming *producer* of the OData JSON format, a client or service:

1. MUST support generating OData 4.0 JSON compliant payloads with an OData-Version header value of 4.0.
	1. MUST NOT omit the odata. prefix from [format parameters](#_Controlling_the_Amount) or [control information](#_Control_Information_1)
	2. MUST NOT omit the # prefix from [@odata.type](#_Annotation_odata.type_(type)) values
	3. MUST NOT include entity values or entity references within a collection-valued navigation property in the body of a PATCH, PUT, or POST request
	4. MUST NOT return decimal values written in exponential notation unless the [ExponentialDecimals](#_Controlling_the_Representation) format parameter is specified.
	5. MUST NOT advertise available actions or functions using name/value pairs prefixed with a property name
	6. MUST NOT return a null value for name/value pairs representing actions or functions that are not available
	7. SHOULD represent numeric value exceptions for single and double values using the string property values INF, -INF, and Nan
	8. MAY include the Core.NumericValueException annotation, defined in [**[OData-VocCore]**](#VocCore), for any numeric value exception
2. MAY support generating OData 4.01 JSON compliant payloads for requests with an OData-Version header value of 4.01.
	1. MUST return property annotations immediately before the property they annotate
	2. SHOULD omit the odata. prefix from [format parameters](#_Controlling_the_Amount) and [odata control information](#_Control_Information_1)
	3. SHOULD omit the # prefix from [@type](#_Annotation_odata.type_(type)) primitive values
	4. MAY include inline related entities or nested delta collections within a delta payload
	5. MAY include TargetId within a [deleted link](#_Representing_Deleted_Links_1) for a relationship with a maximum cardinality of 1
	6. MAY return decimal values written in exponential notation
	7. MUST represent numeric value exceptions using the Core.NumericValueException annotation, defined in [**[OData-VocCore]**](#VocCore), for any numeric value exception
	8. MUST NOT use the string property values INF, -INF, and Nan for numeric value exceptions

In addition, in order to conform to the OData JSON format, a service:

1. MUST comply with one of the conformance levels defined in [[OData-Protocol]](#odata)
2. MUST support the application/json media type in the Accept header (section 3)
3. MUST return well-formed JSON payloads
4. MUST support odata.metadata=full (section 3.1.2)
5. MUST include the odata.nextLink annotation in partial results for entity collections (section 4.5.5)
6. MUST support entity instances with external metadata (section 4.5.1)
7. MUST support properties with externally defined data types (section 4.5.3)
8. MUST NOT violate any other aspects of this OData JSON specification
9. SHOULD support the $format system query option (section 3)
10. MAY support the odata.streaming=true parameter in the Accept header (section 4.4)
11. MAY return full metadata regardless of odata.metadata (section 3.1.2)
12. MUST NOT omit null or default values unless the omit-values preference is specified in the Prefer request header and the omit-values preference is included in the Preference-Applied response header
13. MUST return OData JSON 4.0-compliant responses for requests with an OData-MaxVersion header value of 4.0
14. MUST support OData JSON 4.0-compliant payloads in requests with an OData-Version header value of 4.0
15. MUST support returning, in the final response to an asynchronous request, the application/json payload that would have been returned had the operation completed synchronously, wrapped in an application/http message

In addition, in order to comply with the OData 4.01 JSON format, a service:

1. SHOULD return the OData JSON 4.01 format for requests with an OData-MaxVersion header value of 4.01
2. MUST support the OData JSON 4.01 format in request payloads for requests with an OData-Version header value of 4.01
3. MUST honor the [odata.etag](#_Annotation_odata.etag_(etag)) annotation within PUT, PATCH or DELETE payloads, if specified
4. MUST support returning, in the final response to an asynchronous request, the application/json payload that would have been returned had the operation completed synchronously
5. Acknowledgments

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

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
| Working Draft 01 | 2016-06-22 | Michael PizzoRalf Handl | Import material from OData 4.0 Errata 3 JSON document and initial application of 4.01 features |
| Committee Specification Draft 01 | 2016-12-08 | Michael Pizzo Ralf Handl | Integrated 4.01 features |