OData Atom Format Version 4.0

Committee Specification Draft 01 / Public Review Draft 01

26 April 2013

Specification URIs

This version:

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

Previous version:

N/A

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 which also includes:

- OData Metadata XML Schema: http://docs.oasis-open.org/odata/odata-atom-format/v4.0/csprd01/schemas/metadata.xsd
- OData Atom Vocabulary: http://docs.oasis-open.org/odata/odata-atom-format/v4.0/csprd01/vocabularies/Org.OData.Atom.V1.xml

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.* http://docs.oasis-open.org/odata/odata/v4.0/csprd01/abnf/

- *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) is a set of specifications for representing and interacting with structured content. This document extends the core OData Protocol 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. 26 April 2013. OASIS Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/odata/odata-atom-format/v4.0/csprd01/odata-atom-format-v4.0-csprd01.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 7
1.1 Terminology 7
1.2 Normative References 7
2 Atom Format Design 9
2.1 Namespaces 9
2.1.1 Atom Syndication 9
2.1.2 Atom Publishing Protocol 9
2.1.3 Atom Tombstone 9
2.1.4 OData Data 9
2.1.5 OData Metadata 9
2.1.6 XML Schema Definition for OData Metadata 10
3 Requesting the Atom Format 11
4 Common Characteristics 12
4.1 Header `Content-Type` 12
4.2 Message Body 12
5 Service Document 13
5.1 Element `app:service` 13
5.1.1 Element `app:workspace` 13
5.1.2 Element `app:collection` 14
5.1.3 Element `metadata:function-import` 14
5.1.4 Element `metadata:entity` 15
5.1.5 Element `metadata:service-document` 15
6 Entity 16
6.1 Element `atom:entry` 16
6.1.1 Attribute `metadata:etag` 16
6.1.2 Attribute `metadata:metadata` 16
6.1.3 Attribute `metadata:metadata-etag` 17
6.2 Element `atom:id` 17
6.2.1 Element `atom:category` 17
6.3 Element `atom:content` 17
6.3.1 Self and Edit Links 17
7 Property 19
7.1 Primitive Value 19
7.2 Element `metadata:properties` 19
7.3 Element `data:[PropertyName]` 19
7.3.1 Primitive and Enumeration Property 20
7.3.2 Complex Property 20
7.3.3 Primitive and Enumeration Collection Property 20
7.3.4 Complex Collection Property 20
7.3.5 Attribute `metadata:null` 21
7.3.6 Attribute `metadata:type` 21
8 Navigation Property 22

8.1.1 Element atom:link 22
8.2 Association Link 23
8.2.1 Element atom:link 23
8.3 Expanded Navigation Property 23
8.4 Deep Inserts 24
8.5 Bind Operations 25
9 Stream Property 26
9.1 Element atom:link 26
9.1.1 Attribute rel 26
9.1.2 Attribute href 26
9.1.3 Attribute type 26
9.1.4 Attribute metadata:etag 26
9.1.5 Attribute title 26
10 Media Entity 27
10.1 Element atom:link 27
10.1.1 Attribute rel 27
10.1.2 Attribute href 27
10.2 Element atom:content 27
10.2.1 Attribute src 27
10.2.2 Attribute type 27
11 Individual Property 28
11.1 Single Scalar Value 28
11.1.1 Element metadata:value 28
11.2 Collection of Scalar Values 28
11.2.1 Element metadata:value 29
12 Collection of Entities 30
12.1 Element atom:feed 30
12.1.1 Attribute metadata:metadata 30
12.1.2 Attribute metadata:metadata-etag 30
12.1.3 Element atom:id 30
12.1.4 Element metadata:count 30
12.1.5 Element atom:link 30
13 Resource Reference 32
13.1 Element metadata:ref 32
13.1.1 Attribute ref 32
14 Delta Response 33
14.1 Added/Changed Entity 34
14.2 Deleted Entity 34
14.2.1 Element atom-tombstone:deleted-entry 34
14.3 Link 35
14.3.1 Element metadata:link-entry 35
14.4 Deleted Link 35
14.4.1 Element metadata:deleted-link-entry 35
15 Function 37

15.1 Element metadata:function 37
15.1.1 Attribute metadata 37
15.1.2 Attribute title 37
16 Action 38
16.1 Element metadata:action 38
16.1.1 Attribute metadata 38
16.1.2 Attribute target 38
16.1.3 Attribute title 38
17 Action Parameters 39
18 Instance Annotations 40
18.1 Element metadata:annotation 40
18.1.1 Attribute target 40
18.1.2 Attribute term 40
18.1.3 Attribute metadata:type 40
18.1.4 Attribute metadata:null 40
18.2 Annotation Values 40
18.2.1 Primitive Values 40
18.2.2 Collection Values 41
18.2.3 Structure Annotations 41
18.3 Instance Annotation Targets 42
18.3.1 Feed 42
18.3.2 Entry 42
18.3.3 Complex Type 42
18.3.4 Property 42
18.3.5 Navigation Property 42
18.3.6 Function or Action 42
18.3.7 Error 42
19 Error Reponse 43
19.1 Element metadata:error 43
19.2 Element metadata:code 43
19.3 Element metadata:message 43
19.4 Element metadata:target 43
19.5 Element metadata:details 43
19.5.1 Element metadata:detail 43
19.5.2 Element metadata:code 44
19.5.3 Element metadata:message 44
19.5.4 Element metadata:target 44
19.6 Element metadata:innererror 44
20 Extensibility 45
21 Conformance 46
Appendix A. Acknowledgments 47
Appendix B. Revision History 48

1 Introduction

The OData protocol is comprised of a set of specifications for representing and interacting with structured content. This document describes the OData Atom Format of the payload returned from an OData Service when requesting the `application/atom+xml` mime type.

An OData payload may represent:

- a single primitive value
- a sequence of primitive values
- a single structured ("complex") value
- a sequence of structured ("complex") values
- an entity (a structured type with an identity)
- a resource reference
- a sequence of entities
- a sequence of changes
- a media resource
- a single instance of a mime type
- a single link to a related entity
- a collection of links to related entities
- a service document describing the collections (entity sets) exposed by the service
- an xml document describing the entity model exposed by the service
- an error document
- 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 **[Error! Reference source not found.]**.

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-VocAtom] *OData Atom Vocabulary.*
See link in “Additional artifacts” 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.

2 Atom Format Design

The Atom Syndication Format **[RFC4287]** defines an XML-based format for describing collections ("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.

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 annotations 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: `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: `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: `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.

2.1.6 XML Schema Definition for OData Metadata

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

It only define 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`.

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`.

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, refereces, 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.

Message Body

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 or a complex type instance, a primitive value, a collection of primitive values, a collection of complex values, a collection of entities, or a collection of entries that represent changes to a previous result.

Relative URLs

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

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, named entities, and parameterless function imports published by the service.

Example:

```
<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"
  metadata:metadata="http://host:port/service/$metadata"
  metadata:metadata-etag="Hugo"
>
  <app:workspace>
    <atom:title type="text">Data</atom:title>
    <app:collection href="OrderDetails">
      <atom:title type="text">Order Details</atom:title>
    </app:collection>
    <metadata:entity href="Contoso">
      <atom:title>Contoso Ltd.</atom:title>
    </metadata:entity>
    <metadata:function-import href="TopProducts">
      <atom:title>Best-Selling Products</atom:title>
    </metadata:function-import>
    <metadata:service-document href="EasternRegionSales">
      <atom:title>Eastern Region Sales</atom:title>
    </metadata:service-document>
  </app:workspace>
</app:service>
```

5.1 Element `app:service`

The service document contains a single `app:service` element. The `app:service` element contains one or more `app:workspace` elements.

5.1.1 Element `app:workspace`

OData represents the each entity container 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 continer, and zero or more `metadata:entity` elements, one for each named entity published by the container.

5.1.1.1 Attribute `metadata:name`

The `metadata:name` attribute MUST contain the namespace- or alias-qualified name of the entity container that is represented by the `app:workspace` element. It MAY be omitted if the workspace represents the default entity container.

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

5.1.1.2 Attribute `metadata:metadata`

An `app:workspace` element MUST have a `metadata` 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]**.

5.1.1.3 Attribute `metadata:metadata-etag`

An `app:workspace` 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 **[Error! Reference source not found.]**.

5.1.1.4 Element `title`

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

5.1.2 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.

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

5.1.2.1 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.

5.1.2.2 Attribute `metadata:name`

The `metadata:name` attribute MUST contain the name of the entity set which MAY be unqualified if the enclosing `app:workspace` represents the default entity container.

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 **Error! Reference source not found.**.

5.1.2.3 Element `atom:title`

The `atom:title` element within the `app:collection` MUST contain a human-readable description of the entity set which MAY be the name of the entity set.

5.1.3 Element `metadata:function-import`

OData represents function imports that are marked with `IncludeInServiceDocument="true"` (see **[OData-CSDL]**) as `metadata:function-import` elements.

5.1.3.1 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.

5.1.3.2 Attribute `metadata:name`

The `metadata:name` attribute MUST contain the name of the function import which MAY be unqualified if the enclosing `app:workspace` represents the default entity container.

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 **Error! Reference source not found.**.

5.1.3.3 Element atom:title

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

5.1.4 Element metadata:entity

OData represents named entities as metadata:entity elements.

5.1.4.1 Attribute href

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

5.1.4.2 Attribute metadata:name

The metadata:name attribute MUST contain the name of the entity which MAY be unqualified if the enclosing app:workspace represents the default entity container.

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 **Error! Reference source not found.**.

5.1.4.3 Element atom:title

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

5.1.5 Element metadata:service-document

OData represents related service documents as metadata:service-document elements.

```
<metadata:service-document href="EasternRegionSales">
  <atom:title>Eastern Region Sales</atom:title>
</metadata:service-document>
```

5.1.5.1 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.

5.1.5.2 Element atom:title

The metadata:service-document element MUST contain an atom:title element containing 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:

```
<entry>
  <id>http://services.odata.org/OData/OData.svc/Products(0)</id>
  <title />
  <summary />
  <updated>2012-03-30T07:11:05Z</updated>
  <author>
    <name />
  </author>
  <link rel="edit" title="Product" href="Products(0)" />
  <link rel="http://docs.oasis-open.org/odata/ns/related/Category"
        type="application/atom+xml;type=entry"
        title="Category" href="Products(0)/Category" />
  <link rel="http://docs.oasis-open.org/odata/ns/related/Supplier"
        type="application/atom+xml;type=entry"
        title="Supplier" href="Products(0)/Supplier" />
  <category term="ODataDemo.Product"
            scheme="http://docs.oasis-open.org/odata/ns/scheme" />
  <content type="application/xml">
    <metadata:properties>
      <data:ID metadata:type="Int32">0</data:ID>
      <data:Name>Bread</data:Name>
      <data:Description>Whole grain bread</data:Description>
      <data:ReleaseDate metadata:type="Date">
         1992-01-01
      </data:ReleaseDate>
      <data:DiscontinuedDate metadata:type="Date" metadata:null="true" />
      <data:Rating metadata:type="Int32">4</data:Rating>
      <data:Price metadata:type="Decimal">2.5</data:Price>
    </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 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 **[Error! Reference source not found.]**.

6.1.2 Attribute `metadata:metadata`

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

For more information on the metadata URL, see **[Error! Reference source not found.]**.

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 **[Error! Reference source not found.]**.

6.2 Element `atom:id`

The `atom:id` element defines a durable, opaque, globally unique identifier for the entry. Its content must be an IRI as defined in **RFC3987**. The consumer of the feed must not assume this IRI can be de-referenced, nor assume any semantics from its structure.

6.2.1 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 the namespace- or alias-qualified name of the entity type of the entry, in which case the type MUST be defined in the metadata document indicated by the current metadata URL, otherwise it MUST be a full URL to a metadata document with the namespace-qualified name of the instance's type appended as a URL fragment.

For example, the following represents an entity whose type is "`Model.VipCustomer`", defined within the current metadata document:

```
<category rel="http://docs.oasis-open.org/odata/ns/scheme"
          term="Model1.VipCustomer"/>
```

The following represents an entity whose type is "`Model.VipCustomer`", defined within the "`http://host/alternate/$metadata`" document:

```
<category rel="http://docs.oasis-open.org/odata/ns/scheme"
          term="http://host/alternate/$metadata#Model1.VipCustomer"/>
```

For more information on namespace- 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.

6.3 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.

6.3.1 Self and Edit Links

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 SHOULD contain a self link, an edit link, or both for a particular entry, but MUST NOT contain more than one edit link for a given entity. The absence of a

self link implies that the edit link can be used to retrieve the entity. The absence of an edit link implies that the entity is read-only.

7 Property

7.1 Primitive Value

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

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

[OData-ABNF].

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

Values of the other primitive types are represented according to the `xxxValue` rules, i.e. `Edm.Binary` as `binaryValue` etc.

Example:

```
<metadata:properties>
  <data:NullValue metadata:null="true"/>
  <data:TrueValue metadata:type="Boolean">true</data:TrueValue>
  <data:FalseValue metadata:type="Boolean">false</data:FalseValue>
  <data:IntegerValue metadata:type="SByte">-128</data:IntegerValue>
  <data:DoubleValue metadata:type="Double"
       >3.1415926535897931</data:DoubleValue>
  <data:SingleValue metadata:type="Double">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 MAY include a `metadata:type` attribute to specify the type of the primitive- or complex-typed instance.

For example, the following element within an `metadata:properties` element represents the “Rating” field with an integer value of 4:

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

The `data:[PropertyName]` element MAY include a `metadata:null` attribute to specify that the primitive- or complex-typed instance has the `null` value.

For example:

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

7.3.1 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. For example, the following would represent the string value “`CEO`” for the `Title` property of an entity:

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

The following would represent the combined enumeration values `Yellow` and `Solid` for the `Pattern` property of an entity:

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

7.3.2 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. It MAY include a `metadata:type` attribute to specify the type.

For example, the complex typed property “`Name`”, with properties “`FirstName`” and “`LastName`” would be respresented as:

```
<data:Name metadata:type="MyModel.FullName">
  <data:FirstName>Julie</data:FirstName>
  <data:LastName>Swansworth</data:LastName>
</data:Name>
```

7.3.3 Primitive and Enumeration Collection Property

For properties that represent a collection of primitive or enumeration values, the `data:[PropertyName]` element may include a `metadata:type` attribute with a value of `"Collection([PrimitiveTypeName])"`. The content of the element consists of nested child elements named “`element`” in the OData Metadata Namespace for each value in the collection.

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

`<metadata:element>` elements MUST NOT contain the `metadata:null="true"` attribute value.

Example:

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

7.3.4 Complex Collection Property

For properties that represent a collection of complex types, the `data:[PropertyName]` element may include a `metadata:type` attribute with a value of `"Collection([ComplexTypeName])"` attribute.

The content of the element consists of nested child elements named "`element`", in the OData Metadata Namespace, for each complex typed value in the collection.

The `<metadata:element>` element representing the instance may include a `metadata:type` attribute to specify the type of the individual element. The value of each complex-typed `<metadata:element>` follows the syntax for complex-typed properties.

`metadata:element` elements MUST NOT be empty and MUST NOT contain the `metadata:null="true"` attribute.

Example:

```
<data:PhoneNumbers metadata:type="Collection(Person.PhoneNumber)">
  <metadata:element>
    <data:Number>425-555-1212</data:Number>
    <data:PhoneType>Home</data:PhoneType>
  </metadata:element>
  <metadata:element metadata:type="Person.CellPhoneNumber">
    <data:Number>425-555-0178</data:Number>
    <data:PhoneType>Cell</data:PhoneType>
    <data:CellCarrier>Sprint</data:CellCarrier>
  </metadata:element>
</data:PhoneNumbers>
```

7.3.5 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).

For example, the following represents an empty apartment number:

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

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

7.3.6 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 namespace- or alias-qualified type of the property. If the type is defined in a different metadata document than specified by the current metadata URL, 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 example, the following specifies that the Age property is a 32-bit integer with the value 25:

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

8 Navigation Property

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

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 for products related to a category:

<atom:link

```
<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:metadata`, and `title` are to be used as described in the following subsections.

8.1.1.1 Attribute `rel`

The `rel` attribute MUST be present and is made up of 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.

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:metadata`

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

For details on the metadata URL, see **[Error! Reference source not found.]**.

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.

Example for products related to a category:

Example for 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`, `metadata:metadata`, and `title` are to be used as described in the following subsections.

8.2.1.1 Attribute `rel`

The `rel` attribute MUST be present. The value MUST be made up of the string `"http://docs.oasis-open.org/odata/ns/relatedlinks/"` 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.

8.2.1.2 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 set 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:

```
<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 Inserts

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 for inserting a new order with order details and a new customer:

```
<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/Details"
        type="application/atom+xml;type=feed"
        title="Details" href="Orders(11643)/Details">
    <metadata:inline>
      <feed>
        <entry>
          ...
          <content type="application/xml">
            <metadata:properties>
              <data:ProductID metadata:type="Int32">28</data:ProductID>
              ...
            </metadata:properties>
          </content>
        </entry>
        <entry>
          ...
          <content type="application/xml">
            <metadata:properties>
              <data:ProductID metadata:type="Int32">39</data:ProductID>
              ...
            </metadata:properties>
          </content>
        </entry>
      ...
      </feed>
    </metadata:inline>
  </link>
```

```
  <content type="application/xml">
    <metadata:properties>
      <data:OrderID metadata:type="Int32">11643</data:ID>
      <data:CustomerID>ANEWONE</data:CustomerID>
      <data:EmployeeID metadata:type="Int32">6</data:EmployeeID>
      ...
    </metadata:properties>
  </content>
</entry>
```

8.5 Bind Operations

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 for assigning a product to an existing category with an update request:

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

9 Stream Property

9.1 Element atom:link

An entity 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 named stream property of an entity.

For example, a stream property named “Photo” could be represented by an atom:link element as a child of the atom:entry element as follows:

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

A stream property named “Photo” could be edited through an atom:link element as a child of the atom:entry element as follows:

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

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 specifies the meda-type of the stream property.

9.1.4 Attribute metadata:etag

The metadata:etag attribute specifies an etag value that can be used in an if-match header to conditionally write to the stream property as described in **[Error! Reference source not found.]**.

9.1.5 Attribute title

The title attribute provides human-readable, possibly language-dependent, and not necessarily unique information about the link. It has no implied semantics in OData.

10 Media Entity

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

```
<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 URI that can be used to write the stream. This URI 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.

11.1 Single Scalar Value

For example, a request for the first name of a given customer may return the following payloads:

```
<value xmlns="http://docs.oasis-open.org/odata/ns/metadata">CEO</value>
```

or

```
<metadata:value xmlns:metadata=="http://docs.oasis-open.org/odata/ns/metadata"
                metadata:null="true" />
```

Similarly, the following payload represents a full name:

```
<metadata:value metadata:type="HumanResources.Address"
          xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
          xmlns="http://docs.oasis-open.org/odata/ns/data">
   <FirstName>Julie</FirstName>
   <LastName>Swansworth</LastName>
</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:metadata`

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

For more information on the metadata URL, see **[Error! Reference source not found.].**

11.1.1.2 Attribute `metadata:metadata-etag`

The `metadata:value` 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 **[Error! Reference source not found.]**.

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.

For example, the following payload represents a collection of phone numbers.

```
<metadata:value 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>
```

Similarly, the following payload represents a collection of full names.

```
<metadata:value xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata"
                xmlns="http://docs.oasis-open.org/odata/ns/data">
  <metadata:element metadata:type="HumanResources.FullName">
    <FirstName>Julie</FirstName>
    <LastName>Swansworth</LastName>
  </metadata:element>
  <metadata:element metadata:type="HumanResources.FullName">
    <FirstName>Mark</FirstName>
    <LastName>Swansworth</LastName>
  </metadata:element>
</metadata:value>
```

11.2.1 Element `metadata:value`

A Collection of scalar values is represented as a single `metadata:value` root element that contains a `<metadata:element>` child element for each member 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:type`

The attribute `metadata:type` MUST be present and specify the namespace- or alias-qualified collection type.

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 types with a `metadata:type` attribute on the `<metadata:element>` element.

12 Collection of Entities

Collections of entities are represented in Atom as an Atom Feed.

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.

12.1.1 Attribute metadata:metadata

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

For more information on the metadata URL, see **[Error! Reference source not found.]**.

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 **[Error! Reference source not found.]**.

12.1.3 Element atom:id

OData does not add any conventions or semantics beyond **[RFC4287]** to the atom:id element for feeds.

12.1.4 Element metadata:count

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

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:metadata="http://docs.oasis-open.org/odata/ns/metadata" >
  <metadata:count>42</metadata:count>
  ...
</feed>
```

12.1.5 Element atom:link

Atom requires that feeds contain a “self link”. The atom:feed element MAY contain a *next link* to indicate the presence of additional entries that belong to the feed. 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.

Note that the actual set of entries contained within the atom:feed MAY be a subset of those retrieved using the self link, for example, if filtering has been applied.

12.1.5.1 Attribute rel

A self link is represented as an atom:link with a rel="self" attribute and an href that can be used to retrieve the feed from which the current entries are taken. If the feed represents a set of related entities

(addressed with a request URL ending in a to-many navigation property, or an inlined feed requested with $expand), the self link MUST identify the specific feed of related entities.

A self link is represented as an `atom:link` with a `rel="self"` attribute and an `href` that can be used to retrieve the feed from which the current entries are taken. If the feed represents a set of related entities (addressed with a request URL ending in a to-many navigation property, or an inlined feed requested with $expand), the self link MUST identify the specific feed of related entities.

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.

For example, the following `atom:link` element within an `atom:feed` element indicates that additional results can be returned by following the specified `href`:

```
<atom:link rel="next"
           href="http://myservice/customers?$skiptoken=1237"/>
```

The contents of the `href` 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.

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.

For example, the following `atom:link` element within an `atom:feed` element indicates that changes may be retrieved by following the specified `href`:

```
<atom:link rel=" http://docs.oasis-open.org/odata/ns/delta"
           href="http://myservice/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 Resource Reference

A resource reference is a reference to an entity or a property of an entity. A resource reference referring to an entity is called an *entity reference*.

An entity reference MAY take the place of an entity instance in an Atom payload, based on the client request.

For example, the following shows an entity reference to order 10643:

```
<metadata:ref ref="http://services.odata.org/OData/OData.svc/Orders(10643)" />
```

13.1 Element `metadata:ref`

A reference to an entity or one of its properties is represented in Atom using a `metadata:ref` element. The `ref` attribute MUST be present and used as described in the following subsection.

13.1.1 Attribute `ref`

The `ref` attribute MUST be present. For entities the `ref` attribute MUST be the `atom:id` of the referenced entity, for entity properties it MUST be the `atom:id` of the entity followed by the resource path segment identifying the property.

14 Delta Response

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 set specified by the value of `metadata:metadata` MUST include the `metadata:metadata` attribute in the `atom:id` element to specify the 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.

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.

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 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, the returned count is the count of added, changed, or deleted entities. `$count` and `$inlinecount` returned from a delta link do not include added or deleted links.

The following example shows the following ordered changes:

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

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<feed xml:base="http://northwinddelta.cloudapp.net/DeltaService.svc/"
      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" >
 <title type="text">Customers</title>
 <id>http://DeltaService.svc/Customers</id>
 <updated>2011-02-16T01:00:25Z</updated>
 <link rel="self" title="Customers" href="Customers" />
 <entry>
  <id>http://DeltaService/Customers('BOTTM')</id>
  <title type="text" />
  <updated>2011-02-16T01:00:25Z</updated>
  <author><name /></author>
  <link rel="edit" title="Customer" href="Customers('BOTTM')"/>
  <category term="NorthwindModel.Customer" scheme="http://odata.org/scheme"/>
  <content type="application/xml">
  <metadata:properties>
   <d:ContactName>Susan Halvenstern</d:ContactName>
  </metadata:properties>
  </content>
 </entry>
 <metadata:deleted-link-entry
   source="http://DeltaService/Customers('ALFKI')"
```



```
   relationship="Orders"
   target="http://DeltaService/Orders(10643)"
   when="2011-02-16T01:00:25Z"/>
 <metadata:link-entry
   source="http://DeltaService/Customers('BOTTM')"
   relationship="Orders"
   target="http://DeltaService/Orders(10645)"
   when="2011-02-16T01:00:25Z"/>
 <entry metadata:metadata="Orders/@Element">
  <id>http://DeltaService/Orders('10643')</id>
  <title type="text" />
  <updated>2011-02-16T01:00:25Z</updated>
  <author><name/></author>
  <link rel="edit" title="Order" href="Orders('10643')" />
  <category term="NorthwindModel.Order" scheme="http://odata.org/scheme" />
  <content type="application/xml">
   <metadata:properties>
    <d:ShipName>Bottom-Dollar Markets</d:ShipName>
    <d:ShipAddress>23 Tsawassen Blvd.</d:ShipAddress>
    <d:ShipCity>Tsawassen</d:ShipCity>
    <d:ShipRegion>BC</d:ShipRegion>
    <d:ShipPostalCode>T2F 8M4</d:ShipPostalCode>
    <d:ShipCountry>Canada</d:ShipCountry>
   </metadata:properties>
  </content>
 </entry>
 <at:deleted-entry
    ref="http://DeltaService/Customers('ANTON')"
    when="2011-02-16T01:00:30Z"
    metadata:reason="deleted"/>
 <link
    rel="http://odata.org/deltaLink"
    href="http://DeltaService/Customers?$expand=orders&$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 metadata URL of the feed MUST include the `metadata:metadata` attribute in the `atom:entry` element to specify the metadata URL of the entity (in this metadata URL specifies the set, the entity belongs to). This metadata URL MAY be absolute or relative to the metadata URL of the feed.

14.2 Deleted Entity

14.2.1 Element `atom-tombstone:deleted-entry`

A deleted entitiy 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.

14.2.1.2 Attribute `when`

As defined in **[RFC6721],** the `when` attribute MUST be present The value of the `when` attribute MUST specify the time at which the entity was deleted. The value may be the empty string if the service is unable to determine the time at which the deletion occurred.

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 Link

14.3.1 Element `metadata:link-entry`

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

A Delta Response MUST contain a `metadata:link-entry` for each added link that corresponds to a `$expand` path in the initial request.

The `source`, `relationship, and target` attribute MUST be present, the `when` attribute MAY 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.

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.

14.3.1.4 Attribute `when`

The `when` attribute MAY be present and specify the time at which the link was created. The attribute MAY be the empty string if the service is unable to determine the time at which the creation occurred.

14.4 Deleted Link

14.4.1 Element `metadata:deleted-link-entry`

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

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

The service MAY return a `metadata:deleted-link-entry` where one of the entitites has also been deleted, or where there is a subsequent `metadata:link-entry` 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, the `when` attribute MAY 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.

14.4.1.2 Attribute `relationship`

The `target` attribute MUST be present and specify the `atom:id` of the related entity.

14.4.1.3 Attribute `target`

The `target` attribute MUST be present and specify the `atom:id` of the related entity.

14.4.1.4 Attribute `when`

The `when` attribute MAY be present and specify the time at which the link was created. The attribute MAY be the empty string if the service is unable to determine the time at which the creation occurred

15 Function

Zero or more functions MAY be bindable to a feed or entry.

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

Example:

```
<metadata:function
  metadata="#DemoService.TopProducts"
  target="http://host/service/Categories(0)/TopProducts"
  title="Best-Selling Products"
/>
```

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 or entity on which the function exists.

15.1.1 Attribute `metadata`

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

A function may have multiple overloads with different parameters. 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 a comma-separated ordered list of parameter type names, enclosed in parenthesis. For example, `#Schema.Function(Schema.Product,Edm.String)`.

Example:

```
<metadata:function
  metadata="#DemoService.TopProducts(DemoModel.Category,Edm.String)"
  target="http://host/service/Categories(0)/TopProducts"
  title="Best-selling products"
/>
```

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.

15.1.2 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 Action

Zero or more actions may be bindable to a feed or entry.

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

Example:

```
<metadata:action
  metadata="#DemoService.OrderProduct"
  target="http://services.odata.org/OData/OData.svc/Products(1)/OrderProduct"
  title="Order"
/>
```

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 feed or entity on which the action exists.

16.1.1 Attribute `metadata`

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

This function element name combined with the binding parameter type MUST be unique within the entity container.

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 `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 Parameters

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 default value specified in the service metadata, see **[OData-CSDL]**.

Example:

```
<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 default value from $metadata-->
<parameters>
```

18 Instance Annotations

Annotations MAY be applied to an instance of a feed, entity, property, complex scalar value, function, action, 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- 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 Values

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

18.2.1 Primitive Values

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.

For example; the following annotates the "Phone" property with a string value of "Home" for the “PhoneNumberType” annotation term.

```
<metadata:properties>
  <data:CustomerID>ALFKI</data:CustomerID>
  <data:ContactName> Alfreds Futterkiste </data:ContactName>
  <data:Phone>030-0074321</data:Phone>
```

```
  <metadata:annotation term="com.contoso.PersonalInfo.PhoneNumberType"
                       target="Phone">Home</metadata:annotation>
</metadata:properties>
```

18.2.2 Collection Values

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 Complex Scalar Values.

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

For example, the following annotates the customer instance with two phone numbers.

```
<entry>
  <id>Customers(ALFKI)</id>
  <content>
    <metadata:properties>
      <data:CustomerID>ALFKI</data:CustomerID>
      <data:ContactName>Alfreds Futterkiste</data:ContactName>
      <data:Phone>030-0074321</data:Phone>
    </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 Structure Annotations

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.

For example; the following specifies the `StreetAddress`, `City`, `Region`, `Country` and `PostalCode` properties of an `Address` annotation applied to a customer entity:

```
<entry>
  <id>Customers(ALFKI)</id>
  <content>
    <metadata:properties>
      <data:CustomerID>ALFKI</data:CustomerID>
      <data:ContactName>Alfreds Futterkiste</data:ContactName>
      <data:Phone>030-0074321</data:Phone>
    </metadata:properties>
  </content>
  <metadata:annotation term="com.contoso.Locations.Address"
                       type="Locations.Address">
    <data:StreetAddress>Obere Str. 578</data:StreetAddress>
    <data:City>Toronto</data:City>
    <data:Region metadata:null="true"/>
    <data:PostalCode>12209</data:PostalCode>
    <data:Country>Germany</data:Country>
  </metadata:annotation>
<entry>
```

18.3 Instance Annotation Targets

Instance annotations may target model elements represented by a feed, entity, complex scalar value, property, function, action, or error element in 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 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.

18.3.4 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.

18.3.5 Navigation Property

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

18.3.6 Function or Action

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

18.3.7 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, servers 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.

For example:

```
<error xmlns="http://docs.oasis-open.org/odata/ns/metadata">
  <code>501</code>
  <message>Functionality not supported.</message>
  <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:detail 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 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

The exception to this are the constructs defined in Delta Response, which are only required for clients that request changes.

A conforming OData service MUST comply with one of the conformance levels defined in **[Error! Reference source not found.]**.

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

- MUST support the `application/atom+xml`, `application/xml` and `application/atomsvc+xml` media types in the `Accept` header
- SHOULD support the `$format` system query option
- MUST include the next link in feeds containing partial results
- MUST return service documents as Atom service documents
- MUST return XML responses in well formed XML according to this specification
- MUST return well-formed Atom payloads with the following exceptions:
  - The next link MAY be returned at the end of the payload
  - The delta link MAY be returned at the end of the payload
- MUST NOT violate any other aspects of this OData Atom specification

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

- MUST be prepared to receive all data types
  - defined in this specification defined in **[OData-CSDL]** (client)
  - exposed by the service (service)
- MUST be prepared to receive custom annotations
- MUST be prepared to receive additional constructs not defined in this version of the specification

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