Network Working Group
Internet-Draft
Expires: April 30, 2006

J. Gregorio, Ed.
BitWorking, Inc
B. de hOra, Ed.
Propylon Ltd.
October 27, 2005

The Atom Publishing Protocol
draft-ietf-atompub-protocol-06.txt

Status of this Memo

Copyright Notice

Abstract

The Atom Publishing Protocol (APP) is an application-level protocol for publishing and editing Web resources. The protocol is based on 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]. The protocol supports the creation of arbitrary web resources and provides facilities for:

- Collections: Sets of resources, which may be retrieved in whole or in part.
- Introspection: Discovering and describing collections.
- Editing: Creating, updating and deleting resources.

2. 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].

Note: The Introspection Document 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. How to map an IRI to a URI is specified in Section 3.1 of Internationalized Resource Identifiers (IRIs) [RFC3987].

3. Terminology

For convenience, this protocol may be referred to as "Atom Protocol" or "APP".

The phrase "the IRI of a document" in this specification is shorthand for "an IRI which, when dereferenced, is expected to produce that document as a representation".

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 IRI, as defined in [RFC2616]. See [W3C.REC-webarch-20041215] for further discussion on resources.

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

collection - A resource that contains a set of member IRIs, as described in Section 8 of this specification.

member - A resource whose IRI is listed in a collection.

IRI template - A parameterized string that becomes a IRI when the parameters are filled in. See Section 9.

introspection document - A document that describes the location and capabilities of one or more collections. See Section 7

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

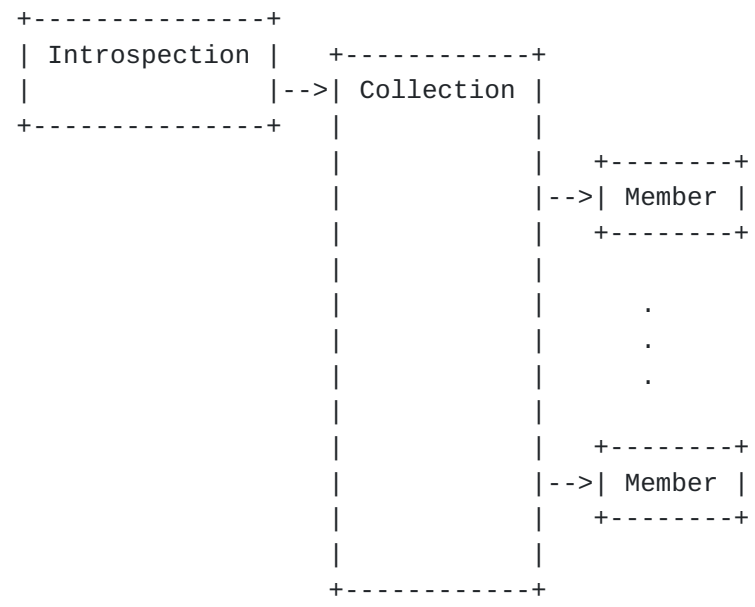
server-controlled element - An element of an Atom Entry whose value is enforced by the server and not editable by the client.

4. Protocol Model

The Atom Publishing Protocol uses HTTP to edit resources on the web. It provides a list based mechanism for managing collections of editable resources called member resources. Collections contain the IRIs and metadata describing member resources. The APP uses these HTTP verbs:

- GET is used to retrieve a representation of a resource or perform a query.
- POST is used to create a new, dynamically-named resource.
- PUT is used to update a known resource.
- DELETE is used to remove a resource.

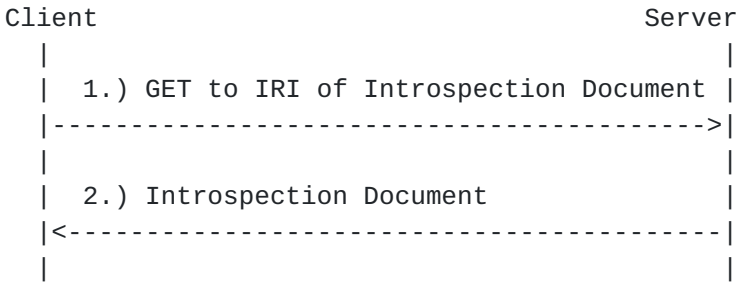
This diagram shows the APP model:



The introspection document contains the IRIs of one or more collections. A collection contains IRIs and metadata describing member resources. The protocol allows editing of resources with representations of any media-type. Some types of collections are specialized and restrict the resource representations of their members.

5. Protocol Operations

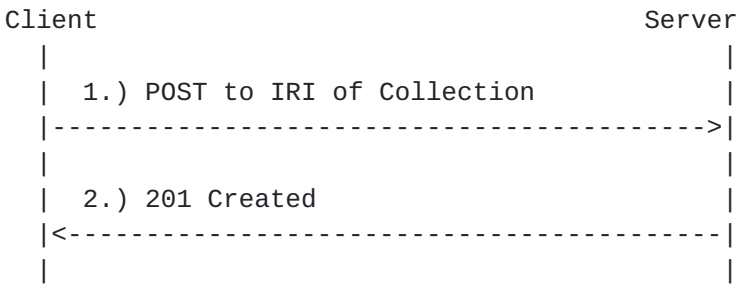
5.1 Retrieving an Introspection Document



1. The client sends a GET request to the IRI of the introspection document.

2. The server responds with the introspection document which enumerates the IRIs of all the collections, and the capabilities of those collections, that the service supports.

5.2 Creating a Resource



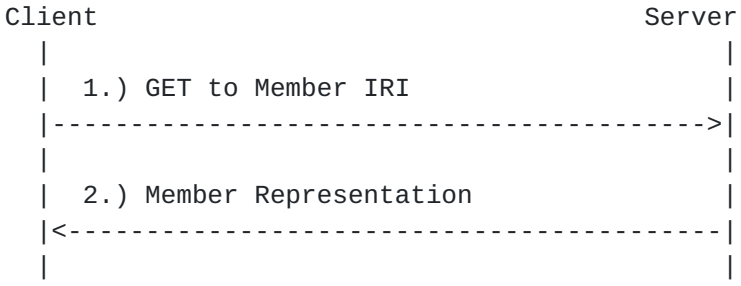
1. The client POSTs a representation to the IRI of the collection.

2. If the member resource was created successfully the server responds with a status code of 201 and a Location: header that contains the IRI of the newly created member resource.

5.3 Editing a Resource

Once a resource has been created and its IRI is known, that IRI may be used to retrieve, update, and delete it.

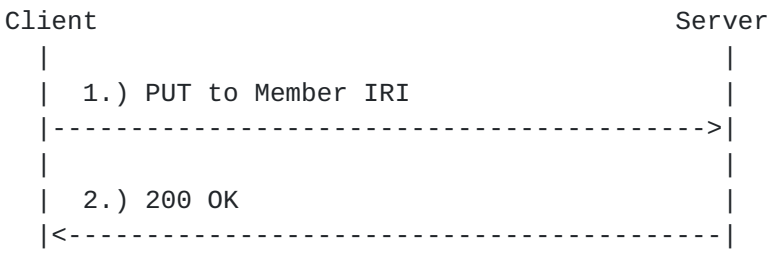
5.3.1 Retrieving a Resource



1. The client sends a GET request to the member's IRI to retrieve its representation .

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

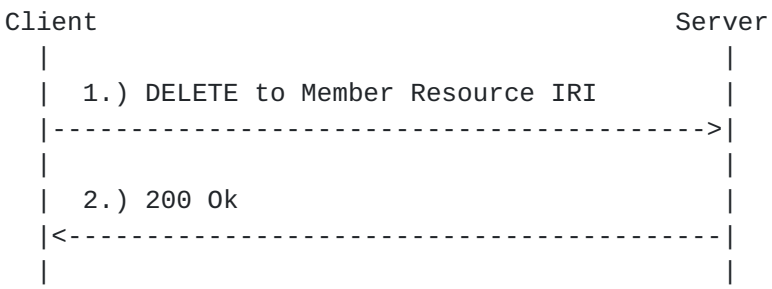
5.3.2 Updating a Resource



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

2. Upon a successful update of the resource the server responds with a status code of 200.

5.3.3 Deleting a Resource



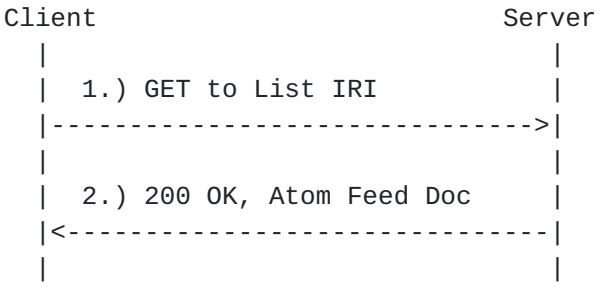
1. The client sends a DELETE request to the member's IRI.

2. Upon the successful deletion of the resource the server responds with a status code of 200.

Note: deleting a member also removes it from all the collections to which it belongs.

5.4 Listing Collections

To enumerate the members of a collection the client sends a GET to its IRI. This IRI is constructed from information in the introspection document. An Atom Feed Document is returned with one Atom Entry for each member resource that matches the selection criteria in the IRI. See Section 9 and Section 10 for a description of the feed contents.



1. The client sends a GET request to the membership list IRI.

2. The server responds with an Atom Feed Document containing the IRIs of all the collection members that match the selection criteria.

5.5 Success and Failure

The Atom Protocol uses HTTP status codes to signal the results of protocol operations. 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. Consult the HTTP specification [RFC2616] for the definitions of HTTP status codes.

6. XML-related Conventions

The data format in this specification is 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 any DTDs for Atom Protocol, and hence does not require them to be valid (in the sense used by XML).

6.1 Referring to Information Items

This specification uses a shorthand for two common terms: the phrase "Information Item" is omitted when naming Element Information Items and Attribute Information Items. Therefore, when this specification uses the term "element," it is referring to an Element Information Item in Infoset terms. Likewise, when it uses the term "attribute," it is referring to an Attribute Information Item.

6.2 XML Namespace Usage

The Namespace URI [W3C.REC-xml-names-19990114] for the data format described in this specification is:

http://purl.org/atom/app#

This specification uses the prefix "app:" for the Namespace URI. The choice of namespace prefix is not semantically significant.

This specification also uses the prefix "atom:" for the Namespace URI identified in [AtomFormat].

6.3 RELAX NG Schema

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. A complete schema appears in Appendix B.

6.4 Use of xml:base and xml:lang

XML elements defined by this specification MAY have an xml:base attribute [W3C.REC-xmlbase-20010627]. When xml:base is used, 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 Section 2.12 of XML 1.0 [W3C.REC-xml-20040204], .

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

7. Introspection Documents

7.1 Introduction

For authoring to commence, a client needs to first discover the capabilities and locations of collections offered. This is done using Introspection Documents. An Introspection Document describes workspaces, which are server-defined groupings of collections.

7.2 Example

```
<?xml version="1.0" encoding='utf-8'?>
<service xmlns="http://purl.org/atom/app#">
  <workspace title="Main Site" >
    <collection
      title="My Blog Entries"
      href="http://example.org/reilly/main" >
      <member-type>entry</member-type>
      <list-template>http://example.org/{index}</list-template>
    </collection>
    <collection
      title="Pictures"
      href="http://example.org/reilly/pic" >
      <member-type>media</member-type>
      <list-template>http://example.org/p/{index}</list-template>
    </collection>
  </workspace>
  <workspace title="Side Bar Blog">
    <collection title="Remaindered Links"
      href="http://example.org/reilly/list" >
      <member-type>entry</member-type>
      <list-template>http://example.org/l/{index}</list-template>
    </collection>
  </workspace>
</service>
```

This Introspection Document describes two workspaces. The first, called 'Main Site', has two collections called 'My Blog Entries' and 'Pictures' whose IRIs are 'http://example.org/reilly/main' and 'http://example.org/reilly/pic' respectively. 'My Blog Entries' is an Entry collection and 'Pictures' is a Media collection. Entry and Media collections are discussed in Section 7.3.4.

The second workspace is called 'Side Bar Blog' and has a single collection called 'Remaindered Links' whose collection IRI is 'http://example.org/reilly/list'. 'Remaindered Links' is an Entry collection.

Introspection documents are identified with the "application/atomserv+xml" media type (see Section 14).

While an introspection document allows multiple workspaces, there is no requirement that a service support multiple workspaces. In addition, a collection MAY appear in more than one workspace.

7.3 Element Definitions

7.3.1 The 'app:service' Element

The root of an introspection document is the app:service element.

```
namespace app = "http://purl.org/atom/app#"
start = appService
```

The "app:service" element is the container for introspection information associated with one or more workspaces. An app:service element MUST contain one or more app:workspace elements.

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

7.3.2 The 'app:workspace' Element

The 'app:workspace' element contains information elements about the collections of resources available for editing. The app:workspace element MUST contain one or more app:collection elements.

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

7.3.2.1 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.

7.3.3 The 'app:collection' Element

The app:collection contains information elements that describe the location and capabilities of a collection.

```
appCollection =
   element app:collection {
      appCommonAttributes,
      attribute title { text },
      attribute href { text },
      ( appMemberType
        & appListTemplate
        & extensionElement* )
   }
```

7.3.3.1 The 'title' Attribute

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

7.3.3.2 The 'href' Attribute

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

This specification defines two child elements for app:collection:

- app:member-type: a single element that contains the type of members that the collection can contain.

- app:list-template: a single element that contains a IRI template of a membership list. (See Section 9).

7.3.4 The 'app:member-type' Element

The app:collection element MUST contain one 'app:member-type' element. The app:member-type element value specifies the types of members that can appear in the collection.

```
appMemberType =
   element app:member-type {
         appCommonAttributes,
         ( appTypeValue )
   }

appTypeValue = "entry" | "media"
```

An Entry POSTed to a collection MUST meet the constraints of the app:member-type element.

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

- "entry" - The collection contains only member resources whose representation MUST be an Atom Entry. Further constraints on the representations of members in a collection of type "entry" are listed in Section 8.2.

- "media" - The collection contains member resources whose representation can be of any media type. Additional constraints are listed in Section 8.3.

In general the value of 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 Link Relations Section 14, and thus the IRI that would be obtained by appending the value of the rel attribute to the string "http://www.iana.org/assignments/member-type/".

7.3.5 The 'app:list-template' Element

The app:collection element MUST contain one 'app:list-template' elements. The element content of app:list-template is an IRI template (Section 9) for a collection.

```
appListTemplate =
   element app:list-template {
         appCommonAttributes,
         ( appUriTemplate )
   }

appUriTemplate = xsd:string { pattern = ".+\{.+\}.*" }
```

8. Collections

8.1 Creating resources with POST

Every collection accepts POST requests to create resources - the client POSTs a representation of the desired resource to the IRI of the collection. 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").

The status code returned for a successful creation POST MUST be 201 ("Created").

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

Clients MAY POST invalid Atom for initial resource creation - specifically the id and link elements MAY be omitted.

Below, the client requests to create a resource in a collection:

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

<entry xmlns="http://www.w3.org/2005/Atom">
    <title>Atom-Powered Robots Run Amok</title>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
</entry>
```

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

Successful creation is indicated by a 201 created response and includes a Location: header.

```
HTTP/1.1 201 Created
Date: Fri, 7 Oct 2005 17:17:11 GMT
Content-Length: 0
Location: http://example.org/edit/first-post.atom
```

8.1.1 Title: Header

The POST to a collection MAY contain a Title: header that indicates the client's suggested title for the resource. The server MAY ignore the Title: header or modify the requested title.

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

The syntax of this header MUST conform to the augmented BNF grammar in section 2.1 of the HTTP/1.1 specification [RFC2616].

8.2 Entry Collections

Entry Collections are collections that restrict their membership to Atom Entries. They are identified by having an app:member-type of "entry". Every member representation MUST contain an atom:link element with a relation of rel="edit" that contains the IRI of the member resource. Member representations MAY contain an app:control element (Section 10.2).

8.2.1 Role of Atom Entry Elements During Editing

The elements of an Atom Entry Document are either a client writable or server controlled.

Client Writable - An element of an Atom Entry whose value is editable by the client. Servers MAY modify the content of client writable elements. Some reasons that a server may change client writable content include length limits, obscenity filters or the addition of boilerplate text.

Server Controlled - An element of an Atom Entry whose value is enforced by the server and not editable by the client. Clients SHOULD NOT change the value of server controlled elements. Servers MUST NOT rely on clients preserving the values of server controlled elements.

Atom Entry Element	Property
atom:author	Client Writable
atom:category	Client Writable
atom:content	Client Writable
atom:contributor	Client Writable
atom:id	Server Controlled
atom:link	Client Writable
atom:published	Client Writable
atom:source	Client Writable
atom:summary	Client Writable
atom:title	Client Writable
atom:updated	Server Controlled
app:control	Client Writable

Table 1

8.3 Media Collections

Media Collections are collections whose member representations are not constrained. They are identified by having an app:member-type of "media".

8.3.1 Editing Media Resources

When a membership list resource returns an Atom Feed enumerating the contents of a Media Collection, all the Entries MUST have an atom: content element with a 'src' attribute. When creating a public, read-only reference to the member resource, a client SHOULD use the "atom:content/@src" attribute value.

9. Listing Collections

Collections, as identified in an Introspection Document, are resources that MUST provide representations in the form of Atom Feed documents. The entries in the returned Feed MUST be ordered by their 'atom:updated' property, with the most recently updated entries coming first in the document order. Every entry in the Feed Document MUST have an atom:link element with a relation of "edit" (See Section 10.1). Clients MUST NOT assume that an Atom Entry returned in the Feed is a full representation of a member resource. The value of atom:updated is only changed when the change to a member resource is considered significant. Insignificant changes do not result in changes to the atom:updated value and thus do not change the position of the corresponding entry in a membership list. Clients SHOULD be constructed with this in mind and SHOULD perform a GET on the member resource before editing.

Collections can contain extremely large numbers of resources. A naive client such as a web spider or web browser would be overwhelmed if the response to a GET contained every entry in the feed, and the server would waste large amounts of bandwidth and processing time on clients unable to handle the response.

For this reason, Introspection documents refer to collections not with IRIs but with IRI Templates, contained in the "app:member-type" child of "app:collection". An IRI Template is a string containing the embedded token "{index}".

To produce an IRI that can be used to retrieve part or all of the collection, software replaces the "{index}" with a pair of positive integer indices separated by a dash character. An IRI template MUST, after such substitution has been performed, constitute a syntactically valid IRI.

One or other index MAY be omitted, in which case the range is understood as stretching to 0 or infinity. The index values are 0 based and select members from the collection based on the member's index, with all of the members ordered by their 'atom:updated' property. 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. If a membership list is returned with a number of entries that is less than the number of entries requested than the client MAY assume that it has made a request that exceeds the last index of the members.

For example, suppose the client is supplied this IRI 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 0-14, giving:

http://example.org/blog/edit/0-14

10. Atom Entry Extensions

This specification adds one new value to the Registry of Link Relations and also adds a new element to Atom Entries called "app: control" for controlling publishing. These new links and app: control elements MAY appear in both membership lists and in member representations.

10.1 The 'edit' Link Relation

This specification adds the value "edit" to the Registry of Link Relations. The value of "edit" signifies that the IRI in the value of the href attribute is the IRI of the member resource, and is intended to be used to update and delete resources as described in this specification.

10.2 Publishing Control

This specification also adds a new element to Atom Entries for controlling publishing.

```
 pubControl =
    element app:control {
    atomCommonAttributes,
    pubDraft?
    & extensionElement
 }

 pubDraft =
   element app:draft { "yes" | "no" }
```

The "app:control" element MAY appear as a child of an "atom:entry" which is being created or updated via the Atom Publishing Protocol. The "app:control" element, if it does appear in an entry, MUST only appear at most one time.

The "app:control" element and its children elements MAY be included in Atom Feed or Entry Documents. The "app:control" element is considered "foreign markup" as defined in Section 6 of the Atom Syndication Format.

The "app:control" element MAY contain exactly one app:draft element and MAY contain zero or more extension elements as outlined in the Atom Syndication Format, Section 6. Both clients and servers MUST ignore foreign markup present in the app:control element that they do not know.

10.2.1 The app:draft Element

This specification defines only one child element for "app:control", "app:draft".

The number of "app:draft" elements in "app:control" MUST be zero or one. Its content MUST be one of the values "yes" or "no". A value of "no" means that the entry MAY be made publicly visible. If the "app:draft" element is missing then the value is understood to be "no". That is, if "app:control" and/or the "app:draft" elements are missing from an entry then the entry is considered not a draft and can be made publicly visible. Clients MUST understand "app:draft" elements and MUST NOT drop them from Atom Entries during editing. Clients MUST NOT operate on the expectation that a server will honor the value of an "app:draft" element. Servers MAY ignore "app:draft" elements and drop them from Atom Entries.

11. Example

This is an example of a client creating a new entry with an image. The client has an image to publish and an entry that includes an HTML 'img' element that uses that image. In this scenario we consider a client that has IRIs of two collections, an entry collection and a media collection, both of which were discovered through an introspection document. The IRI of the entry collection is:

http://example.net/blog/edit/

The IRI of the media collection is:

http://example.net/binary/edit

First the client creates a new image resource by POSTing the image to the IRI of the media collection.

```
POST /binary/edit/ HTTP/1.1
Host: example.net
User-Agent: Thingio/1.0
Content-Type: image/png
Content-Length: nnnn
Title: A picture of the beach

...binary data...
```

The member resource is created and an HTTP status code of 201 is returned.

```
HTTP/1.1 201 Created
Date: Fri, 25 Mar 2005 17:17:11 GMT
Content-Length: nnnn
Content-Type: application/atom+xml
Location: http://example.net/binary/edit/b/129.png

<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom">
    <title>A picture of the beach.</title>
    <link rel="edit"
        href="http://example.net/binary/edit/b/129.png"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-568596895695</id>
    <updated>2005-09-02T10:30:00Z</updated>
    <summary>Waves</summary>
    <content type="image/png"
        src="http://example.net/binary/readonly/129.png"/>
</entry>
```

The client then POSTs the Atom Entry that refers to the newly created image resource. Note that the client takes the IRI http://example.net/binary/readonly/129.png and uses it in the 'img' element in the Entry content:

```
POST  /blog/edit/ HTTP/1.1
Host: example.net
User-Agent: Thingio/1.0
Content-Type: application/atom+xml
Content-Length: nnnn

<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom">
    <title>What I did on my summer vacation</title>
    <updated>2005-09-02T10:30:00Z</updated>
    <summary>Beach!</summary>
    <content type="xhtml" xml:lang="en">
        <div xmlns="http://www.w3.org/1999/xhtml">
            <p>We went to the beach for summer vacation.
                Here is a picture of the waves rolling in:
                <img
                    src="http://example.net/binary/readonly/129.png"
                    alt="A picture of the beach."
                    />
            </p>
        </div>
    </content>
</entry>
```

12. 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 session using TLS (see [RFC2246]).

There are cases where an authentication mechanism is not be required, such as a publicly editable Wiki, or when using POST to send comments to a site that does not require authentication from a commenter.

12.1 [@@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.

13. Security Considerations

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 [RFC2617] for the 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.

14. IANA Considerations

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. [[anchor22: 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. [[anchor23: 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). [[anchor24: update upon publication]]

15. References

15.1 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.

[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.

[W3C.REC-xmlbase-20010627] Marsh, J., "XML Base", W3C REC W3C.REC-xmlbase-20010627, June 2001.

15.2 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.

Appendix B. RELAX NG Compact Schema

This appendix is informative.

The Relax NG schema explicitly excludes elements in the APP namespace which are not defined in this revision of the specification. Requirements for APP Processors encountering such markup are given in Section 6.2 and Section 6.3 of [AtomFormat].

```
# -*- rnc -*-
# RELAX NG Compact Syntax Grammar for the Atom Protocol

namespace app = "http://purl.org/atom/app#"
namespace local = ""

start = appService

# common:attrs

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

undefinedAttribute =
  attribute * - (xml:base | xml:lang | local:*) { text }

atomUri = text

atomLanguageTag = xsd:string {
   pattern = "[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*"
}

# app:service

appService =
   element app:service {
      appCommonAttributes,
      ( appWorkspace+
        & extensionElement* )
   }

# app:workspace

appWorkspace =
   element app:workspace {
      appCommonAttributes,
      attribute title { text },
```

```
         ( appCollection+
           & extensionElement* )
      }

   # app:collection

   appCollection =
      element app:collection {
         appCommonAttributes,
         attribute title { text },
         attribute href { text },
         ( appMemberType
           & appListTemplate
           & extensionElement* )
      }


   # app:member

   appMemberType =
      element app:member-type {
            appCommonAttributes,
            ( appTypeValue )
      }

   appTypeValue = "entry" | "media"

   # app:list-template

   appListTemplate =
      element app:list-template {
            appCommonAttributes,
            ( appUriTemplate )
      }


   # Whatever an IRI template is, it contains at least {index}

   appUriTemplate = xsd:string { pattern = ".+\{index\}.*" }

   # Simple Extension

   simpleExtensionElement =
      element * - app:* {
         text
      }

   # Structured Extension
```

```
   structuredExtensionElement =
      element * - app:* {
         (attribute * { text }+,
            (text|anyElement)*)
       | (attribute * { text }*,
          (text?, anyElement+, (text|anyElement)*))
      }

   # Other Extensibility

   extensionElement =
      simpleExtensionElement | structuredExtensionElement



   # Extensions

   anyElement =
      element * {
         (attribute * { text }
          | text
          | anyElement)*
      }

   # EOF
```

Appendix C. Revision History

draft-ietf-atompub-protocol-06 - Removed: Robert Sayre from the contributors section per his request. Added in PaceCollectionControl. Fixed all the {daterange} verbage and examples so they all use a dash. Added full rnc schema. Collapsed Introspection and Collection documents into a single document. Removed {dateRange} queries. Renamed search to list. Moved discussion of media and entry collection until later in the document and tied the discussion to the Introspection element app:member-type.

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 its 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 its 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 its 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.