OData Atom Format Version 4.0

Committee Specification 02

17 November 2013

Specification URIs

This version:

http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs02/odata-atom-format-v4.0-cs02.doc (Authoritative)
http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs02/odata-atom-format-v4.0-cs02.html
http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs02/odata-atom-format-v4.0-cs02.pdf

Previous version:

http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs01/odata-atom-format-v4.0-cs01.doc (Authoritative)
http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs01/odata-atom-format-v4.0-cs01.html
http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs01/odata-atom-format-v4.0-cs01.pdf

Latest version:

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

Martin Zurmuehl (martin.zurmuehl@sap.com), SAP AG
Michael Pizzo (mikep@microsoft.com), Microsoft
Ralf Handl (ralf.handl@sap.com), SAP AG

Additional artifacts:

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

- XML schema: http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs02/schemas/

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
  - *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.* 17 November 2013. http://docs.oasis-open.org/odata/odata/v4.0/cs02/abnf/

    - Vocabulary components: *OData Core Vocabulary, OData Measures Vocabulary* and *OData Capabilities Vocabulary*. 17 November 2013. http://docs.oasis-open.org/odata/odata/v4.0/cs02/vocabularies/
  - *OData JSON Format Version 4.0*. Latest version. http://docs.oasis-open.org/odata/odata-json-format/v4.0/odata-json-format-v4.0.html

Declared XML namespaces:

- http://docs.oasis-open.org/odata/ns/data
- http://docs.oasis-open.org/odata/ns/metadata

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 4.0 Part 1: Protocol. This document extends the core specification by defining representations for OData requests and responses using an Atom 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/.

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).

Citation format:

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

[OData-Atom-Format-v4.0]

OData Atom Format Version 4.0. 17 November 2013. OASIS Committee Specification 02. http://docs.oasis-open.org/odata/odata-atom-format/v4.0/cs02/odata-atom-format-v4.0-cs02.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]. The OData Atom Format specification extends the former by defining representations for OData requests and responses using an Atom format.

An OData payload may represent:

- a service document describing the top-level resources exposed by the service
- a single entity (a structured type with an identity)
- a resource reference
- a collection of entities
- a single primitive or complex type value
- a collection of primitive or complex type values
- a media resource
- a collection of changes
- a single link to a related entity
- a collection of links to related entities
- an error document
- an xml document describing the entity model exposed by the service
- a batch of requests to be executed in a single request
- a set of responses returned from a batch request

For a description of the xml format for describing an entity model, see **[OData-CSDL]**. For a description of batch requests and responses, see **[OData-Protocol]** .

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]**.

1.2 Normative References

This document references the following related documents:

[GML] Portele, C., Ed., "OpenGIS Geography Markup Language (GML) Encoding", August 2007. http://portal.opengeospatial.org/files/?artifact_id=20509**.**

[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-MetaXML] *OData Metadata XML Schema.*
See link in “Additional artifacts” section on cover page.

[OData-Protocol] *OData Version 4.0 Part1: 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.

[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.

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

[RFC4287] Nottingham, M., Ed., and R. Sayre, Ed. “The Atom Syndication Format”, RFC 4287, December 2005. http://www.ietf.org/rfc/rfc4287.txt.

[RFC5023] Gregorio, J., Ed., and B. de hOra, Ed., “The Atom Publishing Protocol”, RFC 5023, October 2007. http://www.ietf.org/rfc/rfc5023.txt.

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

[RFC6721] Snell, J., **"**The Atom 'deleted-entry' Element", RFC 6721, September 2012, http://tools.ietf.org/html/rfc6721.

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 Atom Format Design

The Atom Syndication Format **[RFC4287]** defines an XML-based format for describing feeds made up of individual entries. The Atom Publishing Protocol **[RFC5023]** defines an application-level protocol based on HTTP transfer of Atom-formatted representations.

OData builds on **[RFC4287]** and **[RFC5023]** by defining additional conventions and extensions for representing and querying entity data, e.g. OData collections are represented as Atom feeds, with one Atom entry for each entity within a collection.

As specified in **[RFC4287]** and **[RFC5023]** processors that encounter foreign markup MUST NOT stop processing and MUST NOT signal an error. This includes additional elements or attributes in any namespace, including elements and attributes in the OData Data and Metadata namespaces, e.g. values for properties not declared in `$metadata`, and annotation that are not defined in the version of the payload being returned.

2.1 Namespaces

OData defines meaning for elements and attributes defined in the following namespaces.

2.1.1 Atom Syndication

Atom elements and attributes are defined within the Atom namespace, see **[RFC4287]**:

```
http://www.w3.org/2005/Atom
```

In this specification the namespace prefix `atom` is used to represent the Atom Namespace, however the prefix name is not prescriptive.

2.1.2 Atom Publishing Protocol

Atom Publishing Protocol (AtomPub) elements and attributes are defined within the AtomPub namespace, see **[RFC5023]**:

```
http://www.w3.org/2007/app
```

In this specification the namespace prefix `app` is used to represent the AtomPub Namespace, however the prefix name is not prescriptive.

2.1.3 Atom Tombstone

The `deleted-entry` element is defined within the Atom Tombstone namespace, see **[RFC6721]**:

```
http://purl.org/atompub/tombstones/1.0
```

In this specification the namespace prefix `atom-tombstone` is used to represent the Atom Tombstone Namespace, however the prefix name is not prescriptive.

2.1.4 OData Data

Elements that describe the actual data values for an entity are qualified with the OData Data Namespace:

```
http://docs.oasis-open.org/odata/ns/data
```

In this specification the namespace prefix `data` is used to represent the OData Data Namespace, however the prefix name is not prescriptive.

2.1.5 OData Metadata

Attributes and elements that represent metadata (such as type, null usage, and entry-level etags) are defined within the OData Metadata Namespace:

```
http://docs.oasis-open.org/odata/ns/metadata
```

Custom elements or attributes MUST NOT use this namespace.

In this specification the namespace prefix `metadata` is used to represent the OData Metadata Namespace, however the prefix name is not prescriptive.

2.2 XML Schema Definition for OData Metadata

This specification contains a normative XML schema for the OData Metadata namespace, see **[OData-MetaXML]**.

It only defines the shape of well-formed OData metadata, but is not descriptive enough to define what correct OData metadata is. This specification document defines additional rules that correct OData metadata MUST fulfill. In case of doubt on what makes OData metadata correct the rules defined in this specification document take precedence.

3 Requesting the Atom Format

The OData Atom format MAY be requested using the `$format` query option in the request URL with the MIME type `application/atom+xml`, or the case-insensitive abbreviation `atom` which MUST NOT be followed by format parameters.

Alternatively, this format MAY be requested using the `Accept` header with the MIME type `application/atom+xml`.

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

The service document MAY additionally be requested with the more specific MIME type `application/atomsvc+xml` using either `$format` or `Accept`.

All resources MAY additionally be requested with the less specific MIME type `application/xml` using either `$format` or `Accept`, or the case-insensitive abbreviation `xml` using `$format`.

Services SHOULD advertise the supported MIME types by annotating their entity container with the term `Capabilities.SupportedFormats` defined in **[OData-VocCap]**.

4 Common Characteristics

4.1 Header Content-Type

The Content-Type header for Atom responses MUST use the most specific MIME type for the requested resource that is indicated as acceptable by the client.

Requests using the $format query option with the abbreviation atom MUST receive the MIME type

- application/atomsvc+xml for the service document,
- application/atom+xml for entities and collections of entities, references, or changes,
- application/xml for all other resources.

Requests using $format or an Accept header with value application/atom+xml MUST receive the MIME type

- application/xml for the service document,
- application/atom+xml for entities and collections of entities, references, or changes,
- application/xml for all other resources.

Requests using $format or an Accept header with value application/xml or $format with the abbreviation xml MUST receive the MIME type application/xml for all resources.

Data modification requests for entities or collections of entities MUST specify a Content-Type header with a value of either application/atom+xml or application/xml. Data modification requests for all other resources MUST specify a Content-Type header with a value of application/xml.

4.2 Message Body

Each message body MUST be represented as an XML document with a single root element. This element is either the representation of an entity, an entity reference, a primitive value, a complex type instance, a collection of primitive values, a collection of complex values, a collection of entities, or a collection of items that represent changes to a previous result.

Client libraries MUST retain the order of XML elements in document order for ATOM and XML responses.

OData does not impose any ordering constraints on XML attributes within XML elements.

4.3 Relative URLs

OData payloads MAY use relative references as defined in **[RFC3986]** by specifying the xml:base attribute to define a base URL for relative references defined within the scope of the element containing the xml:base attribute.

If no xml:base attribute is present in the context of a relative reference, relative URLs are relative to the request URL. This also applies to relative URLs in the xml:base attribute.

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

5 Service Document

AtomPub defines the concept of a service document to represent the set of available collections. OData uses the service document to describe the entity sets, singletons, and parameterless function imports published by the service.

Example 2:

```
<app:service xmlns:app="http://www.w3.org/2007/app"
             xmlns:atom="http://www.w3.org/2005/Atom"
             xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
             xml:base="http://host/service/"
             metadata:context="$metadata"
             metadata:metadata-etag="W/&quot;MjAxMy0wNS0xM1QxNDo1NFo=&quot;">
  <app:workspace>
    <atom:title type="text">Data</atom:title>
    <app:collection href="Orders">
      <atom:title type="text">Orders</atom:title>
    </app:collection>
    <app:collection href="OrderItems">
      <atom:title type="text">Order Details</atom:title>
    </app:collection>
    <metadata:function-import href="TopProducts">
      <atom:title>Best-Selling Products</atom:title>
    </metadata:function-import>
    <metadata:singleton href="Contoso">
      <atom:title>Contoso Ltd.</atom:title>
    </metadata:singleton>
    <metadata:service-document href="http://host/HR/">
      <atom:title>Human Resources</atom:title>
    </metadata:service-document>
  </app:workspace>
</app:service>
```

5.1 Element `app:service`

The root of a service document is a single `app:service` element. The `app:service` element MUST contain exactly one `app:workspace` elements.

5.1.1 Attribute `metadata:context`

An `app:service` element MUST have a `metadata:context` attribute, defined in the OData Metadata namespace, whose value is the URL that returns the metadata document of the service.

For more information on the format of the metadata document, see **[OData-CSDL]**.

5.1.2 Attribute `metadata:metadata-etag`

An `app:service` element MAY have a `metadata:metadata-etag` attribute to specify an ETag that can be used to determine the current version of the service's metadata document.

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

5.2 Element app:workspace

OData represents the entity container of a service (see **[OData-CSDL])** as an app:workspace element. An app:workspace element contains zero or more app:collection elements, one for each entity set published by the container, zero or more metadata:function-import elements, one for each function import published by the container, zero or more metadata:singleton elements, one for each singleton published by the container, and zero or more metadata:service-document elements, one for each related service document.

As defined in **[RFC-5023],** the app:workspace element MUST contain an atom:title element containing the human-readable description of the workspace. This value may be different from the name of the entity container.

5.3 Element app:collection

OData represents entity sets that are *not* marked with IncludeInServiceDocument="false" (see **[OData-CSDL]**) as app:collection elements contained within the app:workspace element.

Example 3:

```
<app:collection href="OrderItems">
  <atom:title type="text">Order Details</atom:title>
</app:collection>
```

5.3.1 Attribute href

The app:collection element MUST contain an href attribute which represents a URL that can be used to retrieve the members of the entity set.

5.3.2 Attribute metadata:name

The metadata:name attribute MUST contain the name of the entity set.

It MAY be omitted if its value is identical the the value of the href attribute, which is the case if the service uses relative URLs following the OData URL conventions described in **[OData-URL]**.

5.3.3 Element atom:title

As defined in **[RFC-5023],** the app:collection element MUST contain an atom:title element. The atom:title element SHOULD contain a human-readable description of the entity set which MAY be the name of the entity set.

5.4 Element metadata:function-import

OData represents function imports that are marked with IncludeInServiceDocument="true" (see **[OData-CSDL]**) as metadata:function-import elements contained within the app:workspace element.

5.4.1 Attribute href

The metadata:function-import element MUST contain an href attribute which represents a URL that can be used to retrieve the function import result.

5.4.2 Attribute metadata:name

The metadata:name attribute MUST contain the name of the function import.

It MAY be omitted if its value is identical the the value of the href attribute, which is the case if the service uses relative URLs following the OData URL conventions described in **[OData-URL]**.

5.4.3 Element atom:title

The metadata:function-import element MUST contain an atom:title element. The atom:title element SHOULD contain a human-readable description of the function import which MAY be the name of the function import.

5.5 Element metadata:singleton

OData represents singletons as metadata:singleton elements contained within the app:workspace element.

5.5.1 Attribute href

The metadata:singleton element MUST contain an href attribute which represents a URL that can be used to retrieve the singleton.

5.5.2 Attribute metadata:name

If the href attribute of a metadata:singleton element contains a relative URL that follows the conventions described in **[OData-URL]**, the metadata:name attribute can be ommited. Otherwise the metadata:name attribute MUST be specified and MUST contain the name of the singleton.

5.5.3 Element atom:title

The metadata:singleton element MUST contain an atom:title element. The atom:title element SHOULD contain a human-readable description of the singleton which MAY be the name of the singleton.

5.6 Element metadata:service-document

OData represents related service documents as metadata:service-document elements contained within the app:workspace element.

Example 4:

```
<metadata:service-document href="http://host/HR/">
  <atom:title>Human Resources</atom:title>
</metadata:service-document>
```

5.6.1 Attribute href

The metadata:service-document element MUST contain an href attribute which represents a URL that can be used to retrieve the related service document.

5.6.2 Element atom:title

The metadata:service-document element MUST contain an atom:title element. The atom:title element SHOULD contain a human-readable description of the related service document.

6 Entity

Entities, whether individual or within an Atom feed, are represented as `atom:entry` elements.

Example 5:

```
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
       xmlns:data="http://docs.oasis-open.org/odata/ns/data"
       xml:base="http://host/service/"
       metadata:context="$metadata#Customers/$entity"
       metadata:metadata-etag="W/&quot;MjAxMy0wNS0xM1QxNDo1NFo=&quot;">
  <id>http://host/service/$metadata#Customers('ALFKI')</id>
  <title />
  <summary />
  <updated>2012-03-30T07:11:05Z</updated>
  <author>
    <name />
  </author>
  <link rel="edit" title="Customer" href="Customers('ALFKI')" />
  <link rel="http://docs.oasis-open.org/odata/ns/related/Orders"
        type="application/atom+xml;type=feed"
        title="Orders" href="Customers('ALFKI')/Orders" />
  <link rel="http://docs.oasis-open.org/odata/ns/related/Supplier"
        type="application/atom+xml;type=entry"
        title="Supplier" href="Customers('ALFKI')/Supplier" />
  <category term="#ODataDemo.Customer"
            scheme="http://docs.oasis-open.org/odata/ns/scheme" />
  <content type="application/xml">
    <metadata:properties>
      <data:ID>ALFKI</data:ID>
      <data:CompanyName>Alfreds Futterkiste</data:CompanyName>
      <data:ContactName>Maria Anders</data:ContactName>
      <data:ContactTitle>Sales Representative</data:ContactTitle>
      <data:Phone>030-0074321</data:Phone>
      <data:Fax>030-0076545</data:Fax>
      <data:Address>
        <data:Street>Obere Str. 57</data:Street>
        <data:City>Berlin</data:City>
        <data:Region metadata:null="true" />
        <data:PostalCode>D-12209</data:PostalCode>
        <link rel="http://docs.oasis-open.org/odata/ns/related/Country"
              type="application/atom+xml;type=entry"
              title="Country of residence"
              href="Customers('ALFKI')/Address/Country" />
      </data:Address>
    </metadata:properties>
  </content>
</entry>
```

This section defines the elements and attributes within an `atom:entry` element that are assigned meaning in OData.

6.1 Element atom:entry

An atom:entry element is used to represent a single OData entity, which is an instance of a structured type with an identity.

6.1.1 Attribute metadata:etag

The atom:entry element MAY contain a metadata:etag attribute, representing 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 to **[OData-Protocol]**.

6.1.2 Attribute metadata:context

If the root of the response is an atom:entry element, or the entity set cannot be determined from the context URL of the feed, the atom:entry element MUST have a metadata:context attribute, defined in the OData Metadata namespace, whose value is the context URL that describes the entity represented by the atom:entry.

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

6.1.3 Attribute metadata:metadata-etag

If the root of the response is an atom:entry element, it MAY have a metadata:metadata-etag attribute to specify an ETag that can be used to determine the current version of the service's metadata document.

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

6.2 Element atom:id

The atom:id element MUST contain the entity-id; see **[OData-Protocol]**. By convention the entity-id is identical to the canonical URL of the entity, as defined in **[OData-URL]**.

If the entity is transient (i.e. cannot be read or updated), the atom:id SHOULD follow the pattern odata:transient:{some-generated-unique-identifier-to-not-break-atom-parsers}.

Clients MAY assume that an entity with an atom:id that matches the transient pattern cannot be compared to other entities, reread, or updated.

6.3 Element atom:category

An OData entry MUST contain a single atom:category element with a scheme attribute equal to

```
http://docs.oasis-open.org/odata/ns/scheme
```

to identify the entity type of the entry.

An atom:category element describing an OData entity type MUST have a term attribute whose value is a URI indicating the type of the entity. The URI may be an absolute or relative URL containing the namespace-qualified or alias-qualified type name as a fragment, or may simply contain the qualified type name prefixed with hash (#). In the latter case, the type MUST be defined or referenced in the metadata document defined by the current context URL.

Example 6: entity of type Model.VipCustomer defined in the metadata document of the same service

```
<category rel="http://docs.oasis-open.org/odata/ns/scheme"
          term="#Model.VipCustomer"/>
```

Example 7: entity of type Model.VipCustomer defined in the metadata document of a different service

```
<category rel="http://docs.oasis-open.org/odata/ns/scheme"
```

```
term="http://host/alternate/$metadata#Model.VipCustomer"/>
```

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

The entry MAY contain additional `atom:category` elements with different scheme values; such `atom:category` elements have no semantic meaning in OData.

6.4 Element `atom:link`

Atom defines two types of links within an entry that represent retrieve or update/delete operations on the entry:

- `atom:link` elements with a `rel` attribute of `self` can be used to retrieve the entity (via the URL specified in the href attribute).
- `atom:link` elements with a `rel` attribute of `edit` can be used to retrieve, update, or delete the entity (via the URL specified in the href attribute).

An `atom:entry` element representing an OData entity MUST contain an edit link if and only if the entity is updatable. It MUST contain a self link if and only if the entity is read-only or the read link is different from the edit link. Transient entities contain neither a self link nor an edit link.

Clients MAY use the edit link to retrieve the entity if no self link is present. They SHOULD NOT attempt to update an entity that does not contain an edit link.

6.5 Element `atom:content`

The `atom:content` element contains the properties of the entity as a `metadata:properties` element unless the entity is a media entity.

7 Structural Property

7.1 Primitive Value

OData Atom and XML payloads represent values of primitive types following the rules of **[OData-ABNF]**.

Geography and Geometry values are represented as defined in **[GML]**.

Strings are represented according to the XML escaping rules for character data.

Values of the other primitive types are represented according to the appropriate alternative in the `primitiveValue` rule of **[OData-ABNF]**, i.e. `Edm.Binary` as `binaryValue`, `Edm.Boolean` as `booleanValue` etc.

Example 8:

```
<metadata:properties>
  <data:NullValue metadata:null="true"/>
  <data:TrueValue metadata:type="Boolean">true</data:TrueValue>
  <data:FalseValue metadata:type="Boolean">false</data:FalseValue>
  <data:BinaryValue metadata:type="Binary">T0RhdGE</data:BinaryValue
  <data:IntegerValue metadata:type="SByte">-128</data:IntegerValue>
  <data:DoubleValue metadata:type="Double"
       >3.1415926535897931</data:DoubleValue>
  <data:SingleValue metadata:type="Single">INF</data:SingleValue>
  <data:DecimalValue metadata:type="Decimal">34.95</data:DecimalValue>
  <data:StringValue>Say "Hello",
then go!</data:StringValue>
  <data:DateValue metadata:type="Date">2012-12-03</data:DateValue>
  <data:DateTimeOffsetValue metadata:type="DateTimeOffset"
       >2012-12-03T07:16:23Z</data:DateTimeOffsetValue>
  <data:DurationValue metadata:type="Duration"
       >P12DT23H59M59.999999999999S</data:DurationValue>
  <data:TimeOfDayValue metadata:type="TimeOfDay"
       >07:59:59.999</data:TimeOfDayValue>
  <data:GuidValue metadata:type="Guid"
       >01234567-89ab-cdef-0123-456789abcdef</data:GuidValue>
  <data:Int64Value metadata:type="Int64">0</data:Int64Value>
  <data:ColorEnumValue metadata:type="#org.example.Pattern"
       >Yellow</data:ColorEnumValue>
  <data:GeographyPoint metadata:type="GeographyPoint">
    <gml:Point>
      <gml:pos>64.1 142.1</gml:pos>
    </gml:Point>
  </data:GeographyPoint>
</metadata:properties>
```

Note that the line break in the body of `StringValue` is intentional, it represents a line break.

7.2 Element `metadata:properties`

The `metadata:properties` element represents property values for an entity.

7.3 Element `data:[PropertyName]`

Within the `metadata:properties` element, individual entity properties are represented as elements where the name of the element is the name of the entity property within the OData Data Namespace.

The `data:[PropertyName]` element MUST include a `metadata:type` attribute to specify the type of a primitive property whose type is not `Edm.String` or the type of a complex property whose type is derived from the type specified in the metadata document.

Example 9:

```
<data:Rating metadata:type="Int32">4</data:Rating>
```

The `data:[PropertyName]` element MUST be empty and MUST include a `metadata:null` attribute if the primitive- or complex-typed instance has the `null` value.

Example 10:

```
<data:Rating metadata:null="true"/>
```

7.3.1 Attribute `metadata:type`

If the type of the property is anything other than `Edm.String`, the property representation MUST contain a `metadata:type` attribute to specify the URI that identifies the type of the property.

For built-in primitive types the value is the unqualified name of the primitive type.

For non-built in primitive types, the URI may be an absolute or relative URL containing the namespace-qualified or alias-qualified type name as a fragment, or may simply contain the qualified type name prefixed with hash (`#`). In the latter case, the type MUST be defined or referenced in the metadata document defined by the current context URL. For properties that represent a collection of values, the fragment is the namespace-qualified or alias-qualified type name prefixed with `Collection` and enclosed in parentheses.

Example 11:

```
<data:Age metadata:type="Int32">25</data:Age>
```

7.3.2 Attribute `metadata:null`

Null-valued properties are represented as empty elements with the `metadata:null="true"` attribute.

The `metadata:null` attribute distinguishes null values from other empty content (such as an empty string).

Example 12:

```
<data:Apartment metadata:null="true"/>
```

The absence of the `metadata:null` attribute is equivalent to specifying `metadata:null="false"`.

7.4 Primitive and Enumeration Property

For primitive properties, the content of the `data:[PropertyName]` element represents the value of the property following the syntax for primitive values.

Example 13: string value

```
<data:Title>CEO</data:Title>
```

Example 14: enumeration value

```
<data:Pattern metadata:type="#org.example.Pattern">Solid,Yellow</data:Pattern>
```

7.5 Complex Property

For complex properties, the content of the `data:[PropertyName]` element consists of nested `data:[PropertyName]` elements describing the properties of the complex type.

Example 15:

```
<data:ShipTo metadata:type="#Model.Address">
  <data:Street>Obere Str. 57</data:Street>
  <data:City>Berlin</data:City>
  <data:Region metadata:null="true"/>
  <data:PostalCode>D-12209</data:PostalCode>
</data:ShipTo>
```

7.6 Primitive and Enumeration Property Collection

For properties that represent a collection of primitive or enumeration values, the URI fragment specified in the `metadata:type` attribute is the namespace-qualified or alias-qualified element type prefixed with `Collection` and enclosed in parenthesis.

7.6.1 Element `metadata:element`

Each item in the collection is represented as a `metadata:element` element in the OData Metadata namespace.

The value of each `metadata:element` in the collection follows the syntax for primitive values.

An empty `metadata:element` element with the `metadata:null="true"` attribute value represents a null value within the collection.

Example 16:

```
<data:EmailAddresses metadata:type="#Collection(String)">
  <metadata:element>Julie@Swansworth.com</metadata:element>
  <metadata:element>Julie.Swansworth@work.com</metadata:element>
</data:EmailAddresses>
```

7.7 Complex Property Collection

For properties that represent a collection of complex types, the URI fragment specified in the `metadata:type` attribute is the namespace-qualified or alias-qualified element type prefixed with "`Collection`" and enclosed in parenthesis..

7.7.1 Element `metadata:element`

Each item in the collection is represented as a `metadata:element` element in the OData Metadata namespace.

The value of each complex-typed `metadata:element` follows the syntax for complex-typed properties.

An empty `metadata:element` element with the `metadata:null="true"` attribute value represents a null value within the collection.

7.7.1.1 Attribute `metadata:type`

A `metadata:element` element MAY include a `metadata:type` attribute to specify the complex type of the represented instance. It MUST include a `metadata:type` attribute if the instance is of a type derived from the declared type of the property.

Example 17:

```
<data:PhoneNumbers metadata:type="#Collection(Model.PhoneNumber)">
  <metadata:element>
    <data:Number>425-555-1212</data:Number>
    <data:Type>Home</data:Type>
  </metadata:element>
  <metadata:element metadata:type="#Model.CellPhoneNumber">
    <data:Number>425-555-0178</data:Number>
    <data:Type>Cell</data:Type>
    <data:Carrier>Sprint</data:Carrier>
  </metadata:element>
</data:PhoneNumbers>
```


8 Navigation Property

A navigation property is a reference to zero or more related entities. It is represented as a navigation link that MAY be immediately preceded by an association link.

8.1 Navigation Link

The navigation link is a URL that allows retrieving the related entity or collection of entities. It is represented as an `atom:link` element.

Example 18: products related to a category

```
<atom:link
    rel="http://docs.oasis-open.org/odata/ns/related/Products"
    href="Categories(0)/Products"
    type="application/atom+xml;type=feed"
    title="Products"
/>
```

The related data for the relationship MAY be included in the entity using a single child `metadata:inline` element.

8.1.1 Element `atom:link`

In the case where the `atom:link` element describes a navigation link the attributes `rel`, `href`, `type`, `metadata:context`, and `title` MUST be used as described in the following subsections.

8.1.1.1 Attribute `rel`

The `rel` attribute MUST be present and MUST contain the string

`http://docs.oasis-open.org/odata/ns/related/`

followed by the name of the navigation property on the entity.

Note that the full name must be used; the use of relative URLs in the `rel` attribute is not allowed.

8.1.1.2 Attribute `href`

The `href` attribute MUST be present and specifies the URL that can be used to retrieve the related entities. This URL may be relative or absolute.

For navigation properties declared by an entity type the URL should be the canonical URL for the navigation property, i.e. the canonical URL of the source entity followed by a forward slash and the name of the navigation property, see Example 18.

For navigation properties declared by a complex type that is used as a single value in an entity type, the URL should be the canonical URL of the source entity, followed by a forward slash and the path to the navigation property, see second `atom:link` in Example 5.

For navigation properties declared by a complex type that is used in a collection of complex type values, the URL should be the canonical URL of the target entity.

Example 19: country related to an address within a collection

```
<data:Addresses metadata:type="#Collection(Model.Address)">
  ...
  <metadata:element>
    ...
    <atom:link rel="http://docs.oasis-open.org/odata/ns/related/Country"
               href="Countries('DE')"
               type="application/atom+xml;type=entry"
               title="Country" />
  </metadata:element>
  ...
</data:Addresses>
```

8.1.1.3 Attribute `type`

The `type` attribute MUST be present and determines whether the cardinality of the related end is a single entity or a collection of entities. The value `"application/atom+xml;type=entry"` represents a single entity and the value `"application/atom+xml;type=feed"` an collection of entities.

8.1.1.4 Attribute `metadata:context`

The `metadata:context` attribute MUST be present if the navigation property is not defined in metadata. The value of the `metadata:context` attribute, defined in the OData Metadata namespace, specifies the context URL that describes the type of the related entity or entities.

For details on the context URL, see **[OData-Protocol]**.

8.1.1.5 Attribute `title`

The `title` attribute SHOULD be present and equal to the name of the navigation property, and provides human-readable, possibly language-dependent, and not necessarily unique information about the link.

8.2 Association Link

The association link is a URL that allows retrieving the reference or collection of references to the related entity or entities. It is represented as an `atom:link` element. If the URL follows conventions, i.e. is the navigation link with `/$ref` appended, the association link MAY be omitted.

Example 20: products related to a category

```
<atom:link
  rel="http://docs.oasis-open.org/odata/ns/relatedlinks/Products"
  href="Categories(0)/Products/$ref"
  type="application/xml"
  title="Products"
/>
```

8.2.1 Element `atom:link`

A collection of relationship links is represented by an `atom:link` element. The attributes `rel`, `href`, `type`, and `title` MUST be used as described in the following subsections.

8.2.1.1 Attribute `rel`

The `rel` attribute MUST be present. The value MUST contain the string

`http://docs.oasis-open.org/odata/ns/relatedlinks/`

followed by the name of the navigation property of the entity.

Note that the full name must be used; the use of relative URLs in the `rel` attribute is not allowed.

8.2.1.2 Attribute `href`

The `href` attribute MUST be present and MUST specify the URL that represents the collection of relationship links. This URL may be relative or absolute.

8.2.1.3 Attribute `type`

The `type` attribute MUST be present with the string `"application/xml"` as value.

8.2.1.4 Attribute `title`

The `title` attribute SHOULD be present and be set to the name of the navigation property. The `title` attribute provides human-readable, possibly language-dependent, and not necessarily unique information about the link.

8.3 Expanded Navigation Property

An expanded navigation property MUST be represented as a single `metadata:inline` child element of the `atom:link` element representing the navigation link. The value of the `metadata:inline` element MUST be the correct representation of the related entity or collection of entities.

It is valid to include the `metadata:inline` element in only a subset of the entries within a feed.

If at most one entity can be related, the value is the representation of the related entity, or the `metadata:inline` element is empty if no entity is currently related.

If a collection of entities can be related, it MUST be represented as an `atom:feed`. An empty collection of entities (one that contains no entity type instances) MUST be represented as an empty `atom:feed`.

Each entity MUST be represented as an `atom:entry` element or as an entity reference.

Example 21:

```
<atom:link
  rel="http://docs.oasis-open.org/odata/ns/related/Products"
  href="Categories(0)/Products"
  type="application/atom+xml;type=feed"
  title="Products"
>
  <metadata:inline>
    <atom:feed>
      ...
    </atom:feed>
  </metadata:inline>
</atom:link>
```

8.4 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.

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

Example 22: inserting a new order for a new customer with order items related to existing products

```
<entry ...>
  ...
  <link rel="http://docs.oasis-open.org/odata/ns/related/Customer"
        type="application/atom+xml;type=entry"
        title="Customer" href="Orders(11643)/Customer">
    <metadata:inline>
      <entry>
        ...
```

```
        <content type="application/xml">
          <metadata:properties>
            <data:ID>ANEWONE</data:ID>
            ...
          </metadata:properties>
        </content>
      </entry>
      </metadata:inline>
  </link>
  <link rel="http://docs.oasis-open.org/odata/ns/related/Items"
        type="application/atom+xml;type=feed"
        title="Details" href="Orders(11643)/Items">
    <metadata:inline>
    <feed>
        <entry>
          ...
          <link rel="http://docs.oasis-open.org/odata/ns/related/Product"
                href="http://host/service/Products(28)"
                type="application/atom+xml;type=entry"
                title="Product"/>
          <content type="application/xml">
            <metadata:properties>
              ...
            </metadata:properties>
          </content>
        </entry>
          ...
        <entry>
          <link rel="http://docs.oasis-open.org/odata/ns/related/Product"
                href="http://host/service/Products(29)"
                type="application/atom+xml;type=entry"
                title="Product"/>
          <content type="application/xml">
            <metadata:properties>
              ...
            </metadata:properties>
          </content>
        </entry>
      ...
      </feed>
    </metadata:inline>
  </link>
  <content type="application/xml">
    <metadata:properties>
      <data:ID metadata:type="Int32">11643</data:ID>
      ...
    </metadata:properties>
  </content>
</entry>
```

8.5 Bind Operation

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

If at most one entity can be related, the bind operation MUST be represented as a navigation link whose `href` attribute MUST contain the id of the entity to be related.

For update operations a bind operation on a collection navigation property MUST be represented as a navigation link with an inlined collection of entity references. The referenced entities are added as additional related entities, and existing relationships are not updated or deleted.

For insert operations collection navigation property bind operations and deep insert operations MAY be combined by inlining an `atom:feed` that contains `atom:entry` elements and `metadata:ref` elements.

Example 23: assign a product to an existing category

```
<atom:link
    rel="http://docs.oasis-open.org/odata/ns/related/Category"
    href="http://host/service/Categories(6)"
    type="application/atom+xml;type=entry"
    title="Category"
/>
```


9 Stream Property

9.1 Element atom:link

An entity or complex type instance MAY have one or more stream properties (for example, a photo property of an employee entity). Properties that represent streams have a type of `Edm.Stream`.

OData uses the `atom:link` element to represent a stream property.

Example 24: read link of stream property `Thumbnail`

```
<atom:link rel="http://docs.oasis-open.org/odata/ns/mediaresource/Thumbnail"
           type="image/jpeg" title="Photo" href="Products(0)/Thumbnail"
/>
```

Example 25: edit link of stream property `Thumbnail`

```
<atom:link
  rel="http://docs.oasis-open.org/odata/ns/edit-media/Thumbnail"
  type="image/jpeg" title="Photo" href="Products(0)/Thumbnail"
/>
```

The attributes `rel`, `href`, `type`, `metadata:etag`, and `title` are to be used as described in the following subsections.

9.1.1 Attribute rel

The `rel` attribute MUST be present and MUST be made up of the string `http://docs.oasis-open.org/odata/ns/mediaresource/`, followed by the name of the stream property on the entity.

The `rel` attribute for an `atom:link` element that can be used to change a stream property value is made up of the string `http://docs.oasis-open.org/odata/ns/edit-media/`, followed by the name of the stream property on the entity.

In both cases the full name must be used; the use of relative URLs in the `rel` attribute is not allowed.

9.1.2 Attribute href

The `href` attribute MUST be present and MUST contain the URL that can be used to read, or write, the stream, according to the `rel` attribute. This URL may be relative or absolute.

9.1.3 Attribute type

The `type` attribute MAY be present and specifies the MIME-type of the stream.

9.1.4 Attribute metadata:etag

The `metadata:etag` attribute MAY be present and specifies an etag value that can be used in an `if-match` header to conditionally write to the stream property as described in **[OData-Protocol]**.

9.1.5 Attribute title

The `title` attribute MAY be present and provides human-readable, possibly language-dependent, and not necessarily unique information about the link.

10 Media Entity

Media entities (in AtomPub: media link entries, see **[RFC5023]**) are entities that describe and link to a media resource.

Example 26:

```
<entry ...>
  <id>http://host/service/Employees(6)</id>
  ...
  <link rel="edit-media" title="Employee" href="Employees(6)/$value"/>
  <content type="image/jpeg" src="Employees(6)/$value"/>
  <metadata:properties>
    <data:ID metadata:type="Int32">6</data:ID>
    ...
  </metadata:properties>
</entry>
```

10.1 Element `atom:link`

A media entity MAY contain an `atom:link` element with a `rel` attribute of "`edit-media`" to specify a URL that can be used to write to the BLOB associated with the entity. The attributes `rel` and `href` MUST be used as described in the following subsections.

10.1.1 Attribute `rel`

The `rel` attribute MUST be present and MUST have the string "`edit-media`" as value.

10.1.2 Attribute `href`

The `href` MUST be present and its value MUST specify the URL that can be used to write the stream. This URL may be relative or absolute.

10.2 Element `atom:content`

For media entities the `atom:content` element MUST be empty. Properties of the media entity are represented by the `metadata:properties` element as a sibling to, rather than a child of, the `atom:content` element.

10.2.1 Attribute `src`

The `atom:content` element MUST contain a `src` attribute and the value of the `src` attribute MUST be a URL that can be used to retrieve the content of the media resource.

10.2.2 Attribute `type`

The `atom:content` element MUST specify a `type` attribute that SHOULD contain the MIME type of the media resource.

11 Individual Property

A valid OData payload may consist of a single primitive or complex value, or of a collection of these.

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

11.1 Single Scalar Value

Example 27: string value

```
<value xmlns="http://docs.oasis-open.org/odata/ns/metadata"
       context="http://host/service/$metadata#Edm.String">CEO</value>
```

Example 28: primitive `null` value:

```
<metadata:value xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
                metadata:context="http://host/service/$metadata#Edm.Date"
                metadata:null="true" />
```

Example 29: complex value

```
<metadata:value metadata:type="#Model.Address"
          metadata:context="http://host/service/$metadata#Model.Address"
          xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
          xmlns="http://docs.oasis-open.org/odata/ns/data">
  <Street>Obere Str. 57</Street>
  <City>Berlin</City>
  <Region metadata:null="true"/>
  <PostalCode>D-12209</PostalCode>
</metadata:value>
```

11.1.1 Element `metadata:value`

Single scalar values are represented as a `metadata:value` root element that contains the representation of the scalar value. The attributes `metadata:type` and `metadata:null` MUST be used as described in the following subsections.

11.1.1.1 Attribute `metadata:context`

The `metadata:value` element MUST have a `metadata:context` attribute, defined in the OData Metadata namespace, whose value is the context URL that describes the element.

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

11.1.1.2 Attribute `metadata:metadata-etag`

The `metadata:value` element MAY have a `metadata:metadata-etag` attribute to specify an ETag for the service's metadata document. It can be used to determine whether the client's cached copy of the metadata document is outdated.

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

11.1.1.3 Attribute `metadata:type`

If the type of the scalar value being specified is anything other than `Edm.String` the `metadata:type` attribute MUST be present and specify the namespace - or alias - qualified type of the value.

11.1.1.4 Attribute `metadata:null`

The metadata:null attribute distinguishes null values from other empty content (such as an empty string).

Null-values are represented as an empty `metadata:value` element with a `metadata:null="true"` attribute.

11.2 Collection of Scalar Values

A valid OData payload MAY consist of a collection of primitive or complex properties.

Example 30: collection of strings

```
<metadata:value
  metadata:context="http://host/service/$metadata#Collection(Edm.String)"
  xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
>
  <metadata:element>(203)555-1718</metadata:element>
  <metadata:element>(203)555-1719</metadata:element>
</metadata:value>
```

Example 31: collection of complex values

```
<metadata:value xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
metadata:context="http://host/service/$metadata#Collection(Model.BaseAddress)"
  xmlns="http://docs.oasis-open.org/odata/ns/data"
>
  <metadata:element>
    <Street>Obere Str. 57</Street>
    <City>Berlin</City>
    <PostalCode>D-12209</PostalCode>
  </metadata:element>
  <metadata:element metadata:type="#Model.Address">
    <Street>12345 Grant Street</Street>
    <City>Taft</City>
    <Region>Ohio</Region>
    <PostalCode>OH 98052</PostalCode>
  </metadata:element>
</metadata:value>
```

11.2.1 Element `metadata:value`

A collection of scalar values is represented as a `metadata:value` root element that contains a `metadata:element` child element for each item of the collection whose content is an individual primitive or complex value as defined above.

The `metadata:value` element MUST NOT contain a `metadata:null` attribute. The attribute `metadata:type` MUST be used as described in the following subsection.

11.2.1.1 Attribute `metadata:context`

The `metadata:value` element MUST have a `metadata:context` attribute, defined in the OData Metadata namespace, whose value is the context URL that describes the element.

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

11.2.1.2 Attribute `metadata:metadata-etag`

The `metadata:value` element MAY have a `metadata:metadata-etag` attribute to specify an ETag for the service's metadata document. It can be used to determine whether the client's cached copy of the metadata document is outdated.

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

11.2.1.3 Attribute `metadata:type`

The attribute `metadata:type` MUST be present and specify the collection type according to the rules described in section 7.3.1.

For collections of complex scalar values this attribute specifies a collection type for the base type of the collection. Individual elements of a derived type MUST specify their derived type with a `metadata:type` attribute on the `metadata:element` element.

11.3 Element `atom:link`

The `metadata:value` element MAY contain a *next link* to indicate the presence of additional items that belong to the collection.

11.3.1 Attribute `rel`

A next link is represented as an `atom:link` with a `rel="next"` attribute and an `href` attribute containing a URL that can be used to retrieve the next set of results.

Example 33: next link

```
<atom:link rel="next"
      href="http://host/service/Suppliers('S01')/Addresses?$skiptoken=1237"/>
```

The contents of the `href` attribute SHOULD be treated as an opaque URL that can be used to fetch the next set of results and should not be modified other than resolving a relative URL.

12 Collection of Entities

Collections of entities are represented in Atom as an `atom:feed` element.

12.1 Element `atom:feed`

Collections of entities are represented using an `atom:feed` Element, where each entity is represented as an `atom:entry` or `metadata:ref` element.

12.1.1 Attribute `metadata:context`

If the root of the response is an `atom:feed` element, it MUST have a `metadata:context` attribute, defined in the OData Metadata namespace, whose value is the context URL that describes the entity set represented by the feed.

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

12.1.2 Attribute `metadata:metadata-etag`

The `metadata:metadata-etag` attribute MAY appear in an `atom:feed` in order to specify an ETag that can be used to determine the current version of the service's metadata document.

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

12.2 Element `atom:id`

The `atom:id` element MUST uniquely identify the collection from which the feed was generated.

12.3 Element `metadata:count`

The `atom:feed` element MAY contain a `metadata:count` element to specify the total count of entities in the result to the request. This MAY be greater than the number of entries in the feed, if server-side paging has been applied, in which case the feed MUST include a next results link.

Example 32:

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
      metadata:context="http://host/service/$metadata#Customers" >
  <metadata:count>42</metadata:count>
  ...
  <atom:link rel="next"
             href="http://host/service/Customers?$skiptoken=1237"/>
</feed>
```

12.4 Element `atom:link`

The `atom:feed` element MAY contain a *self link* to allow reread the feed.

The `atom:feed` element MAY contain a *next link* to indicate the presence of additional entities that belong to the collection.

The `atom:feed` element representing the final page of results MAY contain a *delta link* that can be used to fetch subsequent changes (deltas) to the result.

All three cases are distinguished from another by the value of the rel attribute as described in the following subsection.

In a valid OData Atom response Payload the atom:link element representing a *next link* or a *delta link* MAY be positioned after the last `atom:entry` or `metadata:ref` element. This defines an exception to the Atom Specification **[RFC4287]**.

12.4.1 Attribute `rel`

A self link is represented as an `atom:link` with a `rel="self"` attribute. The `href` attribute MUST contain the request URL that produced this collection.

A next link is represented as an `atom:link` with a `rel="next"` attribute and an `href` attribute containing a URL that can be used to retrieve the next set of results.

Example 33: next link

```
<atom:link rel="next"
           href="http://host/service/Customers?$skiptoken=1237"/>
```

The contents of the `href` attribute SHOULD be treated as an opaque URL that can be used to fetch the next set of results.

A delta link is represented as an `atom:link` element with a `rel` attribute of "`http://docs.oasis-open.org/odata/ns/delta`" and an `href` attribute containing a URL that can be used to retrieve subsequent changes.

Example 34: delta link

```
<atom:link rel=" http://docs.oasis-open.org/odata/ns/delta"
           href="http://host/service/Customers?$deltatoken=1234"/>
```

The contents of the `href` should be treated as an opaque URL that can be used to fetch subsequent changes.

The delta link MUST only appear on the last page of results. A page of results MUST NOT have both a delta link and a next link.

13 Entity Reference

An entity reference (see **[OData-Protocol]**) MAY take the place of an entity in an Atom payload, based on the client request. The id may be absolute or relative.

Example 35: entity reference to order 10643

```
<metadata:ref xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
              metadata:context="http://host/service/$metadata#$ref"
              id="http://host/service/Orders(10643)" />
```

Example 36: collection of entity references

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
      metadata:context="http://host/service/$metadata#Collection($ref)" >
  <metadata:ref id="http://host/service/Orders(10643)" />
  <metadata:ref id="http://host/service/Orders(10759)" />
</feed>
```

13.1 Element metadata:ref

A reference to an entity or one of its properties is represented in Atom using a `metadata:ref` element.

13.1.1 Attribute `metadata:context`

If the `metadata:ref` element is the root element of a response, it MUST have a `metadata:context` attribute, defined in the OData Metadata namespace, whose value is the context URL that describes the reference. If it is part of an Atom feed, the attribute is optional.

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

13.1.2 Attribute `id`

The `id` attribute MUST be present. For entities the `id` attribute MUST be the `atom:id` of the referenced entity. It may be relative or absolute.

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 an `atom:feed`. The feed MUST contain all added, changed, or deleted entities, as well as added or deleted links between entities, and MAY contain additional, unchanged entities.

All added, changed, or deleted entities and links, including related entities, are returned as direct children of the `atom:feed` element.

Entities that are not part of the entity set specified by the `metadata:context` attribute in the `atom:feed` element MUST include a `metadata:context` attribute in the `atom:entry` element to specify the entity set of the related entity.

If the delta response contains a partial list of changes, it MUST include a next link for the client to retrieve the next set of changes.

The last page of a delta response SHOULD contain a 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 a `metadata:count` element, the returned number MUST include all added, changed, or deleted entities, as well as added or deleted links.

Example 37: delta response with five changes, in order of occurrence

- *ContactName for customer 'BOTTM' was changed to "Susan Halvenstern"*
- *Order 10643 was removed from customer 'ALFKI'*
- *Order 10645 was added to customer 'BOTTM'*
- *The shipping information for order 10643 was updated*
- *Customer 'ANTON' was deleted*

```
<feed xml:base="http://host/service/"
      xmlns:data="http://docs.oasis-open.org/odata/ns/data"
      xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
      xmlns="http://www.w3.org/2005/Atom"
      xmlns:at="http://purl.org/atompub/tombstones/1.0"
      metadata:context="$metadata#Customers/$delta">
  <title type="text">Customers</title>
  <id>http://host/service/Customers</id>
  <updated>2012-11-27T15:38:25Z</updated>
  <metadata:count>5</metadata:count>
  <entry>
    <id>http://host/service/Customers('BOTTM')</id>
    <title type="text" />
    <updated>2012-11-17T15:38:22Z</updated>
    <author><name /></author>
    <link rel="edit" title="Customer" href="Customers('BOTTM')"/>
    <category term="#Model.Customer"
              scheme="http://docs.oasis-open.org/odata/ns/scheme"/>
    <content type="application/xml">
      <metadata:properties>
        <data:ContactName>Susan Halvenstern</data:ContactName>
      </metadata:properties>
    </content>
  </entry>
```

```
  <metadata:deleted-link
    metadata:context="$metadata#Customers/$deleted-link"
    source="http://host/service/Customers('ALFKI')"
    relationship="Orders"
    target="http://host/service/Orders(10643)"/>
  <metadata:link
    metadata:context="$metadata#Customers/$link"
    source="http://host/service/Customers('BOTTM')"
    relationship="Orders"
    target="http://host/service/Orders(10645)"/>
  <entry metadata:context="$metadata#Orders/$entity">
   <id>http://host/service/Orders(10643)</id>
   <title type="text" />
   <updated>2012-11-27T15:38:24Z</updated>
   <author><name/></author>
   <link rel="edit" title="Order" href="Orders(10643)" />
   <category term="#Model.Order"
             scheme="http://docs.oasis-open.org/odata/ns/scheme" />
   <content type="application/xml">
     <metadata:properties>
       <data:ShippingAddress>
         <data:Street>23 Tsawassen Blvd.</data:Street>
         <data:City>Tsawassen</data:City>
         <data:Region>BC</data:Region>
         <data:PostalCode>T2F 8M4</data:PostalCode>
       </data:ShippingAddress>
     </metadata:properties>
   </content>
  </entry>
  <at:deleted-entry
    metadata:context="$metadata#Customers/$deleted-entry"
    ref="http://host/service/Customers('ANTON')"
    when="2012-11-27T15:38:25Z"
    metadata:reason="deleted"/>
  <link
    rel="http://docs.oasis-open.org/odata/ns/delta"
    href="http://host/service/Customers?$expand=Orders&amp;$deltatoken=8015"/>
</feed>
```

14.1 Added/Changed Entity

Added or changed entities within a delta response are represented as `atom:entry` elements.

Added or changed entities MUST NOT include inline content.

Added entities MUST include all 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.

Added entities MUST include navigation links.

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

Entities whose set cannot be determined from the context URL of the feed MUST include the `metadata:context` attribute in the `atom:entry` element to specify the set that the entity belongs to.

14.2 Deleted Entity

14.2.1 Element atom-tombstone:deleted-entry

A deleted entity within a delta response is represented as an `atom-tombstone:deleted-entry` element, defined within the Atom Tombstone namespace, as defined in **[RFC6721]**.

The `ref` and a `when` attribute MUST be present, the `metadata:reason` attribute MAY be present. All attributes have to be used as described in the following subsection.

14.2.1.1 Attribute `ref`

As defined in **[RFC6721]**, the `ref` attribute MUST be present. The value of the `ref` attribute MUST specify the `atom:id` of the deleted entry. It may be relative or absolute.

14.2.1.2 Attribute `when`

As defined in **[RFC6721],** the `when` attribute MUST be present to specify the time at which the entity was deleted. This attribute is not used in OData and MAY be set to the time the delta response was generated if the service does not track when deletions occur. OData clients MUST NOT assume any semantics around this value.

14.2.1.3 Attribute metadata:reason

The `metadata:reason` attribute MAY be present. The value of the `metadata:reason` attribute MUST specify the string value "`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).

14.3 Added Link

14.3.1 Element metadata:link

A link within a delta response is represented by a `metadata:link` element.

A delta response MUST contain a `metadata:link` for each added link that corresponds to a `$expand` path in the initial request.

The `source`, `relationship`, and `target` attribute MUST be present. All attributes have to be used as described in the following subsection.

14.3.1.1 Attribute `source`

The `source` attribute MUST be present and specify the `atom:id` of the entity from which the link originates. It may be relative or absolute.

14.3.1.2 Attribute `relationship`

The `relationship` MUST be present and specify the name of the navigation property on the `source` entity for which the link exists.

14.3.1.3 Attribute `target`

The `target` attribute MUST be present and specify the `atom:id` of the related entity. It may be relative or absolute.

14.4 Deleted Link

14.4.1 Element metadata:deleted-link

A deleted link within a delta response is represented as a `metadata:deleted-link` element.

Delta responses MUST contain a `metadata:deleted-link` 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 `metadata:link` that specifies the same `source` and `relationship`.

The service MAY return a `metadata:deleted-link` where one of the entities has also been deleted, or where there is a subsequent `metadata:link` with the same `source` and `relationship` and a maximum cardinality of one for the related end.

The `source`, `relationship` and `target` attribute MUST be present. All attributes have to be used as described in the following subsection.

14.4.1.1 Attribute `source`

The `source` attribute MUST be present and specify the `atom:id` of the entity from which the link originates. It may be relative or absolute.

14.4.1.2 Attribute `relationship`

The `relationship` attribute MUST be present and specify the name of the navigation property on the `source` entity for which the link is deleted.

14.4.1.3 Attribute `target`

The `target` attribute MUST be present and specify the `atom:id` of the related entity. It may be relative or absolute.

15 Bound Function

Zero or more functions MAY be bound to a collection of entities or an entity.

The functions associated with a particular collection of entities or an entity MAY be described using `metadata:function` elements that are direct children of the feed or entry to which the functions can be bound.

Example 38: a function bound to an entry:

```
<atom:entry>
  ...
  <metadata:function
    metadata="#Model.RemainingVacation"
    target="http://host/service/Employees(2)/RemainingVacation"
    title="Remaining Vacation"
  />
  ...
</atom:entry>
```

Example 39: a function bound to a feed:

```
<atom:feed>
  ...
  <metadata:function
    metadata="#Model.RemainingVacation"
    target="http://host/service/Managers(22)/Employees/RemainingVacation"
    title="Remaining Vacation"
  />
  ...
</atom:feed>}
```

15.1 Element `metadata:function`

Each function is represented as a `metadata:function` element that MUST be a child of the `atom:feed` or `atom:entry` element representing the collection of entities or the entity on which the function exists.

15.1.1 Attribute `metadata`

The `metadata` attribute MUST be present and specify the namespace-qualified or alias-qualified name of the function, preceded by a `#`.

A function may have multiple overloads with different parameters. If function overloads exist that cannot be bound to the current entity type, the `metadata` attribute SHOULD address a specific function overload by appending the parentheses-enclosed, comma-separated list of non-binding parameter names, see rule `qualifiedFunctionName` in **Error! Reference source not found.**. If the URL in the `target` attribute of the `metadata:function` element cannot be used to invoke all overloads for the function, then it MUST further be distinguished by appending the parentheses-enclosed, comma-separated list of non-binding parameter names.

Example 40:

```
<metadata:function
  metadata="#Model.RemainingVacation(Year)"
  target="http://host/service/Employees(2)/RemainingVacation(Year=@Year)"
  title="Remaining vacation from year..."
/>
```

15.1.2 Attribute target

The `target` attribute MUST be present and specify the URL to GET from in order to invoke the function.

The first parameter of the function MUST be a binding parameter that is bound to the feed or entity on which the function is specified, and MUST NOT be provided as a separate parameter by the client when invoking the function.

15.1.3 Attribute `title`

The `title` attribute MUST be present and contain a human-readable, possibly language-dependent, and not necessarily unique name for the function, commonly used by clients to describe the function to a user.

16 Bound Action

Zero or more actions MAY be bound to a collection of entities or an entity.

The actions associated with a particular collection of entities or an entity MAY be described using `metadata:action` elements that are direct children of the feed or entry to which the actions can be bound.

Example 41: action bound to an entity

```
<atom:entry>
  ...
  <metadata:action
    metadata="#Model.Approval"
    target="http://host/service/LeaveRequests(2)/Approval"
    title="Approve Leave Request"
  />
  ...
</atom:entry>
```

Example 42: action bound to a feed

```
<atom:feed>
  ...
  <metadata:action
    metadata="#Model.Approval"
    target="http://host/service/Managers(22)/Inbox/Approval"
    title="Approve All Leave Requests"
  />
  ...
</atom:feed>
```

16.1 Element `metadata:action`

Each action is represented as a `metadata:action` element that MUST be a direct child of the `atom:feed` or `atom:entry` element representing the the collection of entities or the entity on which the action exists.

16.1.1 Attribute `metadata`

The `metadata` attribute MUST be present and specify the namespace-qualified or alias-qualified name of the action element describing the action, preceded by a `#`.

16.1.2 Attribute `target`

The `target` attribute MUST be present and specify the URL to POST to in order to invoke the action.

The first parameter of the action MUST be a binding parameter that is bound to the feed or entity on which the action is specified, and MUST NOT be provided as a separate parameter by the client when invoking the action.

16.1.3 Attribute `title`

The `title` attribute MUST be present and contain a human-readable, possibly language-dependent, and not necessarily unique name for the action, commonly used by clients to describe the action to a user.

17 Action Invocation

Action parameter values in the request body MUST be encoded as an individual complex scalar value with the name `parameters` and no `metadata:type` attribute for the `parameters` element.

Each non-binding parameter value specified MUST be encoded as an individual primitive or complex scalar value. The name of the scalar value is the name of the parameter. The value is the parameter value in the XML representation appropriate for its type.

Any parameter values not specified in the request body MUST be assumed to have the `null` value.

Example 43:

```
<parameters>
  <param1>42</param1>
  <param2 metadata:type="#Model.Address">
    <Street>One Microsoft Way</Street>
    <Zip>98052</Zip>
  </param2>
  <param3>
    <element>1</element>
    <element>42</element>
    <element>99</element>
  </param3>
  <param4 metadata:null="true"/>
  <!-- <param5/> not specified, has null value -->
<parameters>
```

18 Instance Annotations

Annotations MAY be applied to an instance of a feed, entry, entity reference, complex scalar value, property, navigation property, function, action, added link, deleted link, or error within an Atom payload.

18.1 Element metadata:annotation

An instance annotation in Atom is represented as an XML element with the name `Annotation` in the metadata namespace.

The value of the annotation is specified according to the Annotation Value, described below.

18.1.1 Attribute target

The `target` attribute MAY be used to specify the annotation target. If the `target` attribute is not specified the target of the annotation is the element represented by the direct parent of the `metadata:annotation` element.

18.1.2 Attribute term

The `metadata:annotation` element MUST have a `term` attribute that specifies the namespace-qualified or alias-qualified name of the term being applied.

18.1.3 Attribute metadata:type

If the type of the annotation value being specified is anything other than `Edm.String` the `metadata:annotation` element MUST contain a `metadata:type` attribute to specify the appropriate type of the annotation value.

18.1.4 Attribute metadata:null

Null-valued annotations are represented as empty `metadata:annotation` elements with the `metadata:null="true"` attribute.

The `metadata:null` attribute distinguishes null values from other empty content (such as an empty string).

The absence of the `metadata:null` attribute is equivalent to specifying `metadata:null="false"`.

18.2 Annotation Value

An instance annotation value may be specified as a primitive value, collection value, or structured value.

18.2.1 Primitive Value

When specified in the content of an annotation element representing a primitive value, the content MUST be formatted as per Primitive Types in Atom. If the type of the annotation value is anything other than `Edm.String`, then the annotation element MUST contain the `metadata:type` attribute specifying the appropriate primitive type.

Example 44:

```
<entry ...>
  <id>Customers('ALFKI')</id>
  <content>
    <metadata:properties>
      <data:ID>ALFKI</data:ID>
      <data:CompanyName>Alfreds Futterkiste</data:CompanyName>
    </metadata:properties>
  </content>
  <metadata:annotation term="com.contoso.display.highlight"
                       metadata:type="Boolean">true</metadata:annotation>
</entry>
```

18.2.2 Collection Value

The content of an element representing a collection-valued annotation MUST be the individual elements of that collection formatted as direct child elements of the `metadata:annotation` element as described in Collections of Primitive or Collection of Complex Scalar Values.

For collection-valued annotations, the annotation element MUST contain the `metadata:type` attribute specifying the appropriate collection type.

Example 45:

```
<entry>
  <id>Customers('ALFKI')</id>
  <content>
    <metadata:properties>
      <data:ID>ALFKI</data:ID>
      <data:CompanyName>Alfreds Futterkiste</data:CompanyName>
    </metadata:properties>
  </content>
  <metadata:annotation term="com.contoso.PersonalInfo.PhoneNumbers"
                       type="Collection(String)">
    <element>(203)555-1718</element>
    <element>(203)555-1719</element>
  </metadata:annotation>
</entry>
```

18.2.3 Structured Value

The content of an element representing a structured annotation MUST be a single child element for each property of the annotation type being specified, formatted as per properties within an entity type.

For structural-valued annotations, the annotation element MUST contain the `metadata:type` attribute specifying the appropriate structural type.

Example 46:

```
<entry>
  <id>Customers('ALFKI')</id>
  <link rel="http://docs.oasis-open.org/odata/ns/related/Orders"
        href="Customers('ALFKI')/Orders"
        type="application/atom+xml;type=feed"
        title="List of Orders">
    <metadata:annotation term="com.contoso.display.style"
                         metadata:type="#com.contoso.display.styleType">
      <data:order metadata:type="Int32">2</data:order>
    </metadata:annotation>
  </link>
```

```
  <content>
    <metadata:properties>
      <data:CustomerID>ALFKI</data:CustomerID>
      <data:CompanyName>Alfreds Futterkiste</data:CompanyName>
      <metadata:annotation term="com.contoso.display.style"
                           target="CompanyName"
                           metadata:type="#com.contoso.display.styleType">
        <data:title metadata:type="Boolean">true</data:title>
        <data:order metadata:type="Int32">1</data:order>
      </metadata:annotation>
    </metadata:properties>
  </content>
<entry>
```

18.3 Instance Annotation Target

Instance annotations MAY target model elements represented by a feed, entry, complex scalar value, property, navigation property, function, action, or error within an Atom payload.

18.3.1 Feed

When annotating a feed, annotation elements MUST be direct children of the `atom:feed` element, and they MUST appear in a group at the beginning of the feed or (another) group at the end of the feed, depending on whether they are needed beforehand to understand the feed content, or can only be computed after serializing the feed content.

18.3.2 Entry

When annotating an entity, the annotation element MUST be a direct child of the `atom:entry` element representing the entity.

18.3.3 Entity Reference

When annotating an entity reference, the annotation element MUST be a direct child of the `metadata:ref` element.

18.3.4 Complex Type

When annotating an instance of a complex type, the annotation element MUST be a direct child of the metadata:value element representing the complex-typed value.

18.3.5 Property

When annotating a property, the annotation element MUST be a direct child of the `metadata:properties` element, or a direct child of the element representing a complex type in the case of annotating the property of a complex type. The value of the `target` attribute MUST specify the name of the property being annotated. The annotation elements MUST immediately precede the target property element.

Instance annotations are not supported when serializing single primitive properties in XML as described in Individual Primitive or Complex Scalar Values.

18.3.6 Navigation Property

When annotating a navigation property, stream property, or other element represented by an `atom:link` element, the annotation element must be a direct child of the `atom:link` element.

18.3.7 Function or Action

When annotating a function or action, the annotation element must be a direct child of the `metadata:function` or `metadata:action` element.

18.3.8 Added Link or Deleted Link

When annotating an added or deleted link in a delta response, the annotation element must be a direct child of the `metadata:link` or `metadata:deleted-link` element.

18.3.9 Error

When annotating an error, the `metadata:annotation` element MUST be a direct child of the `metadata:error` element. The annotation element MAY have a `target` attribute value of "`code`", "`message`", or "`innererror`". If the `target` attribute is not specified, then the annotation is applied to the error itself. The annotation elements MUST follow the other child elements of the error element.

19 Error Reponse

In the case of an error being generated in response to a request specifying an Accept header of `application/xml` or `application/atom+xml`, or that does not specify an Accept header, the service MUST respond with an error formatted as XML.

When formatting error responses as XML, services SHOULD include a `Content-Type` response header with the value `"application/xml"`.

19.1 Element `metadata:error`

Errors formatted as XML MUST have a root `metadata:error` element. The `metadata:error` element MUST have at least two child elements: `metadata:code` and `metadata:message`.

In addition, errors may be annotated using custom annotations.

Example 47:

```
<error xmlns="http://docs.oasis-open.org/odata/ns/metadata">
  <code>501</code>
  <message>Unsupported functionality</message>
  <target>query</target>
  <details>
    <detail>
      <code>301</code>
      <message>$search query option not supported</message>
      <target>$search</target>
    </detail>
  </details>
</error>
```

19.2 Element `metadata:code`

The `metadata:error` element MUST contain one `metadata:code` element specifying a service-defined string. This value MAY be used to provide a more specific substatus to the returned HTTP response code.

19.3 Element `metadata:message`

The `metadata:error` element MUST contain a `metadata:message` element specifying a human readable, language-dependent message describing 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.

19.4 Element `metadata:target`

The `metadata:error` element MAY contain a `metadata:target` element to specify the target of the error (for example, the name of the property in error).

19.5 Element `metadata:details`

The `metadata:error` element MAY contain a `metadata:details` element containing one or more `metadata:detail` elements specifying detail about the error.

19.5.1 Element metadata:detail

The metadata:detail element specifies information about an individual error detail.

19.5.2 Element metadata:code

The metadata:detail element MUST contain one metadata:code element specifying a service-defined string. This value MAY be used to provide a more specific substatus to the returned HTTP response code.

19.5.3 Element metadata:message

The metadata:detail element MUST contain a metadata:message element specifying a human readable, language-dependent message describing the error.

19.5.4 Element metadata:target

The metadata:detail element MAY contain a metadata:target element to specify the target of the error.

19.6 Element metadata:innererror

The metadata:error element MAY contain a metadata:innererror element containing service specific debugging information that might assist a service implementer in determining the cause of an error.

The metadata:innererror element SHOULD only be used in development environments in order to guard against potential security concerns around information disclosure.

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 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]**.

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 ATOM-relevant security implications please cf. the relevant sections of **[RFC4287]** (8. Security Considerations), **[RFC5023]** (15. Security Considerations) and for the deleted-entry element: see **[RFC6721]** (7. Security Considerations) as starting points.

22 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, which are only required for clients that request changes

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

1. MUST be prepared to receive all data types (section 7.1)
   a. defined in this specification (client)
   b. exposed by the service (service)
2. MUST be prepared to receive custom annotations (section 18)
3. MUST be prepared to receive additional constructs not defined in this version of the specification (section 20)

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

4. MUST comply with one of the conformance levels defined in **[OData-Protocol]**
5. MUST support the `application/atom+xml`, `application/xml` and `application/atomsvc+xml` media types in the `Accept` header (section 3)
6. MUST include the next link in feeds containing partial results (section 12.4)
7. MUST return service documents as Atom service documents (section 5)
8. MUST return XML responses in well formed XML according to this OData Atom specification
9. MUST return well-formed Atom payloads with the exceptions for the next link and the delta link (section 12.4)
10. MUST support entity instances with external metadata (section 6.1.2)
11. MUST support properties with externally defined data types (section 11.1.1.3)
12. MUST NOT violate any other aspects of this OData Atom specification
13. SHOULD support the `$format` system query option (section 3)

Appendix A. Acknowledgments

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

Appendix B. Revision History

Revision	Date	Editor	Changes Made
Working Draft 01	2012-08-22	Michael Pizzo	Translated Contribution to OASIS format/template
Committee Specification Draft 01	2013-04-26	Martin Zurmuehl Ralf Handl Michael Pizzo	Expanded error information Added enumerations Fleshed out descriptions and examples and addressed numerous editorial and technical issues processed through the TC Added Conformance section
Committee Specification Draft 02	2013-07-01	Martin Zurmuehl Ralf Handl Michael Pizzo	Improved metadata:type Improved entity references Simplified delta responses GML for Geo types Improved description of primitive value representation Improved examples, aligned with JSON format specification Aligned terms across specifications
Committee Specification 01	2013-07-30	Martin Zurmuehl Ralf Handl Michael Pizzo	Non-Material Changes
Committee Specification Draft 03	2013-10-03	Martin Zurmuehl Ralf Handl Michael Pizzo	Next link for collections of complex and primitive types Null elements in collections of complex and primitive types Binary values are base64url-encoded