Network Working Group
Internet-Draft
Expires: April 14, 2006

J. Gregorio, Ed.
BitWorking, Inc
B. de hOra, Ed.
Propylon Ltd.
October 11, 2005

The Atom Publishing Protocol
draft-ietf-atompub-protocol-05.txt

Status of this Memo

Copyright Notice

Abstract

This memo presents a protocol for using XML (Extensible Markup Language) and HTTP (HyperText Transport Protocol) to edit content.

The Atom Publishing Protocol (APP) is an application-level protocol for publishing and editing Web resources. The protocol at its core is the HTTP transport of Atom-formatted representations. The Atom format is documented in the Atom Syndication Format

(draft-ietf-atompub-format-11.txt).

Editorial Note

To provide feedback on this Internet-Draft, join the atom-protocol mailing list (http://www.imc.org/atom-protocol/index.html) [1].

Table of Contents

1. Introduction

The Atom Publishing Protocol is an application-level protocol for publishing and editing Web resources using HTTP [RFC2616] and XML 1.0 [W3C.REC-xml-20040204].

2. XML Namespace and Language

The XML Namespaces URI [W3C.REC-xml-names-19990114] for the XML data format described in this specification is: http://purl.org/atom/app#

XML elements defined by this specification MAY have an xml:lang attribute, whose content indicates the natural language for the element (and its descendents). The language context is only significant for elements and attributes declared to be "Language-Sensitive" by this specification. Requirements regarding the content and interpretation of xml:lang are specified in [W3C.REC-xml-20040204], Section 2.12.

3. Notational Conventions

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].

Some sections of this specification are illustrated with fragments of a non-normative RELAX NG Compact schema [RNC]. However, the text of this specification provides the definition of conformance.

This specification uses the namespace prefix "app:" for the Namespace URI identified in Section 2 above. It uses the namespace prefix "atom:" for the Namespace URI identified in [AtomFormat]. Note that choices of namespace prefix are arbitrary and not semantically significant.

4. Terminology

For convenience, this protocol may be referred to as "Atom Protocol" or "APP". This specification uses both internally.

URI/IRI - A Uniform Resource Identifier and Internationalized Resource Identifier, respectively. These terms (and the distinction between them) are defined in [RFC3986] and [RFC3987].

Resource - A network data object or service that can be identified by a URI, as defined in [RFC2616].

Representation - An entity included with a request or response as defined in [RFC2616].

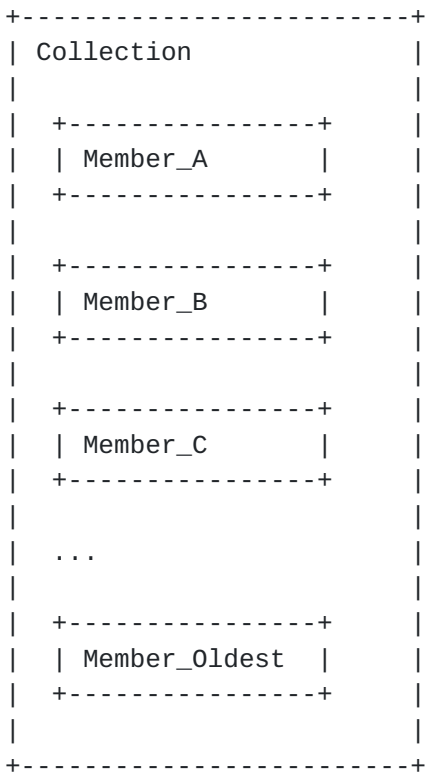
5. The Atom Publishing Protocol Model

The Atom Publishing Protocol is a subset of HTTP that is used to edit resources on the web. The APP operates on collections of Web resources. Collections are HTTP resources, as are the members of the collection. Both Collections and collection member resources support the same basic interactions. The patterns of interaction are based on the common HTTP verbs.

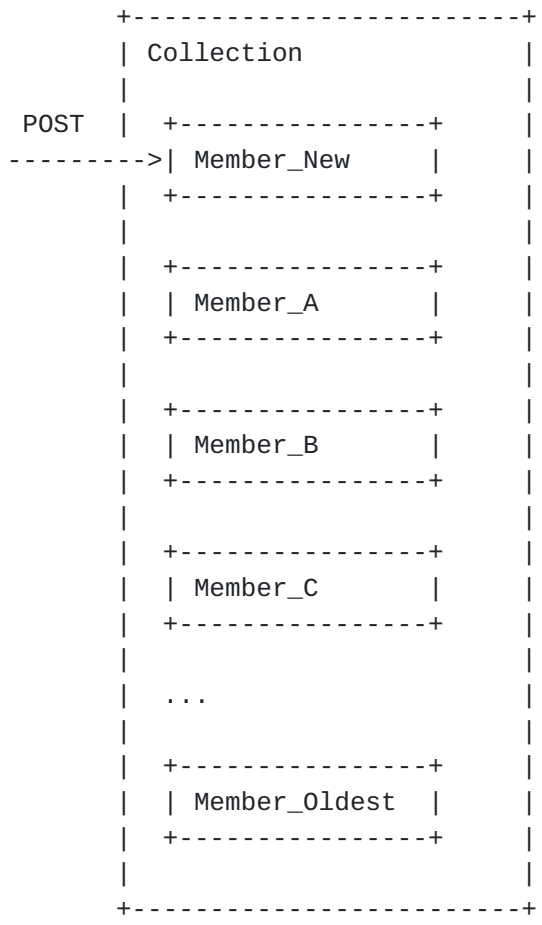
- GET is used to retrieve a representation of a resource or perform a read-only query.
- POST is used to create a new, dynamically-named resource, or to provide a block of data to a data-handling process.
- PUT is used to update a known resource.
- DELETE is used to remove a resource.

5.1 Collections

The APP groups resources into "Collections", which are analogous to folders or directories found in a file system. In the figure we have member resources in a collection.



To add a new member to a collection an appropriate representation is POSTed to the URI of the collection resource. Here we show it being added to the beginnng of the list. The ordering of the members of collections is in terms of the time at which each resource was last updated, which includes the act of creating the resource. The ordering of collection members is covered in more detail in Section 8 and Section 11.



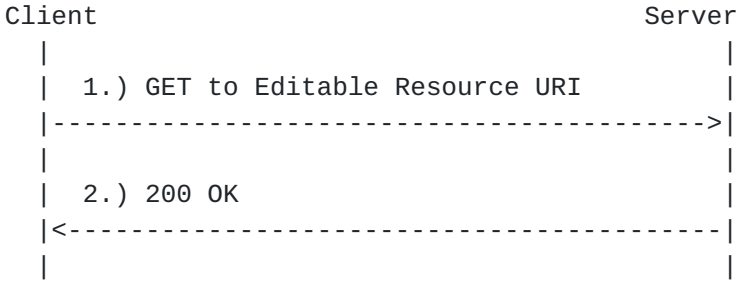
You'll note that up until now we haven't said what kinds of representations we are expecting at each of the resources. There are two kinds of collections, Entry and Generic. In Entry Collections all the members MUST have representations as Atom Entries. For further restrictions on Entry Collection see Section 9 The other type of collection is a Generic Collection. Generic Collections make no restriction on the representations of their member resources.

5.2 Editable Resources

All the members of a collection are Editable Resources. An Editable resource is a resource whose available HTTP methods can be used to retrieve, update and delete it.

5.2.1 Read

To retrieve a representation of the resource, you send a GET to the URI of the Editable Resource. Remember that for members of Entry Collections, the served representation will be an Atom Entry.

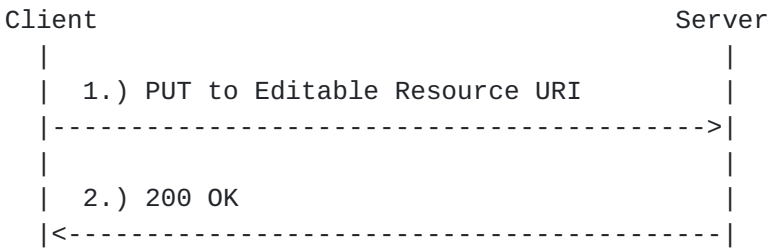


1. The client sends a GET request to the member's URI.

2. The server responds with the representation of the resource.

5.2.2 Update

To update an Editable Resource the client will PUT an updated representation to the URI of the resource.

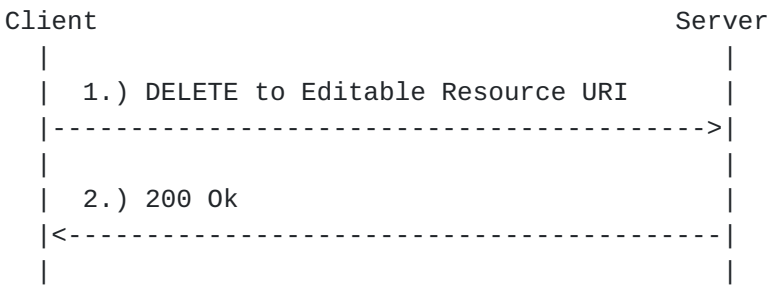


1. The client PUTs an updated representation to the member's URI.

2. The server MAY respond with an updated representation of the member's new state.

5.2.3 Delete

An Editable Resource is deleted by sending it DELETE. Note that this also removes it from all the collections that it belonged to.

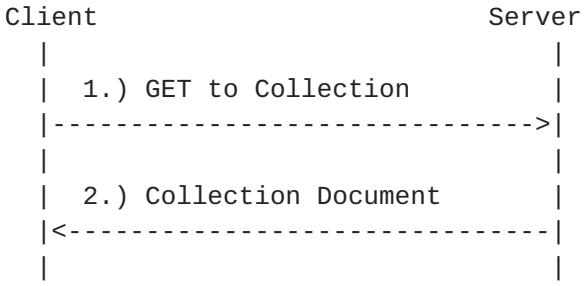


1. The client sends a DELETE request to the member's URI.

2. The server responds with successful status code.

5.3 Capabilities Discovery

Each collection resource responds to GET and can return a Collection Document as it's representation. The Collection Document enumerates the capabilities of each collection and the format is described in Section 6.2.



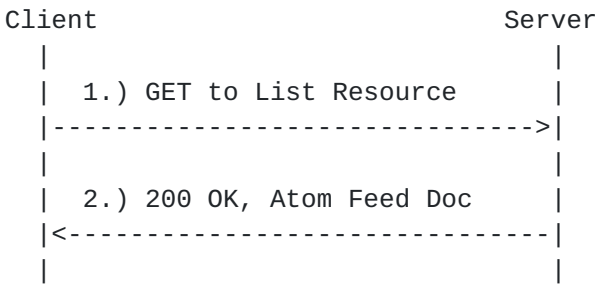
1. The client sends a GET request to the Collection Resource.

2. The server responds with a Collection Document containing a description of the capabilities of the collection. The content of this document can vary based on aspects of the client request, including, but not limited to, authentication credentials.

5.4 Listing

Clients can request a listing of the Collection's membership. Listing the Editable Resources that are members of a collection is done using one of the List Resources in the Introspection Document, utilizing the 'app:uri-template' element. The List Resource returns Atom Feed Documents with one Atom Entry for each member resource that match the selection criteria. This is true whether the collection is an Entry Collection or a Generic Collection. If an Entry Collection is being interrogated, the entries returned by a list resource SHOULD

NOT to be considered complete representations of the member resources. See Section 11 and Section 12 for more details on the extensions and constraints found on the entries returned from List Resources.



1. The client sends a GET request to the Collection's URI.

2. The server responds with an Atom Feed Document containing a full or partial listing of the Collection's membership.

5.5 Success and Failure

HTTP defines different classes of response, which are used by the Atom Protocol. HTTP status codes of the form 2xx signal that a request was successful. HTTP status codes of the form 4xx or 5xx signal that an error has occurred, and the request has failed. Consult the HTTP specification [RFC2616] for more detailed definitions of each status code.

6. Atom Publishing Protocol Documents

This specification describes two kinds of Atom Publishing Protocol Documents: Atom Collections Documents and Atom Introspection Documents.

An Atom Collection Document is a representation of an Atom collection, including metadata about the collection, and some or all of the members associated with it. Its root is the app:collection element.

An Atom Introspection Document represents one or more workspaces, which describe server-defined groupings of collections. Its root is the app:service element.

namespace app = "..." start = appCollection | appIntrospection

Both kinds of Atom Publishing Protocol Documents are specified in terms of the XML Information Set, serialised as XML 1.0 ([W3C.REC-xml-20040204]). Atom Publishing Protocol Documents MUST be well-formed XML. This specification does not define a DTD for Atom Protocol, and hence does not require them to be valid (in the sense used by XML).

Atom Collection Documents are identified with the "application/atomcoll+xml" media type.

Atom Introspection Documents are identified with the "application/atomserv+xml" media type.

Atom allows the use of IRIs [RFC3987], as well as URIs [RFC3986]. Every URI is an IRI, so any URI can be used where an IRI is needed. While IRIs must, for many protocols, be mapped to URIs prior to dereferencing, they MUST NOT be so mapped for comparison when used in atom:id. Section 3.1 of [RFC3987] describes how to map an IRI to a URI when necessary.

6.1 Use of xml:base xml:lang

Any element defined by this specification MAY have an xml:base attribute [W3C.REC-xmlbase-20010627]. When xml:base is used in an Atom Publishing Protocol Document, it serves the function described in section 5.1.1 of [RFC3986], establishing the base URI (or IRI) for resolving any relative references found within the effective scope of the xml:base attribute.

Any element defined by this specification MAY have an xml:lang attribute, whose content indicates the natural language for the

element and its descendents. The language context is only significant for elements and attributes declared to be "Language-Sensitive" by this specification. Requirements regarding the content and interpretation of xml:lang are specified in XML 1.0 ([W3C.REC-xml-20040204]), Section 2.12.

```
 appCommonAttributes =
      attribute xml:base { atomUri }?,
      attribute xml:lang { atomLanguageTag }?,
     undefinedAttribute*
```

6.2 Collection Documents

The Collection Document describes the capabilities of a Collection, the types of Entries that it will support, the URI Templates it supports.

The Collection Document has the media-type 'application/atomcoll+xml' (see Section 15).

Here's an example document:

```
<?xml version="1.0" encoding='utf-8'?>
<app:collection xmlns:app="http://purl.org/atom/app#">
  <app:member-type>entry</pub:member-type>
  <app:uri-template>http://example.org/{index}</pub:uri-template>
  <app:uri-template>http://example.org/{daterange}</pub:uri-template>
</app:collection>
```

This example says the Collection contains Atom Entry documents, and that there are two means of selecting entries using what are called 'URI Templates'; one based on the collection's order, and another based on dates. See Section 11.1 for more about URI Templates.

6.2.1 Element Definitions

6.2.1.1 The 'app:collection' Element

The app:collection is the document element of a Collection Document.

```
appCollection =
   element app:collection {
      appCommonAttributes,
      ( appMemberType+
        appSearchTemplate
        & anyElement* )
```

```
   }
```

This specification defines two child elements for app:collection:

- app:member-type: any number of elements listing the types of Entries that the Collection may contain.

- app:uri-template: any number of URI Templates for a List Resource (See Section 11).

6.2.1.2 The 'app:member-type' Element

The app:member-type element contains information elements about the types of Entries that the Collection may contain.

```
appMember =
   element app:member-type {
         appCommonAttributes,
         appTypeValue
   }
```

The element content of an app:member-type MUST be a string that is non-empty, and matches either the "isegment-nz-nc" or the "IRI" production in [RFC3987]. Note that use of a relative reference other than a simple name is not allowed. If a name is given, implementations MUST consider the link relation type to be equivalent to the same name registered within the IANA Registry of Member Types (Section 15), and thus the IRI that would be obtained by appending the value of the rel attribute to the string "http://www.iana.org/assignments/entrytype/".

The content of an app:member-type specifies constraints on the Entries that may appear in the Collection. The app:collection element MAY have multiple app:member-type elements. An Entry POSTed to a Collection MUST meet the constraints of at least one of the app:member-type constraints. It MAY meet more than one, but the minimum requirement is at least one.

This specification defines two initial values for app:member-type IANA registry:

- "entry" - The Collection is an Entry Collection as defined in Section 9.

- "generic" - The Collection is a Generic Collection as defined in Section 10.

6.2.1.3 The 'app:uri-template' Element

The element content of an app:uri-template is a URI Template for a List Resource (See Section 11). Every List resource, whose URI is determined by filling in the parameters in a URI Template, MUST return an Atom feed document as its representation. This Atom feed document MUST NOT contain entries which do not match the selection criteria.

6.3 Introspection Documents

In order for authoring to commence, a client must first discover the capabilities and locations of collections offered.

The Introspection Document describes "workspaces", which are server-defined groupings of collections. There is no requirement that servers support multiple workspaces, and a collection may appear in more than one workspace.

The Introspection Document has the media-type 'application/atomserv+xml', see Section 15

Here's an example document:

```
<?xml version="1.0" encoding='utf-8'?>
<app:service xmlns:app="http://purl.org/atom/app#">
  <app:workspace title="Main Site" >
      <app:collection contents="entries"
          title="My Blog Entries"
          href="http://example.org/reilly/feed" />
      <app:collection contents="generic"
          title="Documents"
          href="http://example.org/reilly/pic" />
  </app:workspace>
  <app:workspace title="Side Bar Blog">
      <app:collection contents="entries" title="Entries"
          href="http://example.org/reilly/feed" />
      <app:collection contents="http://example.net/booklist"
          title="Books"
          href="http://example.org/reilly/books" />
  </app:workspace>
</app:service>
```

This example says there are two workspaces, each consisting of two

collections. The first workspace is called 'Mail', and has two collections, called 'My Blog Entries' and 'Documents' whose locations are 'http://example.org/reilly/feed' and 'http://example.org/reilly/pic'. 'My Blog Entries' contains Atom Entries and 'Documents' contains Generic Entries. The second workspace is called 'Side Bar Blog' and also has two collections, called 'Entries' and 'Books' whose locations are 'http://example.org/reilly/feed' and 'http://example.org/reilly/booklist'. 'Entries' contains Atom Entries and 'Books' contains Generic Entries (since its contents attribute is not present you MUST assume it is a Generic Collection).

6.3.1 Element Definitions

6.3.1.1 The 'app:service' Element

The "app:service" element is the document element of a Introspection Document, acting as a container for service data associated with one or more workspaces. An app:service elements MAY contain any number of app:workspace elements.

```
appService =
   element app:service {
      appCommonAttributes,
      ( appWorkspace*
        & anyElement* )
   }
```

6.3.1.2 The 'app:workspace' Element

The 'workspace' element contains information elements about the collections of resources available for editing. The app:workspace elements MAY contain any number of app:collection elements.

```
appWorkspace =
   element app:workspace {
      appCommonAttributes,
      attribute title { text },
      ( appCollection*
        & anyElement* )
   }
```

6.3.1.2.1 The 'title' Attribute

The app:workspace element MUST contain a 'title' attribute, which conveys a human-readable name for the workspace. This attribute is Language-Sensitive.

6.3.1.3 The 'app:collection' Element

The 'app:collection' element describes collections and their member resources.

```
appCollection =
   element app:collection {
      appCommonAttributes,
      attribute title { text },
      attribute href { text },
      attribute contents { text },
      anyElement*
   }
```

6.3.1.3.1 The 'title' Attribute

The app:collection element MUST contain a 'title' attribute, whose value conveys a human-readable name for the workspace. This attribute is Language-Sensitive.

6.3.1.3.2 The 'href' Attribute

The app:collection element MUST contain an 'href' attribute, whose value conveys the IRI of the collection.

6.3.1.3.3 The 'contents' Attribute

The app:collection element MAY contain a 'contents' attribute. The 'contents' attribute conveys the nature of a collection's member resources. This specification defines two initial values for the 'contents' attribute:

- 'entry': A value of 'entry' for the contents attribute indicates that the Collection is an Entry Collection (Section 9).

- 'generic': A value of 'generic' for the contents attribute indicates that the Collection is a Generic Collection (Section 10).

If the attribute is not present, its value MUST be considered to be

'generic'.

7. Introspection Resource

To retrieve an Introspection Document, the client sends a GET request to its URI.

```
GET /service-desc HTTP/1.1
Host: example.org
User-Agent: Cosimo/1.0
Accept: application/atomserv+xml
```

The server responds to a GET request by returning an Introspection Document in the message body.

```
HTTP/1.1 200 OK
Date: Mon, 21 Mar 2005 19:20:19 GMT
Server: CountBasic/2.0
Last-Modified: Mon, 21 Mar 2005 19:17:26 GMT
ETag: "4c083-268-423f1dc6"
Content-Length: nnnn
Content-Type: application/atomserv+xml

<?xml version="1.0" encoding='utf-8'?>
<app:service xmlns:app="http://purl.org/atom/app#">
    ...
</app:service>
```

7.1 Discovery

[[anchor18: Add in desc of an HTML link element that points to the Introspection Resource, or add it to the autodisco draft]]

8. Collection Resources

An Atom Collection is a set of related resources. All members of a collection have an "app:updated" property, and the Collection is considered to be ordered by this property.

This specification defines two HTTP methods for use with collection resources: GET and POST.

8.1 GET

A GET to a Collection Resource returns a Collection Document, outlining the Collection. Collection Documents are described in Section 6.2.

8.2 POST

In addition to GET, a Collection Resource also accepts POST requests. The client POSTs a representation of the desired resource to the Collection Resource. Note that some collections may impose constraints on the media-types that are created in a Collection and MAY generate a response with a status code of 415 ("Unsupported Media Type").

In the case of a successful creation, the status code MUST be 201 ("Created").

Every successful POST MUST return a Location: header with the URI of the newly created resource.

Here's an example. Below, the client requests to create a resource in a Collection:

```
POST /edit HTTP/1.1
Host: example.org
User-Agent: Cosimo/1.0
Accept: application/atom+xml
Content-Type: application/atom+xml
Content-Length: 601

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
  <atom:title>Mars Attacks!</atom:title>
  <atom:summary type="html">
    Why cant we all just... get along?
  </atom:summary>
  <atom:author>
    <atom:name>The President</atom:name>
    <atom:uri>http://www.example.org/blog</atom:uri>
  </atom:author>
  <atom:content type="html" xml:lang="en"
      xml:base="http://www.example.org/blog/">
    <p>
    Why can't we...work out our differences?
    Why can't we...work things out?
    Little people...why can't we all just...get along?
    </p>
  </atom:content>
</atom:entry>
```

The resource is created by sending an Atom Entry as the entity body.

Assuming the server created the resource successfully, it sends back a 201 Created response with a Location: header that contains the IRI of the newly created member as an Editable Resource.

```
HTTP/1.1 201 Created
Date: Fri, 7 Oct 2005 17:17:11 GMT
Content-Length: 663
Content-Type: application/atom+xml; charset="utf-8"
Location: http://example.org/edit/first-post.atom
```

8.3 Title: Header

The POST to a Collection Resource MAY contain a Title: header that indicates the clients suggested name for the resource. The server MAY ignore the Title: header or modify the requested name to suit local conventions.

```
     Title     = "Title" ":" [text]
```

9. Entry Collections

Entry Collections are Collections that restrict their membership to Atom entries.

9.1 Editing Entry Resources

Atom entries are edited by sending HTTP requests to an individual entry's URI. Servers can determine the processing necessary to interpret a request by examining the request's HTTP method and 'Content-Type' header.

Processing Client Requests

	GET	PUT	DELETE	POST
No Body	Read	x	Delete	x
Atom Body	x	Update	x	x

9.2 Role of Atom Entry Elements During Editing

The elements of an Atom Entry Document are either a 'Writable Element' or a 'Round Trip Element'.

Writable Element - An element of an Atom Entry whose value is editable by the client and not enforced by the server.

Round Trip Element - An element of an Atom Entry whose value is enforced by the server and not editable by the client.

That categorization will determine the elements' disposition during editing.

Atom Entry Element	Property
atom:author	Writable
atom:category	Writable
atom:content	Writable
atom:contributor	Writable
atom:id	Round Trip
atom:link	Writable
atom:published	Writable
atom:source	Writable
atom:summary	Writable
atom:title	Writable
atom:updated	Round Trip

Table 2

10. Generic Collections

Generic Collections are Collections that do not have uniform restrictions on the representations of the member resources.

10.1 Editing Generic Resources

Member resources are edited by sending HTTP requests to an individual resource's URI. Servers can determine the processing necessary to interpret a request by examining the request's HTTP method and 'Content-Type' header.

Processing Client Requests

	GET	PUT	DELETE	POST
No Body	Read	x	Delete	x
Any Body	x	Update	x	x

When a List resource returns an Atom Feed enumerating the contents of a Generic Collection, all the Entries MUST have an atom:content element with a 'src' attribute.

10.2 Title: Header

The POST to a Generic Collection Resource MAY contain a Title: header that indicates the clients suggested title for the resource. The server MAY ignore the Title: header or modify the requested title to suit local conventions.

```
Title     = "Title" ":" [text]
```

11. List Resources

List resources are resources which are identified by URI templates indicating selection criteria. They can be used where clients require fine control over the range or size of a server's response. A list resource MUST return an Atom feed document as its representation. The entries in the returned document MUST be ordered by their 'atom:updated' property, with the most recently updated entries coming first in the document order. Clients MUST NOT assume that the entry returned in the feed is a full representation of a member resource. If the entry is an Editable Resource then the client should perform a GET on the member resource before editing.

note: in this section some URIs carry across onto the next line; this is indicated by a '\'

11.1 URI Templates

URI Templates are a mechanism for declaring criteria against a list resource. By itself a URI Template is not a valid URI. Instead there are multiple parameters embedded in the URI and distinguished by closing braces which can be populated and used as selection criteria. The value of each app:uri-template element in a Collection document is a URI Template.

Each URI template has one or more parameters that MUST be substituted with values to construct a valid URI. The substitution MUST ensure that the resulting value is also properly percent-encoded utf-8.

Here are some examples of template URIs and corresponding populated values:

```
http://example.org/blog/edit/{index}
http://example.org/blog/edit/3-9

http://example.org/blog/edit/{index}/foo
http://example.org/blog/edit/0-100/foo

http://example.org/blog/edit/{daterange}
http://example.org/blog/edit/daterange=\
    2003-12-13T18:30:02Z-2003-12-13T18:30:02Z

http://example.org/blog/edit?dr={daterange}/bar/
http://example.org/blog/edit?dr=\
    2003-12-13T18:30:02Z,2003-12-13T18:30:02Z/bar/
```

Note that the parameters MAY appear at any place in the URI template.

11.2 URI Template Parameters

This specification defines two parameters for use in URI Templates:

- index: allows selection into a collection's resources based as though ordered by their 'atom:updated' property.

- daterange: allows selection into a collection's resources based on their 'atom:updated' property

In both cases, the response to the selection request MUST be an Atom Feed where all the entries fall within the requested criteria. The request range is considered a closed set - if an entry matches one end of the range exactly it MUST be included in the response. If no members fall in the requested range, the server MUST respond with an Atom Feed containing no entries.

A Collection Document MUST contain at least two app:uri-template elements - one for the {index} parameter template and the other for the {daterange} parameter template. The two parameters are not mutually exclusive and MAY appear together in a single Template URI.

11.2.1 \{index\} URI template variable

The value of the {index} criterion MUST be a pair of non-negative integer indices separated by a dash character. One or other index MAY omitted, in which case the range is understood as stretching to zero, or infinity.

```
index-specifier = [index] "-" [index]
```

For example, suppose the client is supplied this {index} URI template:

```
http://example.org/blog/edit/{index}
```

If the client wants the first 15 entries in the Collection it would substitute the brace-delimited parameter {index}, with the value 1-15, giving:

```
http://example.org/blog/edit/1-15
```

11.2.2 \{daterange\} URI template variable

A URI Template with the variable 'daterange' allows querying for Atom Entries in a Collection according to their 'atom:updated' property.

The value of the {daterange} criterion should be a pair of ISO formatted dates separated by a dash character; either index may be optionally omitted, in which case the range is understood as stretching to infinity on that end.

```
daterange-specifier = [iso-date] "," [iso-date]
```

The [iso-date] terminal MUST conform to the "date-time" production in [RFC3339]. In addition, an uppercase "T" character MUST be used to separate date and time, and an uppercase "Z" character MUST be present in the absence of a numeric time zone offset.

For example, suppose the client is supplied this {daterange} URI Template:

```
http://example.org/blog/edit/{daterange}
```

If the client wants the entries in the collection between January and February 2006 it would substitute the brace-delimited parameter {daterange} with the desired selection value, giving this URI:

```
http://example.org/blog/edit/2006-01-01T00:00:00Z,\
    2006-02-01T00:00:00Z
```

11.2.3 Other URI Template parameters

Other specifications MAY define new parameters for use in URI templates and declared in the app:uri-template element.

12. Atom Entry Extensions

This specification adds three new values to the Registry of Link Relations.

The value of 'collection' signifies that the IRI in the value of the href is the Collection that this Entry belongs to. Any entry MAY contain a link with a relation of 'collection'.

The value of 'edit' signifies that the IRI in the value of the href attribute identifies the resource that is used to edit the entry. That is, it is the URI of the Entry as an Editable Resource.

The value of 'srcedit' signifies that the IRI in the value of the href attribute identifies the resource that is used to edit the resource pointed to by the 'src' attribute of the atom:content element. That is, it is the IRI of the atom:content@src as an Editable Resource. If a link element with a relation of "srcedit" is not given, then it's value defaults to the "src" attribute of the content element. List Resources for Generic Collections MUST return entries that have 'srcedit' links or MUST have a atom:content@src value.

If the "srcedit" link is present, and it's value is an empty string, then there is no URI that can be treated in the way such a value would be treated.

Clients SHOULD use the "srcedit" value to manipulate the resource within the context of the APP itself. Clients SHOULD prefer the "atom:content@src" value in any other context. For example, if the resource is an image, a client may replace the image data using a PUT on the "srcedit" value, and may even display a preview of the image by fetching the "srcedit" URI. But when creating a public, read-only reference to the same image resource, the client should use the "atom:content@src" value.

13. Securing the Atom Protocol

All instances of publishing Atom entries SHOULD be protected by authentication to prevent posting or editing by unknown sources. Atom servers and clients MUST support one of the following authentication mechanisms, and SHOULD support both.

- HTTP Digest Authentication [RFC2617]
- [@@TBD@@ CGI Authentication ref]

Atom servers and clients MAY support encryption of the Atom session using TLS [RFC2246].

There are cases where an authentication mechanism may not be required, such as a publicly editable Wiki, or when using the PostURI to post comments to a site that does not require authentication to create comments.

13.1 [@@TBD@@ CGI Authentication]

This authentication method is included as part of the protocol to allow Atom servers and clients that cannot use HTTP Digest Authentication but where the user can both insert its own HTTP headers and create a CGI program to authenticate entries to the server. This scenario is common in environments where the user cannot control what services the server employs, but the user can write their own HTTP services.

14. Security Considerations

Because Atom is a publishing protocol, it is important that only authorized users can create and edit entries.

The security of Atom is based on HTTP Digest Authentication and/or [@@TBD@@ CGI Authentication]. Any weaknesses in either of these authentication schemes will affect the security of the Atom Publishing Protocol.

Both HTTP Digest Authentication and [@@TBD@@ CGI Authentication] are susceptible to dictionary-based attacks on the shared secret. If the shared secret is a password (instead of a random string with sufficient entropy), an attacker can determine the secret by exhaustively comparing the authenticating string with hashed results of the public string and dictionary entries.

See RFC 2617 for more detailed description of the security properties of HTTP Digest Authentication.

@@TBD@@ Talk here about using HTTP basic and digest authentication.

@@TBD@@ Talk here about denial of service attacks using large XML files, or the billion laughs DTD attack.

15. IANA Considerations

A Atom Collection Document, when serialized as XML 1.0, can be identified with the following media type:

MIME media type name: application

MIME subtype name: atomcoll+xml

Mandatory parameters: None.

Optional parameters:

"charset": This parameter has identical semantics to the charset parameter of the "application/xml" media type as specified in [RFC3023].

Encoding considerations: Identical to those of "application/xml" as described in [RFC3023], section 3.2.

Security considerations: As defined in this specification. [[anchor31: update upon publication]]

In addition, as this media type uses the "+xml" convention, it shares the same security considerations as described in [RFC3023], section 10.

Interoperability considerations: There are no known interoperability issues.

Published specification: This specification. [[anchor32: update upon publication]]

Applications that use this media type: No known applications currently use this media type.

Additional information:

Magic number(s): As specified for "application/xml" in [RFC3023], section 3.2.

File extension: .atomcoll

Fragment identifiers: As specified for "application/xml" in [RFC3023], section 5.

Base URI: As specified in [RFC3023], section 6.

Macintosh File Type code: TEXT

Person and email address to contact for further information: Joe Gregorio <joe@bitworking.org>

Intended usage: COMMON

Author/Change controller: IESG

An Atom Introspection Document, when serialized as XML 1.0, can be identified with the following media type:

MIME media type name: application

MIME subtype name: atomserv+xml

Mandatory parameters: None.

Optional parameters:

"charset": This parameter has identical semantics to the charset parameter of the "application/xml" media type as specified in [RFC3023].

Encoding considerations: Identical to those of "application/xml" as described in [RFC3023], section 3.2.

Security considerations: As defined in this specification. [[anchor33: update upon publication]]

In addition, as this media type uses the "+xml" convention, it shares the same security considerations as described in [RFC3023], section 10.

Interoperability considerations: There are no known interoperability issues.

Published specification: This specification. [[anchor34: update upon publication]]

Applications that use this media type: No known applications currently use this media type.

Additional information:

Magic number(s): As specified for "application/xml" in [RFC3023], section 3.2.

File extension: .atomsrv

Fragment identifiers: As specified for "application/xml" in [RFC3023], section 5.

Base URI: As specified in [RFC3023], section 6.

Macintosh File Type code: TEXT

Person and email address to contact for further information: Joe Gregorio <joe@bitworking.org>

Intended usage: COMMON

Author/Change controller: This specification's author(s). [[anchor35: update upon publication]]

16. References

16.1 Normative References

[AtomFormat] Nottingham, M. and R. Sayre, "The Atom Syndication Format", 1.0, July 2005.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[W3C.REC-xml-20040204] Yergeau, F., Paoli, J., Sperberg-McQueen, C., Bray, T., and E. Maler, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C REC REC-xml-20040204, February 2004.

[W3C.REC-xml-names-19990114] Hollander, D., Bray, T., and A. Layman, "Namespaces in XML", W3C REC REC-xml-names-19990114, January 1999.

16.2 Informative References

[RNC] Clark, J., "RELAX NG Compact Syntax", December 2001.

[W3C.REC-webarch-20041215]
Walsh, N. and I. Jacobs, "Architecture of the World Wide Web, Volume One", W3C REC REC-webarch-20041215, December 2004.

URIs

[1] <http://www.imc.org/atom-protocol/index.html>

Authors' Addresses

Joe Gregorio (editor)
BitWorking, Inc
1002 Heathwood Dairy Rd.
Apex, NC 27502
US

Phone: +1 919 272 3764
Email: joe@bitworking.com
URI: http://bitworking.com/

Bill de hOra (editor)
Propylon Ltd.
45 Blackbourne Square, Rathfarnham Gate
Dublin, Dublin D14
IE

Phone: +353-1-4927444
Email: bill.dehora@propylon.com
URI: http://www.propylon.com/

Appendix A. Contributors

The content and concepts within are a product of the Atom community and the Atompub Working Group. Robert Sayre was an editor for drafts 00-04.

Appendix B. Revision History

draft-ietf-atompub-protocol-05 - Added: Contributors section. Added: de hOra to editors. Fixed: typos. Added diagrams and description to model section. Incorporates PaceAppDocuments, PaceAppDocuments2, PaceSimplifyCollections2 (large-sized chunks of it anyhow: the notions of Entry and Generic resources, the section 4 language on the Protocol Model, 4.1 through 4.5.2, the notion of a Collection document, as in Section 5 through 5.3, Section 7 "Collection resources", Selection resources (modified from pace which talked about search); results in major mods to Collection Documents, Section 9.2 "Title: Header" and brokeout para to section 9.1 Editing Generic Resources). Added XML namespace and language section. Some cleanup of front matter. Added Language Sensitivity to some attributes. Removed resource descriptions from terminology. Some juggling of sections. See: http://www.imc.org/atom-protocol/mail-archive/msg01812.html.

draft-ietf-atompub-protocol-04 - Add ladder diagrams, reorganize, add SOAP interactions

draft-ietf-atompub-protocol-03 - Incorporates PaceSliceAndDice3 and PaceIntrospection.

draft-ietf-atompub-protocol-02 - Incorporates Pace409Response, PacePostLocationMust, and PaceSimpleResourcePosting.

draft-ietf-atompub-protocol-01 - Added in sections on Responses for the EditURI. Allow 2xx for response to EditURI PUTs. Elided all mentions of WSSE. Started adding in some normative references. Added the section "Securing the Atom Protocol". Clarified that it is possible that the PostURI and FeedURI could be the same URI. Cleaned up descriptions for Response codes 400 and 500.

Rev draft-ietf-atompub-protocol-00 - 5Jul2004 - Renamed the file and re-titled the document to conform to IETF submission guidelines. Changed MIME type to match the one selected for the Atom format. Numerous typographical fixes. We used to have two 'Introduction' sections. One of them was moved into the Abstract the other absorbed the Scope section. IPR and copyright notifications were added.

Rev 09 - 10Dec2003 - Added the section on SOAP enabled clients and servers.

Rev 08 - 01Dec2003 - Refactored the specification, merging the Introspection file into the feed format. Also dropped the distinction between the type of URI used to create new entries and the kind used to create comments. Dropped user preferences.

Rev 07 - 06Aug2003 - Removed the use of the RSD file for auto-discovery. Changed copyright until a final standards body is chosen. Changed query parameters for the search facet to all begin with atom- to avoid name collisions. Updated all the Entries to follow the 0.2 version. Changed the format of the search results and template file to a pure element based syntax.

Rev 06 - 24Jul2003 - Moved to PUT for updating Entries. Changed all the mime-types to application/x.atom+xml. Added template editing. Changed 'edit-entry' to 'create-entry' in the Introspection file to more accurately reflect it's purpose.

Rev 05 - 17Jul2003 - Renamed everything Echo into Atom. Added version numbers in the Revision history. Changed all the mime-types to application/atom+xml.

Rev 04 - 15Jul2003 - Updated the RSD version used from 0.7 to 1.0. Change the method of deleting an Entry from POSTing <delete/> to using the HTTP DELETE verb. Also changed the query interface to GET instead of POST. Moved Introspection Discovery to be up under Introspection. Introduced the term 'facet' for the services listed in the Introspection file.

Rev 03 - 10Jul2003 - Added a link to the Wiki near the front of the document. Added a section on finding an Entry. Retrieving an Entry now broken out into it's own section. Changed the HTTP status code for a successful editing of an Entry to 205.

Rev 02 - 7Jul2003 - Entries are no longer returned from POSTs, instead they are retrieved via GET. Cleaned up figure titles, as they are rendered poorly in HTML. All content-types have been changed to application/atom+xml.

Rev 01 - 5Jul2003 - Renamed from EchoAPI.html to follow the more commonly used format: draft-gregorio-NN.html. Renamed all references to URL to URI. Broke out introspection into it's own section. Added the Revision History section. Added more to the warning that the example URIs are not normative.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.