OData JSON Format Version 4.0

Committee Specification Draft ~~01~~02 /
Public Review Draft ~~01~~02

~~26 April~~24 June 2013

Specification URIs

This version:

~~http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd01/odata-json-format-v4.0-csprd01.doc (Authoritative)~~
~~http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd01/odata-json-format-v4.0-csprd01.html~~
~~http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd01/odata-json-format-v4.0-csprd01.pdf~~
http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd02/odata-json-format-v4.0-csprd02.doc (Authoritative)
http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd02/odata-json-format-v4.0-csprd02.html
http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd02/odata-json-format-v4.0-csprd02.pdf

Previous version:

~~N/A~~
http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd01/odata-json-format-v4.0-csprd01.doc (Authoritative)
http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd01/odata-json-format-v4.0-csprd01.html
http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd01/odata-json-format-v4.0-csprd01.pdf

Latest version:

~~http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.doc (Authoritative)~~
~~http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.html~~
~~http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.pdf~~
http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.doc (Authoritative)
http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.html
http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.pdf

Technical Committee:

OASIS Open Data Protocol (OData) TC

Chairs:

Barbara Hartel (barbara.hartel@sap.com), SAP AG
Ram Jeyaraman (Ram.Jeyaraman@microsoft.com), Microsoft

Editors:

Ralf Handl (ralf.handl@sap.com), SAP AG
Mike Pizzo (mikep@microsoft.com), Microsoft
~~Mark Biamonte (mark.biamonte@progress.com), Progress Software~~

Mark Biamonte (mark.biamonte@progress.com), Progress Software

Related work:

This specification is related to:

- *OData Version 4.0*, a multi-part Work Product which includes:
    - *OData Version 4.0 Part 1: Protocol*. Latest version. ~~http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html~~http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html
    - *OData Version 4.0 Part 2: URL Conventions*. Latest version. http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part2-url-conventions.html
    - *OData Version 4.0 Part 3: Common Schema Definition Language (CSDL)*. Latest version. http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part3-csdl.html
    - ABNF components: *OData ABNF Construction Rules Version 4.0* and *OData ABNF Test Cases*. 24 June 2013. http://docs.oasis-open.org/odata/odata/v4.0/~~csprd01~~csprd02/abnf/
    - Vocabulary components: *OData Core Vocabulary, OData Measures Vocabulary* and *OData Capabilities Vocabulary*. 24 June 2013. http://docs.oasis-open.org/odata/odata/v4.0/csprd02/vocabularies/
- *OData Atom Format Version 4.0.* 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) ~~is a set of specifications for representing and interacting with structured content. This document extends the core OData Protocol specification~~enables the creation of REST-based data services, which allow resources, identified using Uniform Resource Identifiers (URIs) and defined in an Entity Data Model (EDM), to be published and edited by Web clients using simple HTTP messages. OData version 4.0 defines the core semantics and facilities of the protocol, a set of recommended (but not required) rules for constructing URLs to identify the data and metadata exposed by an OData service as well as a set of reserved URL query string operators, an Entity Data Model (EDM), and an XML representation of the entity data model exposed by an OData service. OData JSON Format version 4.0 extends the former 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.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at ~~http://www.oasis-open.org/committees/odata/~~http://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 (~~http://www.oasis-open.org/committees/odata/ipr.php~~http://www.oasis-open.org/committees/odata/ipr.php).

Citation format:

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

[OData-JSON-Format-v4.0]

OData JSON Format Version 4.0. ~~26 April~~24 June 2013. OASIS Committee Specification Draft ~~01~~02 / Public Review Draft ~~01. http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd01/odata-json-format-v4.0-csprd01.html~~02. http://docs.oasis-open.org/odata/odata-json-format/v4.0/csprd02/odata-json-format-v4.0-csprd02.html.

Notices

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

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

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

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

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, 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 http://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1 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]**~~],~~; 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 **[RFC4627]**.

An OData JSON payload may represent:

- ~~a single primitive value~~
- ~~a sequence of primitive values~~
- ~~a single complex type value~~
- ~~a sequence of complex type values~~
- ~~a single entity or entity reference~~
- ~~a sequence of entities or entity references~~
- ~~a sequence of changes~~
- ~~a service document~~a single primitive value
- a collection of primitive values
- a single complex type value
- a collection of complex type values
- a single entity or entity reference
- a collection of entities or entity references
- a collection of changes
- a service document describing the top-level resources exposed by the service
- ~~an error~~
- ~~This document contains many example JSON payloads or partial JSON payloads. These examples are non-normative and informative only.~~an error.

1.1 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]**.

1.2 Normative References

[GeoJSON]	Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., Schmidt, C., "The GeoJSON Format Specification", Revision 1.0, June 2008. http://geojson.org/geojson-spec.html.
[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-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 Capabilities 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.
~~http://www.ietf.org/rfc/rfc2119.txt.~~http://www.ietf.org/rfc/rfc2119.txt.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, IETF RFC3986, January 2005.
~~http://www.ietf.org/rfc/rfc3986.txt.~~http://www.ietf.org/rfc/rfc3986.txt.

[RFC3987] Duerst, M. and, M. Suignard, “Internationalized Resource Identifiers (IRIs)”, RFC 3987, January 2005.
~~http://www.ietf.org/rfc/rfc3987.txt.~~http://www.ietf.org/rfc/rfc3987.txt.

[RFC4627] Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON)”, RFC 4627, July 2006.
~~http://tools.ietf.org/html/rfc4627.~~http://tools.ietf.org/html/rfc4627.

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

~~1.3 Non-Normative References~~

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

1.3 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 1: text describing an example uses this paragraph style

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

All examples in this document are non-normative and informative only.

All other text is normative unless otherwise labeled.

2 JSON Format Design

JSON, as described in ~~[RFC4627]~~**[RFC4627],** 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 of entities) 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.

3 Requesting the JSON Format

The OData JSON format ~~MAY~~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 ~~MAY~~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.

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

3.1 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 `odata.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. ~~For the purpose of this section, we will take the following two assumptions:~~

- ~~The media-range for the `Accept` header is set to `application/json`.~~

~~•~~ Other `Accept` header parameters (e.g., `odata.streaming`) are orthogonal to the `odata.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 ~~`odata.metadata=minimal.`~~`odata.metadata=minimal`. If compute is more expensive than wire size or the client is incapable of computing control information, ~~`odata.metadata=full`~~`odata.metadata=full` directs the ~~server~~service to inline the control information that normally would be computed from metadata expressions in the payload. ~~`odata.metadata=none`~~`odata.metadata=none` is an option for clients that have out-of-band knowledge or don't require control information.

3.1.1 `odata.metadata=minimal`

The ~~client MAY specify~~ `odata.metadata=minimal` ~~to indicate~~format parameter indicates that the ~~server~~service SHOULD remove computable control information from the payload wherever possible. This is the default value for the `odata.metadata` parameter and will be assumed if no other value is specified in the `Accept` header or `$format` query option. The response payload MUST contain at least the following common ~~annotations~~annotations:

- ~~`odata.metadata`~~`odata.context`: the ~~metadata~~context URL of the payload~~.~~
- ~~`odata.etag`~~`odata.etag`: the ETag of the entity~~.~~, as appropriate
- ~~`odata.count`~~`odata.count`: the ~~inline~~total count of a ~~set~~collection of entities or collection of entity references, if requested~~.~~
- ~~`odata.nextLink`~~`odata.nextLink`: the next link of a ~~set~~collection of entities or collection of entity references~~.~~ for partial results
- ~~`odata.deltaLink`~~`odata.deltaLink`: 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 ~~MUST be~~are treated as exceptions to the computed values.

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

- ~~`odata.mediaEtag`~~`odata.mediaEtag`: the ETag of the stream~~.~~, as appropriate
- ~~`odata.mediaContentType`~~`odata.mediaContentType`: the content type of the stream~~.~~

3.1.2 `odata.metadata=full`

The ~~client MAY specify~~ `odata.metadata=full` ~~to~~ format parameter indicates that the service MUST include all control information explicitly in the payload. ~~The service MUST return all metadata in this case.~~

The full list of annotations that may appear in an `odata.metadata=full` response ~~are~~is as follows:

- ~~`odata.metadata`: the metadata~~`odata.context`: the context URL for a collection, entity, primitive value, or service document.
- ~~`odata.count`: the inline~~`odata.count`: the total count of a ~~set~~collection of entities or collection of entity references, if requested.
- ~~`odata.nextLink`~~`odata.nextLink`: the next link of a ~~set~~collection of entities or collection of entity references~~.~~ for partial results
- ~~`odata.deltaLink`~~`odata.deltaLink`: the delta link for obtaining changes to the result, if requested~~.~~
- ~~`odata.id`~~`odata.id`: the ID of the entity~~.~~
- ~~`odata.etag`~~`odata.etag`: the ETag of the entity~~.~~
- ~~`odata.kind`: the kind of object (entry, deleted-entry, link, or deleted-link) represented by the entry. If omitted, the entry represents an entity.~~
- ~~`odata.readLink`~~`odata.readLink`: the link used to read the entity, if the ~~odata.id does not represent a URL that can~~edit link cannot be used to read the entity~~.~~
- ~~`odata.editLink`~~`odata.editLink`: the link used to edit/update the ~~entry~~entity, if the entity is updatable and the `odata.id` does not represent a URL that can be used to edit the entity~~.~~
- ~~`odata.navigationLink`~~`odata.navigationLink`: the link used to retrieve the values of a navigation property~~.~~
- ~~`odata.associationLink`~~`odata.associationLink`: the link used to describe the relationship between this entity and related entities~~.~~
- ~~`odata.type`~~`odata.type`: the type ~~name~~ of the containing object or targeted property~~. The type annotation is only present~~ if the type of the object or targeted property cannot be heuristically determined~~.~~

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

- ~~`odata.mediaReadLink`~~`odata.mediaReadLink`: the link used to read the stream~~.~~
- ~~`odata.mediaEditLink`~~`odata.mediaEditLink`: the link used to edit/update the stream~~.~~
- ~~`odata.mediaEtag`~~`odata.mediaEtag`: the ETag of the stream~~.~~, as appropriate
- ~~`odata.mediaContentType`~~`odata.mediaContentType`: the content type of the stream~~.~~

3.1.3 `odata.metadata=none`

The ~~client MAY specify~~ `odata.metadata=none` ~~in order to request~~format parameter indicates that the service SHOULD omit control information~~. In this case, the service MAY omit `odata.*` annotations~~ other than ~~`odata.nextLink`~~`odata.nextLink`, ~~`odata.count`~~`odata.count` and ~~`odata.deltaLink`~~`odata.deltaLink`. These annotations MUST continue to be included, as applicable, even in the `odata.metadata=none` case.

3.2 Controlling the Representation of Numbers

The ~~client MAY specify the format parameter~~ `IEEE754Compatible` ~~for the `application/json` format. If specified the producer~~format parameter indicates that the service MUST serialize `Edm.Int64` and `Edm.Decimal` numbers as strings.

This ~~is due to the fact that~~ 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), so~~section 4.3.1.9) resulting in integers ~~lose~~losing precision past 15 digits, and decimals ~~lose~~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.

4 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.

4.1 Header `Content-Type`

Requests and responses in JSON 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 ~~add~~include the ~~`odata.metadata`~~`odata.metadata` parameter ~~with the same value that was specified in the `Accept` header~~to specify the amount of ~~the request. If no value was specified in the `Accept` header, `odata.`~~metadata~~`=minimal`~~ included in the response.

Responses MUST ~~be used~~include the `IEEE754Compatible` parameter if `Edm.Int64` and `Edm.Decimal` numbers are represented as strings.

Requests and responses MAY add the `odata.streaming` parameter with a value of `true` or `false`, see section ~~Payload Ordering Constraints~~Payload Ordering Constraints.

4.2 Message Body

Each message body ~~MUST be~~is represented as a single JSON object. This object is either the representation of an ~~entity~~entity, an ~~entity reference~~entity reference or a ~~complex type instance~~complex type instance, or it contains a name/value pair whose name MUST be `value` and whose value ~~MUST be~~is the correct representation for a ~~primitive value~~primitive value, a collection of primitive values, a ~~collection of primitive values, a collection of complex values~~collection of complex values, a ~~collection of entities~~collection of entities, or a collection of ~~entries~~objects that represent ~~changes to a previous result~~changes to a previous result.

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

4.3 Relative URLs

URLs present in a payload (whether request or response) MAY be represented as relative URLs~~ to the metadata URL. Processors expanding the URLs MUST use normal URL expansion rules and use the metadata URL as a base. The reference resolution rules defined in **[RFC3986]** imply that the part starting with `$metadata#` is ignored when resolving the relative URL~~.

If a context URL is present in the same JSON object as the relative URL or an enclosing object, the relative URL is relative to the next context URL. This rule also applies to context URLs; they can be relative to the context URL of an enclosing object.

If no context URL is present in a request, relative URLs are relative to the request URL.

If no context URL is present in a response, relative URLs are relative to the `Content-Location` header of the response.

In responses without a `Content-Location` header or context URL, relative URLs are relative to 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.

Example~~:~~ 2:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  ...
  "odata.editLink": "Customers('ALFKI')",
  ...
  "Orders@odata.navigationLink~~":"~~": "Customers('ALFKI')/Orders",
  ...
}
```

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

4.4 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 ~~servers~~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 `odata.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.

Processors MUST only assume streaming support if it is explicitly indicated in the `Content-Type` header via the `odata.streaming=true` parameter. ~~A payload with~~

Example 3: a payload with

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

can ~~thus~~ be assumed to support streaming, whereas a payload with

```
Content-Type: application/json;odata.metadata=minimal
```

cannot be assumed to support streaming.

JSON producers are encouraged to follow the payload ordering constraints whenever possible (and include the `odata.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 `odata.`~~`metadata`~~`context` annotation MUST be the first property in the JSON object.
- The `odata.type` annotation, if present, MUST appear next in the JSON object.
- The `odata.id` and `odata.etag` annotations MUST appear before any property or property annotation.
- All ~~property~~ annotations for a structural or navigation property MUST appear as a group immediately before the property they annotate. The one exception is the `odata.nextlink` annotation of an expanded collection which MAY appear after the navigation property it annotates.
- All other `odata`~~`.*`~~ annotations ~~MAY~~can appear anywhere in the payload ~~(~~as long as they ~~are~~do not ~~violating~~violate any of the above rules~~)~~.
- Annotations for navigation properties MUST appear after all structural properties.

4.5 Control Information

In addition to the “pure data” a message body MAY contain control information that is represented as ~~annotations~~annotations whose names start with `odata` followed by a dot.

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

4.5.1 Annotation `odata.~~metadata~~context`

The `odata.~~metadata~~context` annotation returns the ~~metadata~~context URL (see **[OData-Protocol]**) for the payload. The `odata.~~metdata~~context` annotation MUST be the first property of any JSON response that does not specify ~~`odata.metadata=none`~~`odata.metadata=none`.

The `odata.~~metadata~~context` annotation MUST also be included for entities whose entity set cannot be determined from the ~~metadata URL~~context URL of the collection. This URL ~~MAY~~can be absolute or relative to the ~~metadata~~context URL of the collection.

The `odata.~~metadata~~context` annotation MUST also be applied to navigation links for navigation properties not described in the metadata of the containing type. In this case the ~~metadata~~context URL MAY be relative to the ~~metadata~~context URL describing the parent entity and becomes the root ~~metadata~~context URL for the related entity or collection.

For more information on the format of the ~~metadata~~context URL, see ~~**[OData-Protocol]**~~**[OData-Protocol]**.

Request payloads in JSON do not require ~~metadata~~context URLs. ~~However, if the request does include the metadata URL, relative URLs~~It MAY be ~~used~~included as a base URL for relative URLs in the request payload.

Response payloads ~~MUST~~SHOULD NOT contain the ~~metadata~~context URL if ~~odata.metadata=none~~odata.metadata=none is requested.

Example~~:~~ 4:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  "odata.metadataEtag": "W/\"A1FF3E230954908F~~",~~\"",
  ...
}
```

4.5.2 Annotation `odata.metadataEtag`

The `odata.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.

For details on how ETags are used, see ~~[~~**[OData-Protocol]**~~]~~.

4.5.3 Annotation `odata.type`

~~The annotation `odata.type`~~The `odata.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 all other types, the URI may be absolute or relative to the URI of the containing type. The root `odata.type` may be absolute or relative to the root context URL.

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 `odata.type` annotation MUST appear 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 (~~set~~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 `odata.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 and do not need any additional annotations. If the value of a property is represented as a number without a dot (`.`), `e` or `E` embedded, the type should be interpreted as an integer value~~.~~, otherwise as a decimal, double, or single value.
- ~~Similarly, decimal, double, and single values use the same representation and do not need any additional annotations. If the value of a property is represented as a number with a single dot (`.`), and/or a single `e` or `E` embedded, the type should be interpreted as a decimal, double, or single value. The~~The floating-point values `NaN`, `INF`, and `-INF` are serialized as strings and MUST have an ~~`odata.type`~~`odata.type` annotation to specify the numeric type.
- 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`~~`IEEE754Compatible` 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.

~~If the `odata.type` annotation is present, its value MUST be the namespace- or alias-qualified name of the instance's type, in which case the type MUST be defined by the root of the current metadata URL, otherwise it MUST be a full URL to a metadata document with the namespace- or alias-qualified name of the instance's type appended as a URL fragment.~~

For more information on namespace- and alias-qualified names, see ~~**[OData-CSDL].**~~ **[OData-CSDL]**.

~~For example, the following represents an~~Example 5: entity ~~whose~~of type ~~is~~ `Model.VipCustomer`~~,~~ defined ~~within~~in the ~~`http://host/`~~metadata document of the same service~~`/$metadata` document:~~

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  "odata.type": ~~"~~"#Model.VipCustomer",
  "ID": 2,
  ...
}
```

~~The following represents an~~Example 6: entity ~~whose~~of type ~~is~~ `Model.VipCustomer`~~,~~ defined ~~within~~in the ~~`http://host/alternate/$`~~metadata document~~:~~ of a different service

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  "odata.type": "http://host/alternate/$metadata#Model.VipCustomer",
  "ID": 2,
  ...
}
```

4.5.4 Annotation `odata.count`

The `odata.count` annotation contains the ~~inlinecount~~count of a ~~set of entities~~collection of entities or a ~~collection of entity references~~collection of entity references, see ~~[~~**[OData-Protocol]**~~,~~ section ~~10~~11.2.~~3.7~~ ~~**The $inlinecount**~~4.5 System Query Option~~].~~ `$count`. Its value ~~MUST be~~is an `Edm.Int64` value corresponding to the total count of members in the collection represented by the request.

4.5.5 Annotation `odata.nextLink`

The `odata.nextLink` annotation indicates that a response is only a subset of the requested ~~set~~collection of entities or collection of entity references. It contains a URL that allows retrieving the next subset of the requested ~~set or~~ collection.

~~Expanded to-many navigation properties MAY be also annotated with this annotation.~~

This annotation can also be applied to expanded to-many navigation properties.

4.5.6 Annotation `odata.deltaLink`

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

4.5.7 Annotation `odata.id`

The `odata.id` annotation contains ~~a durable, opaque, globally unique identifier for an~~ the entity~~ or set of entities. Its content MUST be an IRI as defined in **[RFC3987]**. Services are encouraged to use a dereferenceable URL for the~~ id~~, but consumers MUST NOT assume this IRI can be de-referenced, nor assume any semantics from its structure.~~

; see **[OData-Protocol]**. By convention ~~its value~~the entity-id is identical to the canonical URL of the entity, as defined in ~~**[OData-URL]**. The annotation `odata.id` will only appear in minimal metadata and no metadata cases if its value deviates from the canonical URL of the entity.~~**[OData-URL]**.

The `odata.id` annotation MUST appear if `odata.metadata=full` is requested *or* if the entity-id is not identical to the canonical URL of the entity after IRI-toURI conversion as defined in **[RFC3987]**, resolution of relative URLs and percent-encoding normalization as defined in **[RFC3986]**. If the `odata.id` is represented, it MAY be a relative URL.

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

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

Entities with `odata.id` equal to `null` cannot be compared to other entities, reread, or updated. If `odata.metadata=minimal` is specified and the `odata.id` is not present in the entity then the canonical URL MUST be used as the entity-id.

4.5.8 Annotation `odata.editLink` and `odata.readLink`

The `odata.editLink` annotation contains ~~a~~the edit URL ~~that can be used to~~ of the entity; see **[OData-Protocol]**.

The `odata.readLink` annotation contains the read~~, update,~~ URL of the entity or ~~delete the entity. It MUST appear~~ collection; see **[OData-Protocol]**.

The default value of both the edit URL and read URL is the entity's entity-id 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 `odata.editLink` nor the `odata.readLink` annotation is present in an entity, the client uses this default value for the edit URL.

For updatable entities~~ if `odata.metadata=full`~~:

- The `odata.editLink` annotation is written if `odata.metadata=full` is requested *or* if ~~its value~~the edit URL differs from the default value of the ~~`odata.id`~~edit URL.
- The `odata.readLink` annotation ~~contains a~~is written if the read URL ~~that can be used~~is different from the edit URL. If no `odata.readLink` annotation is present, the read URL is identical to ~~read the entity. It MUST appear if `odata.metadata=full`~~the edit URL.

For read-only entities:

- The `odata.readLink` annotation is written if `odata.metadata=full` is requested ~~and~~*or* if its value differs from the default value of the ~~`odata.id` and the `odata.editlink` is not present~~read URL.

~~4.5.9 Annotation `odata.kind`~~

- The `odata.`~~`kind`~~`readLink` annotation may also be written if `odata.metadata=minimal` is ~~used~~specified in order to ~~differentiate the kind of entry represented by the JSON object according to the table below. Where the object represents~~signal that an individual entity~~, or entity reference, the~~ is read-only.

For collections:

- The `odata.`~~`kind`~~`readLink` annotation~~ is optional and generally not included~~, if written, MUST be the request URL that produced the collection.

~~**`odata.kind` value**~~	~~**Entry type**~~
~~`Entry`~~	~~The JSON object represents an entity or entity reference.~~
~~`linkEntry`~~	~~The JSON object represents a link.~~
~~`deletedEntry`~~	~~The JSON object represents a deleted entity.~~
~~`deletedLink`~~	~~The JSON object represents a deleted link.~~

- The `odata.editLink` annotation MUST NOT be written as its meaning in this context is reserved for future versions of this specification.

~~4.5.10~~4.5.9 Annotation `odata.etag`

The `odata.etag` annotation MAY be applied to an ~~entity~~entity. 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]**~~]~~.

~~4.5.11~~4.5.10 Annotation `odata.navigationLink` and `odata.associationLink`

The ~~`odata.navigationLink`~~`odata.navigationLink` annotation 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 `odata.associationLink`: annotation contains a~~The *default computed value of a navigation URL* is the value of the read URL appended with a segment containing the name of the navigation property. The service MAY omit the `odata.navigationLink` annotation if `odata.metadata=minimal` has been specified on the request and the navigation link matches this computed value.

The `odata.associationLink` annotation 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 `odata.associationLink` annotation if `odata.metadata=minimal` has been specified on the request and the association link matches this computed value.

~~4.5.12~~4.5.11 Annotation `odata.media*`

~~For media entities and named stream properties~~For media entities and stream properties that don't follow standard URL conventions as defined in ~~**[OData-URL]**,~~**[OData-URL]**, at least one of the annotations `odata.mediaEditLink` and `odata.mediaReadLink` MUST be included.

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

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

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

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

Example~~:~~ 7:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Employees~~/@Element~~/$entity",
  "odata.mediaReadLink": "Employees(1)/$value",
  "odata.mediaContentType": "image/jpeg",
  "EmployeeID": 1,
  ...
}
```

5 Service Document

A service document in JSON is represented as a single JSON object with at least two properties; `odata.`~~`metadata`~~`context` and `odata.value`.

The value of the `odata.`~~`metadata`~~`context` property 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 ~~named entity~~singleton exposed by the service, see **~~[OData-CSDL].~~[OData-CSDL]**.

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 ~~named entity~~singleton, and one with name `url` containing the URL of the entity set, which may be absolute or relative to the ~~metadata~~context URL. 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 ~~or action~~ import MUST contain ~~this~~the `kind` name/value pair with a value of `FunctionImport` ~~or~~ ~~`ActionImport`~~, respectively.

JSON objects representing a ~~named entity~~singleton MUST contain ~~this~~the `kind` name/value pair with a value of ~~`Entity`~~`Singleton`.

JSON objects representing a related service document MUST contain ~~this~~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.~~

Example~~:~~ 8:

```
{
  "odata.~~metadata~~context": "http://host/service/$metadata",
  "value": [
    {
      "name": "Orders",
      "kind": "EntitySet",
      "url":  "Orders"
    },
    {
      "name": ~~"OrderDetails~~ "OrderItems",
      "title": "Order Details",
      "url": ~~"OrderDetails~~  "OrderItems"
    },
    {
      "name":  "TopProducts",
      "title": "Best-Selling Products",
      "kind":  "FunctionImport",
~~      "kind": "FunctionImport",~~
      "url": ~~"TopOrders~~  "TopProducts"
    },
    {
      "name":  "Contoso",
      "title": "Contoso Ltd.",
      "kind": ~~"Entity~~ "Singleton",
      "url":   "Contoso"
    },
    {
      "name": "Human Resources",
      "kind": "ServiceDocument",
      "url":  "http://host/HR/"
    }
  ]
}
```

6 Entity

An entity ~~MUST be~~is serialized as a JSON object.

Each ~~property~~property to be transmitted ~~MUST be~~is represented as a name/value pair within the object. The order properties appear within the object ~~MUST be~~is considered insignificant.

An entity in a payload ~~MAY~~may be a complete entity, a projected entity (see ~~**[OData-Protocol, section 10.2.3.2 The $select** *System Query Option*]),~~ `$select` **[OData-Protocol]**), or a partial entity update (see ~~**[OData-Protocol, section 10.3.1.2. Differential**~~ *Update*]~~). A complete entity MUST transmit every property, with the exception of unexpanded navigation properties. A projected entity MUST transmit the requested properties and MAY transmit other properties. A partial entity MUST transmit the properties that it intends to change; it MUST NOT transmit any other properties.~~ *an Entity* in **[OData-Protocol]**).

An entity representation can be (modified and) round-tripped to the ~~server~~service directly. The ~~metadata~~context URL ~~can but does not have to be removed; the server will ignore it if it is present~~is used in requests only as a base for relative URLs.

Example~~:~~

```
~~{~~
```

~~"~~ *9: entity with* `odata.metadata=minimal`

```
{
  "odata.context": "http://host/service/$metadata#Customers~~/@Element~~/$entity",
  "~~CustomerID~~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 10: entity with `odata.metadata=full`

```
{
  "odata.context": "http://host/service/$metadata#Customers/$entity",
  "odata.id": "Customers('ALFKI')",
  "odata.etag": "W/\"MjAxMy0wNS0yN1QxMTo1OFo=\"",
  "odata.editLink": "Customers('ALFKI')",
  "Orders@odata.navigationLink": "Customers('ALFKI')/Orders",
  "Orders@odata.associationLink": "Customers('ALFKI')/Orders/$ref",
  "ID": "ALFKI",
  "CompanyName": "Alfreds Futterkiste",
  "ContactName": "Maria Anders",
  "ContactTitle": "Sales Representative",
  "Phone": "030-0074321",
  "Fax": "030-0076545",
  "Address": {
    "Street": "Obere Str. 57",
    "City": "Berlin",
    "Region": null,
    "PostalCode": "D-12209",
    "Country~~": "Germany"~~@odata.navigationLink":
"Customers('ALFKI')/Address/Country",
  ~~}~~
~~}~~
    "Country@odata.associationLink":"Customers('ALFKI')/Address/Country/$ref",
  }
}
```

7 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~~collection of primitive values, or a ~~collection of complex values~~collection of complex values.

7.1 Primitive Value

Primitive values are represented following the rules of **[RFC4627]**.

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.

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` as well as enumeration values are represented as JSON strings whose content satisfies the rules `binaryValue`, `dateValue`, `dateTimeOffsetValue`, `durationValue`, `guidValue`, `timeOfDayValue`, and `enumValue`, respectively, in ~~[~~**[OData-ABNF]**~~]~~.

Geography and geometry values are represented 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.

Example~~:~~ 11:

```
{
  "NullValue": null,
  "TrueValue": true,
  "FalseValue": false,
  "IntegerValue": -128,
  "DoubleValue": 3.1415926535897931,
  "SingleValue": "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]}
}
```

7.2 Complex Value

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

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

Example~~:~~ 12:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  ...
  "Address": {
    "Street": "Obere Str. 57",
    "City": "Berlin",
    "Region": null,
    "PostalCode": "D-12209",
~~    "Country": "Germany"~~
  }
}
```

7.3 Collection of Primitive Values

~~7.3 Collection of Primitive Values~~

A collection of primitive values ~~MUST be~~is represented as a JSON array~~, and~~; each element in the array ~~MUST be~~is the representation of a ~~primitive value~~primitive value. An empty collection is represented as an empty array.

Example~~:~~ 13:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  ...
  "EmailAddresses": [
    "Julie@Swansworth.com",
    "Julie.Swansworth@work.com"
  ]
}
```

7.4 Collection of Complex Values

A collection of complex values ~~MUST be~~is represented as a JSON array~~, and~~; each element in the array ~~MUST be~~is the representation of a ~~complex value~~complex value. An empty collection is represented as an empty array.

Example~~:~~ 14:

```
{
  "PhoneNumbers": [
    {

  ~~      <data:~~     "Number~~>~~": "425-555-1212~~</data:Number>~~",
      ~~<data:PhoneType>~~"Type":  "Home~~</data:PhoneType>~~"
    },
    {
~~  <metadata:element metadata:~~        "odata.type~~="Person~~":
"#Model.CellPhoneNumber~~">  ~~",
    ~~<data:~~   "Number~~>~~":  "425-555-0178~~</data:Number>  ~~",
    ~~<data:PhoneType>~~   "Type": "Cell~~</data:PhoneType>  ~~",
    ~~<data:CellCarrier>~~   "Carrier":  "Sprint~~</data:CellCarrier>  ~~"
    }
  ]
}
```

8 Navigation Property

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

8.1 Navigation Link

The navigation link for a navigation property is represented as a name/value pair. The name ~~MUST be~~is the name of the property, followed by `@odata.navigationLink`. The value ~~MUST be~~is a URL that allows retrieving the related entity or collection of entities. It MAY be relative to the `odata.`~~`metadata`~~`context` URL.

The navigation link for a navigation property is only represented if the client requests `odata.metadata=full` or ~~if the client explicitly selects the navigation property in `$select`~~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~~:~~ 15:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  ...
  "Orders@odata.navigationLink~~":"~~": "Customers('ALFKI')/Orders",
  ...
}
```

8.2 Association Link

The association link for a navigation property is represented as a name/value pair. The name ~~MUST be~~is the name of the property, followed by `@odata.associationLink`. The value ~~MUST be~~is a URL that can be used to retrieve the reference or collection of references to the related entity or entities. It MAY be relative to the `odata.`~~`metadata`~~`context` URL.

The association link for a navigation property is only represented if the client requests `odata.metadata=full` or ~~if the client explicitly selects~~the association link cannot be computed by appending `/$ref` to the navigation ~~property in `$select`~~link. If it is represented, it MUST immediately precede the navigation link if the latter is represented.

Example~~:~~ 16:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  ...
  "Orders@odata.associationLink~~":"~~":  "Customers('ALFKI')/Orders/$ref",
  ...
}
```

8.3 Expanded Navigation Property

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

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

If a collection of entities can be related, it ~~MUST be~~is represented as a JSON array. Each element ~~MUST be~~is the ~~representation of an entity~~representation of an entity or the ~~representation of an entity~~

~~reference~~representation of an entity reference. An empty ~~set~~collection of entities (one that contains no entities) ~~MUST be~~is represented as an empty JSON array. The navigation property MAY be annotated with ~~`odata.count` or `odata.nextlink`~~`odata.context`, `odata.count` or `odata.nextlink`.

Example~~:~~ 17:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Customers~~/@Element~~/$entity",
  ...
  "Orders@odata.count": "42",
  "Orders": [ ... ],
  "Orders@odata.nextLink": "...",
  ...
~~  "CustomerDemographics": { ... },~~
~~  ...~~
}
```

8.4 Deep ~~Inserts~~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 ~~for~~ 18: inserting a new order ~~with order details and~~ for a new customer with order items related to existing products:

```
{
  "Customer": {
    "~~CustomerID~~ID": "ANEWONE",
    ...
  },
  ~~"Details": [~~
  ~~  {~~
~~      "ProductID": 28,~~
~~      "ProductID": 28,~~
~~      ...~~
~~    },~~
~~    {~~
~~      "ProductID": 39,~~
~~      "ProductID": 39,~~
~~      ...~~
~~    }~~
~~  ],~~
  ~~  "OrderID"~~"Items": [
    {
      "Product@odata.bind": "Products(28)",
      ...
    },
    {
      "Product@odata.bind": "Products(39)",
      ...
    }
  ],
  "ID": 11643,
~~  "CustomerID": "ANEWONE",~~
~~  "EmployeeID": 6,~~
  ...
}
```

8.5 Bind ~~Operations~~Operation

When inserting or updating an entity, relationships of navigation properties MAY be inserted or updated via bind operations. A bind operation is encoded as a property annotation `odata.bind` on the navigation property it belongs to and has a single value for singleton navigation properties or an array of values for collection navigation properties.

The values ~~MUST be~~are the ~~ids~~ids of the related entities. They MAY be ~~relative URLs~~relative URLs.

For insert operations collection navigation property bind operations and deep insert operations ~~MAY~~can be combined. In this case, 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.

Example ~~for assigning~~19: assign an existing product to an existing category with a partial update request~~:~~

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

{
  "Category@odata.bind": "Categories(6)"
  "Category@odata.bind": "Categories(6)"
}
```

9 Stream Property

An entity ~~MAY~~can have one or more ~~named~~ stream properties. The actual stream data is not contained in the entity. Instead stream property data is read and edited via URLs. The value for a ~~named~~ stream property contains the URLs for reading and editing the stream data along with other metadata for the stream.

The value of a ~~named~~ stream property is represented as a set of ~~`odata.media*` annotations like in the following example~~`odata.media*` annotations.

Example 20:

```
{
  "odata.~~metadata~~context":
"http://~~server/HR~~host/service/$metadata#~~Employees/@Element~~Products/$entity",
  ...
  "Thumbnail@odata.mediaReadLink": “http://server/Thumbnail546.jpg",
  "Thumbnail@odata.mediaEditLink": “http://server/uploads/Thumbnail546.jpg",
  "Thumbnail@odata.mediaContentType": "image/jpeg",
  "Thumbnail@odata.mediaEtag": ~~"####",~~"W/\"####\"",
  ...
}
```

10 Media Entity

Media entities are entities that describe a media resource, for example a photo. They are represented as ~~other~~ entities ~~and in addition~~that contain ~~`odata.media*` annotations.~~additional `odata.media*` annotations.

Example~~:~~ 21:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Employees~~/@Element~~/$entity",
  "odata.mediaReadLink": "Employees(1)/$value",
  "odata.mediaContentType": "image/jpeg",
~~"EmployeeID~~  "ID": 1,
  ...
}
```

11 Individual Property

~~If the message represents an~~ An individual property~~, it~~ is represented as a JSON object~~,~~.

A single-valued property that has the `null` value does not have a representation; see **[OData-Protocol]**.

A property 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 that is of complex type is represented as a complex value.

A property that is of a collection type is represented ~~similarly~~ 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 or primitive values~~representation of a collection of complex type values or collection of primitive values.

~~A property that is of complex type are represented as a complex value. A complex value is a JSON object, so no "wrapper object" is required.~~

~~Examples:~~

```
~~{~~
```

Example 22: primitive value

```
{
  "odata.~~metadata~~context": "http://host/service/$metadata#Edm.String",
  "value": "Pilar Ackerman"
}
```

```
~~{~~
```

Example 23: collection of primitive values

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Collection(Edm.String)",
  "value": ["~~gazpacho", "tomato", "vegetarian~~small", "medium", "extra
large"]
}
```

```
~~{~~
```

Example 24: empty collection of primitive values

```
{
  "odata.context": "http://host/service/$metadata#Collection(Edm.String)",
  "value": []
}
```

Example 25: complex value

```
{
  "odata.context":
"http://host/service/$metadata#Model.~~BillingAddress~~Address",
  ~~"Line1~~  "Street": "12345 Grant Street",
  ~~"Line2": null,~~
  "City": "Taft",
  ~~"State~~  "Region": "Ohio",
  ~~"ZipCode": "~~  "PostalCode": "OH 98052~~"~~",
  "Country@odata.navigationLink": "Countries('US')"
}
```

Example 26: empty collection of complex values

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

Note ~~that in requests~~: the ~~metadata~~context URL is optional in requests.

12 ~~Collections~~Collection of Entities

A collection of entities ~~MUST be~~is represented as a JSON object~~. This object MUST contain a `value`~~ containing a name/value pair named `value`. It MAY contain ~~`odata.metadata`, `odata.count`~~`odata.context`, ~~`odata.nextLink`~~`odata.count`, `odata.nextLink`, or ~~`odata.deltaLink`~~`odata.deltaLink` annotations.

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

The `odata.count` name/value pair represents the ~~inlinecount.~~number of entities in the collection. If present, it MUST come before the `value` name/value pair.

The value of the `value` ~~MUST be~~name/value pair is a JSON array~~. Each~~ where each element ~~MUST be a correctly formatted~~is ~~representation of an entity~~representation of an entity or a ~~representation of an entity reference.~~representation of an entity reference. An empty collection ~~MUST be~~is represented as an empty JSON array.

~~The `odata.nextLink` annotation MUST be included if the response represents a partial response. If provided, it MUST be the last name/value pair in the response.~~

Functions or actions that are ~~bindable~~bound to this ~~set~~collection of entities are advertised in the “wrapper object” in the same way as ~~functions or actions are advertised in the~~ functions or actions are advertised in the object representing a single entity.

The `odata.nextLink` annotation MUST be included in a response that represents a partial result.

Example~~:~~ 27:

```
{
  "odata.metadatacontext": "...",
  "odata.count": 37,
  "value": [
    { ... },
    { ... },
    { ... }
  ],
  "odata.nextLink": "...?$skiptoken=342r89",
}
```

13 ~~Resource~~Entity Reference

~~A resource reference is a reference to an entity or a property of an entity. Resource references referring to an entity are called entity references.~~

An entity reference (see **[OData-Protocol]**) MAY take the place of an entity instance in a JSON payload, based on the client request.

~~A resource reference MUST be~~ It is serialized as a JSON object~~.~~

~~The first name/value pair of the JSON object MUST be named `odata.ref` and~~ that MUST contain the ~~id~~`id` of the referenced entity~~.~~ and MAY contain the `odata.type`.

~~The following example shows an~~A collection of entity references is represented as a 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 MUST contain an `odata.context` annotation and MAY contain `odata.count`, `odata.nextLink`, or `odata.deltaLink` annotations.

Example 28: entity reference to order 10643~~:~~

```
{
  "odata.~~ref":"~~context":  "http://host/service~~/~~/$metadata#$ref/$entity",
  "odata.id": "Orders(10643)"
}
```

Example 29: collection of entity references

```
{
  "odata.context": "http://host/service/$metadata#$ref",
  "value": [
    { "odata.id": "Orders(10643" },
    { "odata.id": "Orders(10759" }
  ]
}
```

14 Delta Response

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

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~~changed, or ~~deleted~~deleted entities, as well as ~~added~~added or ~~deleted~~deleted 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~~next link for the client to retrieve the next set of changes.

~~Changes are generally ordered by the service according to when the last change occurred to an entity, but MUST be ordered such that applying all changes across all pages, in order, to the initial set yields a consistent result.~~

The last page of a delta response SHOULD contain a ~~delta link~~delta link 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 an ~~inlinecount,~~`odata.count` annotation, the returned ~~count is the count of~~number MUST include all added, changed, or deleted entities~~. `$count` and `$inlinecount` returned from a delta link do not include~~ , as well as added or deleted links.

~~The following example shows the following ordered changes:~~

Example 30: 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*

```
{

"odata.~~metadata~~context":"http://~~DeltaService.svc~~host/service/$metadata#Custome
rs~~/@Delta~~/$delta ",
  "odata.count":5,
  "value":
  [
     {
        "odata.id":"~~http://DeltaService.svc/~~Customers('BOTTM')'",
        "ContactName":"Susan Halvenstern"
     },
     {
        "odata.~~kind" :
"deletedLinkEntry~~context":"$metadata#Customers/$deletedLink",
        "source":"~~http://DeltaService.svc/~~Customers(~~ALFKI~~'ALFKI')'",
        "relationship":"Orders",
        "target":"~~http://DeltaService.svc/~~Orders(10643~~)",~~)"
~~        "when":"2012-11-07T15:38"~~
     },
     {
        "odata.~~kind" : "linkEntry~~context":"$metadata#Orders/$link",
        "source":"~~http://DeltaService.svc/~~Customers('BOTTM')",
        "relationship":"Orders",
        "target":"~~http://DeltaService.svc/~~Orders(10645~~)",~~)"
~~        "when":"2012-11-07T15:38"~~
     },
      {
        "odata.~~type" : "Northwind.Order~~context":"$metadata#Orders/$entity",
        "odata.id":"~~http://DeltaService.svc/Orders(10645)",~~
~~        "odata.metadata":"#~~Orders~~",~~(10643)",
        "~~ShipName":"Bottom-Dollar Markets",~~ShippingAddress":{
        ~~"ShipAddress~~  "Street":"23 Tsawassen Blvd.",
        ~~"ShipRegion~~  "City":"Tsawassen"
          "Region":"BC",
        ~~"ShipPostalCode~~  "PostalCode":"T2F 8M4~~",~~"
        ~~"ShipCountry":"Canada"~~},
      },
      {

"odata.~~kind":"deletedEntry~~context":"$metadata#Customers/$deletedEntity",
        "id":"http://~~DeltaService.svc~~host/service/Customers('ANTON')",
~~        "when":"2012-11-07T15:38",~~
        "reason~~": "~~":"deleted"
      }
   ],
   "odata.deltaLink":

"http://~~DeltaService.svc~~host/service/Customers?$expand=~~orders~~Orders&$deltatoke
n=8015"
}
```

14.1 Added/Changed Entity

Added or changed entities within a delta response are represented as ~~entities.~~entities

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.

Entities that are not part of the entity set specified by the ~~Metadata~~context URL MUST include the ~~`odata.metadata` attribute~~`odata.context` annotation to specify the entity set of the entity.

Entities ~~MUST~~ include annotations for selected navigation links based on `odata.metadata` but MUST NOT include expanded navigation properties inline.

14.2 Deleted Entity

Deleted ~~Entries~~entities in JSON are returned as deleted-~~entry~~entity objects. Delta responses MUST contain a deleted-~~entry~~ entity object for each deleted entity.

The deleted-~~entry~~entity object has the following properties:

~~`odata.kind` – The `odata.kind` property MUST be the first property and MUST be `"deletedEntry"`~~

- ~~`id` – The id~~`odata.context` – the context URL fragment MUST be `#{entity-set}/$deletedEntity`, where `{entity-set}` is the entity set of the deleted entity
- `id` – The id of the deleted entity (same as the ~~`odata.id`~~`odata.id` returned or computed when calling GET on resource~~)~~),
- `reason` – An optional string value; 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).

~~`when` – An optional Datetime value indicating when the element was deleted.~~

14.3 Added Link

Links within a delta response are represented as link~~-entry~~ objects.

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

The link~~-entry~~ object ~~has~~MUST include the following properties:

~~`odata.kind` – The `odata.kind` property MUST be the first property and MUST be `"linkEntry"`~~

- `odata.context` – the context URL fragment MUST be `#{entity-set}/$link`, where `{entity-set}` is the entity set containing the source ~~– The id~~entity
- `source` – The id of the entity from which the relationship is defined, which may be absolute or relative to the context URL
- `relationship` ~~– the~~– The name of the relationship property on the parent object

`target` – The ~~id~~id of the related entity

~~`when` – An optional datetime value indicating when the link was created~~

- ~~The link-entry MUST contain a `source` property specifying the `odata.id` of the entity from which the link exists, a `relationship` property specifying the name of the navigation property for~~, which ~~the link was specified, and a `target` attribute containing the `odata.id` of the related resource. The link-entry MAY include an optional `when` attribute specifying when the link was created.~~may be absolute or relative to the context URL

14.4 Deleted Link

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

Delta responses MUST contain a deleted-link~~-entry~~ 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-entry~~link object that specifies the same `source` and `relationship`.

The deleted-link~~-entry has~~ object MUST include the following properties:

~~`odata.kind` – The `odata.kind` property MUST be the first property and MUST be `"deletedLinkEntry"`~~

- ~~`source` – The id~~`odata.context` – the context URL fragment MUST be `#{entity-set}/$deletedLink`, where `{entity-set}` is the entity set containing the source entity
- `source` – The id of the entity from which the relationship is defined, which may be absolute or relative to the context URL
- `relationship` ~~– the~~– The name of the relationship property on the parent object

`target` – The ~~id~~id of the related entity

~~`when` – An optional datetime value indicating when the link was created~~

- ~~The deleted-link-entry MUST contain a `source` property specifying the `odata.id` of the entity from which the link was deleted, a `relationship` property specifying the name of the navigation property for~~, which ~~the link was specified, and a `target` attribute containing the `odata.id` of the related resource. The deleted-link-entry MAY include an optional `when` attribute specifying when the link was created.~~may be absolute or relative to the context URL

15 Bound Function

A function that is ~~bindable~~bound to the current entity is advertised via a name/value pair~~. The~~ where the name ~~MUST be~~is a hash (`#`) character followed by the namespace- or alias-qualified name of the function.~~ The value MUST be a JSON object.~~

Functions that are ~~bindable~~bound to a ~~set~~collection of entities are advertised in representations of ~~a~~that collection~~ of entities~~.

If function overloads exist that cannot be bound to the current entity type, the name SHOULD address a specific function overload by appending ~~a~~the parentheses-enclosed, comma-separated list of non-binding parameter names, ~~each name followed by a colon and its namespace- or alias-qualified type, ~~see rule `qualifiedFunctionName` in ~~[~~**[OData-ABNF]**~~]~~.

If ~~`odata.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~~annotations. The order of the name/value pairs MUST be considered insignificant.

The `target` name/value pair ~~MUST contain~~contains a bound function or action URL.

The `title` name/value pair ~~MUST contain~~contains the function or action title as a string.

~~If `odata.metadata=minimal`~~If `odata.metadata=minimal` is requested, the `target` name/value pair MUST be included if its value differs from the canonical function or action URL.

Example ~~for a~~31: minimal representation of a function where all overloads are applicable~~:~~

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#Employees~~/@Element~~/$entity",
  "#Model.RemainingVacation": {},
  ...
}
```

Example ~~for a~~32: full representation of a specific overload

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#~~LeaveRequests/@Element~~Employees/$entity",
  ~~...~~
  "#Model.~~ ~~RemainingVacation(Year)": {
    "title": "Remaining vacation from year~~:Edm.Int32)": {~~...",
~~    "title": "Remaining Vacation",~~
    "target": "Employees(2)/RemainingVacation"
  },
  ...
}
```

Example 33: full representation in a collection

```
{
  "odata.context": "http://host/service/$metadata#Employees",
  "#Model.RemainingVacation": {
    "title": "Remaining Vacation",
    "target": "Managers(22)/Employees/RemainingVacation"
  },
  "value": [ ... ]
}
```

16 Bound Action

An action that is ~~bindable~~bound to the current entity is advertised via a name/value pair~~. The~~ where the name ~~MUST be~~is a hash (`#`) character followed by the namespace- or alias-qualified name of the action. ~~The value MUST be a JSON object.~~

Actions that are ~~bindable~~bound to a ~~set~~collection of entities are advertised in representations of ~~a~~that collection~~ of entities~~.

If ~~`odata.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~~annotations. The order of these name/value pairs MUST be considered insignificant.

The `target` name/value pair ~~MUST contain~~contains a bound function or action URL.

The `title` name/value pair ~~MUST contain~~contains the function or action title as a string.

~~If `odata.metadata=minimal`~~If `odata.metadata=minimal` is requested, the `target` name/value pair MUST be included if its value differs from the canonical function or action URL.

~~Examples:~~

Example 34: minimal representation in an entity

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#LeaveRequests~~/@Element~~/$entity",
  "#Model.Approval": {},
  ...
}
```

Example 35: full representation in an entity:

```
{
  "odata.~~metadata~~context":
"http://host/service/$metadata#LeaveRequests~~/@Element~~/$entity",
  ~~...~~
  "#Model.Approval": {
    "title": "Approve Leave Request",
    "target": "LeaveRequests(2)/Approval"
  },
  ...
}
```

Example 36: full representation in a collection

```
{
  "odata.context": "http://host/service/$metadata#LeaveRequests",
  "#Model.Approval": {
    "title": "Approve All Leave Requests",
    "target": "Managers(22)/Inbox/Approval"
  },
  "value": [ ... ]
}
```

17 Action ~~Parameters~~Invocation

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

Each non-binding parameter value ~~specified MUST be~~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 ~~MUST be~~are assumed to have the ~~default~~`null` value ~~specified in the service metadata, see **[OData-CSDL]**.~~.

Example~~:~~ 37:

```
{
  "param1": 42,
  "param2": {
    "Street": "One Microsoft Way",
    "Zip": 98052
  },
  "param3": [ 1, 42, 99 ],
  "param4": null
}
```

18 Instance Annotations

Annotations are an extensibility mechanism that allows ~~servers~~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 ~~easily identifiable as~~ name/value pairs that have a dot (`.`) as part of the name. All annotations that start with `odata` are reserved for future extensions of the protocol and format. Custom annotations are annotations that have a non-empty prefix that is different from `odata`.

Annotations ~~MAY~~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-CSDL]).~~**[OData-CSDL]**).

~~The following example shows annotations applied to many different constructs.~~

Example 38:

```
{
  "odata.~~metadata~~context": "http://host/service/$metadata#Customers",
  "com.contoso.customer.setkind~~" :~~": "VIPs",
  "value": [
    {
~~      "com.contoso.customer.kind": "VIP",~~
      "com.contoso.display.~~order": 1~~highlight": true,
      "~~CustomerID~~ID": "ALFKI",
      "CompanyName@com.contoso.display~~" :~~.style": { "title~~" :~~": true, "order~~"~~
~~:~~": 1 },
      "CompanyName": "Alfreds Futterkiste",
      "Orders@com.contoso.~~purchaseorder.priority" : 1~~display.style": {
"order": 2 }
    }
  ]
}
```

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.

18.1 Annotate a JSON Object

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

The name ~~MUST be~~is the namespace- or alias-qualified name of the annotation~~. This name MUST include~~, i.e. the namespace ~~and name, separated~~or alias of the schema that defines the term, followed by a ~~period (.).~~dot (`.`), followed by the name of the term. The namespace or alias MUST be defined in the metadata document, see **[OData-CSDL]**.

The value MUST be ~~the~~an appropriate value for the annotation.

18.2 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 placed next to the annotated name/value pair and ~~MUST be~~ represented as a single name/value pair.

The name ~~MUST be~~is the same as the name of the name/value pair being annotated, followed by the “at” sign (`@`), followed by the namespace- or alias-qualified name of the annotation~~. This name MUST include namespace and name, separated by a period (.).~~, followed by a dot (`.`), followed by the name of the term. The namespace or alias MUST be defined in the metadata document, see **[OData-CSDL]**.

The value MUST be ~~the~~an appropriate value for the annotation.

19 Error Response

The error response MUST be a single JSON object. This object MUST have a single name/value pair~~.~~ ~~The name MUST be~~ 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 ~~MUST be~~is a language-independent string. Its value ~~MUST be~~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]** 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~~annotations in any of its JSON objects.

Example~~:~~ 39:

```
{
  "error": {
    "code": "501",
    "message": "~~Not supported Functionality~~Unsupported functionality",
    "target": "query",
    "details": [
      {
       "code": "301",
       "target": "$search"
       "message": "$search query option not supported",
      }
    ]
    "innererror": {
      "trace": [...],
      "context": {...}
    }
  }
}
```

20 Extensibility

Implementations ~~MAY add custom content anywhere allowed by **[RFC4287]**, Section 6, “Extending Atom”, and **[RFC5023]**, Section 6.2 “Document Extensibility”. However, custom elements and attributes MUST NOT be defined in the OData Data Namespace nor the OData Metadata Namespace, and SHOULD not~~can add custom 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 ~~foreign markup according to **[RFC4287]**.~~annotations not defined in this version of the OData JSON Specification.

21 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 **[RFC4627]** as starting point.

~~21~~22 Conformance

~~A conforming OData service~~ Conforming clients MUST ~~comply with one~~be prepared to consume a service that uses any or all of the ~~conformance levels~~constructs defined in ~~**[OData-Protocol].**~~

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 conform to the OData JSON format, a service:~~

- ~~MUST support the `application/json` media type in the `Accept` header~~
- ~~SHOULD support the `$format` system query option~~
- ~~MAY support the `odata.streaming=true` parameter in the `Accept` header~~

~~•9. MUST return well-formed JSON payloads~~

- ~~MUST support `odata.metadata=full`~~
- ~~MAY return full metadata regardless of `odata.metadata`~~
- ~~MUST include the `odata.nextLink` annotation in the feed for partial results~~

~~•14. MUST NOT violate any other aspects of this OData JSON specification~~

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

1. MUST either:
   a. understand `odata.metadata=minimal` ~~OR~~ (section 3.1.1) or
   ~~•~~b. explicitly specify `odata.metadata=none` (section 3.1.3) or `odata.metadata=full` (section 3.1.2) in the request (client)

~~•~~2. MUST be prepared to consume a response with full metadata

   - ~~MUST NOT require `odata.streaming=true` in the `Content-Type` header~~

~~•~~3. MUST be prepared to receive all data types (section 7.1)
   ~~o~~a. defined in this specification (client)
   ~~o~~b. exposed by the service (service)

   - ~~MUST support entities returned as entity references~~

~~•~~4. MUST interpret all `odata` annotations defined according to the ~~version~~`OData-Version` header of the ~~response~~payload (section 4.5)
~~•~~5. MUST be prepared to receive any annotations, including ~~custom annotations~~custom annotations and `odata` annotations not defined in the `OData`~~`version`~~`-Version` header of the ~~response~~payload (section 20)
6. MUST NOT require `odata.streaming=true` in the `Content-Type` header (section 4.4)

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

7. MUST comply with one of the conformance levels defined in **[OData-Protocol]**
8. MUST support the `application/json` media type in the `Accept` header (section 3)
9. MUST return well-formed JSON payloads
10. MUST support `odata.metadata=full` (section 3.1.2)
11. MUST include the `odata.nextLink` annotation in partial results for entity collections (section 4.5.5)
12. MUST support entity instances with external metadata (section 4.5.1)
13. MUST support properties with externally defined data types (section 4.5.3)
14. MUST NOT violate any other aspects of this OData JSON specification
15. SHOULD support the `$format` system query option (section 3)
16. MAY support the `odata.streaming=true` parameter in the `Accept` header (section 4.4)
17. MAY return full metadata regardless of `odata.metadata` (section 3.1.2)

Appendix A. Acknowledgments

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

Appendix B. Revision History

Revision	Date	Editor	Changes Made
Working Draft 01	2012-08-22	Michael Pizzo	Translated Contribution to OASIS format/template
Working Draft 01.1	2013-1-31	Ralf Handl	Adopted new, more concise JSON format
Committee Specification Draft 01	2013-04-26	Ralf Handl Michael Pizzo	Expanded error information Added enumerations Fleshed out descriptions and examples and addressed numerous editorial and technical issues through processed through the TC Added Conformance section
Committee Specification Draft 02	2013-07-01	Ralf Handl Michael Pizzo	Improved rules for odata.id, odata.editLink, and odata.readLink Improved action/function advertisement Improved entity references Improved rules for relative URLs Simplified delta responses GeoJSON for Geo types Improved description of primitive value representation Improved examples, aligned with Atom format specification Aligned terms across specifications