Hierarchy Extensions to Atom Feeds
draft-divilly-atompub-hierarchy-00

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 6, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This specification defines a mechanism to create and remove AtomPub collections using the AtomPub protocol as well as to express hierarchies of feeds within the Atom Syndication Format.
Editorial Note

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

Table of Contents

1. Introduction ................................................. 3
   1.1. Namespace ............................................. 3
   1.2. Notational Conventions ................................. 3
2. Terminology .................................................. 4
3. Protocol Model ............................................... 4
   3.1. Feed Classification .................................... 4
   3.2. Protocol Operations ...................................... 6
      3.2.1. Creating a Collection .............................. 6
      3.2.2. Editing a Master Entry ............................ 7
      3.2.3. Removing a Master Entry ........................... 7
4. Backing Collection ............................................7
5. Master Detail Relations ....................................... 8
   5.1. Child feeds ............................................. 9
      5.1.1. The "detail" Link ................................ 9
      5.1.2. The "h:count" Extension Attribute .................. 9
      5.1.3. Example ............................................ 9
   5.2. Master entries ...........................................10
      5.2.1. The "master" Link ................................10
      5.2.2. Example ...........................................11
6. The 'h:role' Extension Attribute ..............................11
   6.1. The 'master' role ......................................12
   6.2. The 'detail' role ......................................13
7. Working With Master-Detail Feeds ..............................13
   7.1. Retrieving Master Feeds ................................14
   7.2. Creating A Master Entry .................................15
   7.3. Creating A Detail Entry ................................16
8. Security Considerations .......................................16
9. IANA Considerations ..........................................17
10. Normative References ........................................17
Appendix A. Acknowledgements ..................................18
Appendix B. Revision History ...................................18
Authors’ Addresses ..............................................18
1. Introduction

Many applications provide their data in the form of syndicated Web feeds using formats such as Atom [RFC4287] in order to enable application integration. Applications may also manipulate the contents of these data feeds using protocols such as AtomPub [RFC5023].

A key requirement for application data feeds is the ability to dynamically create new Collections and identify relationships among such feeds and Collections. This specification defines a mechanism for identifying hierarchical master-detail relations among data feeds so that consumer applications can perform standard AtomPub operations on them.

A hierarchical master-detail relation of an Entry to a Feed implies the detail Feed is created when the master Entry is created and the Feed is removed when the Entry is removed. The Entry is called the "master entry" and the Feed is called "detail feed". This relationship allows a client to use AtomPub [RFC5023] to create a new Collection by posting an Entry to an existing Collection.

This specification proposes optional and compatible extensions to Atom and AtomPub to ease the process of creating and manipulating collections and feeds based on those collections.

1.1. Namespace

The XML Namespaces URI for the XML data format described in this specification is:

    http://purl.org/atom/hierarchy/

This specification uses the prefix "h:" for the namespace name. The prefix "atom:" is used for "http://www.w3.org/2005/Atom", the namespace name of the Atom Syndication Format [RFC4287]. The prefix "app:" is used for "http://www.w3.org/2007/app", the namespace name of the Atom Publishing Protocol [RFC5023]. These namespace prefixes are not semantically significant.

1.2. 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].
2. Terminology

A "detail Feed" is a "logical Feed" as defined in Feed History and Paging [RFC5005] that does not contain a master Entry.

A "master Entry" is an Entry that contains a child Feed.

A "child Feed" is a "logical Feed" as defined in Feed History and Paging [RFC5005] that is contained in a master Entry.

A "master Feed" is a "logical Feed" as defined in Feed History and Paging [RFC5005] that contains only master Entries.

This specification also uses Atom link relations to identify different types of links; see the Atom specification [RFC4287] for information about their syntax, and the IANA link relation registry for more information about specific values.

Note that URI references in link relation values MAY be relative, and used in conjunction with the xml:base attribute [W3C.REC-xmlbase-20010627]. Such a relative URI (or IRI) is resolved as described in Section 5.1 of [RFC3986].

3. Protocol Model

Hierarchy extensions to Atom specify operations for creating, updating, and removing AtomPub Collections using existing AtomPub methods and extensions to Atom syntax.

3.1. Feed Classification

AtomPub [RFC5023] defines Feed, Entry, and Collection resources in the Atom syntax [RFC4287]. The hierarchy extensions to Atom are designed for use with unmodifiable Atom Feeds as well as with AtomPub Collections.

The extensions in this specification define two specialized kinds of feeds -- master Feed and detail Feed. Both are represented as Atom Feed Documents [RFC4287].
A master Feed is a container for master Entries. Each master Entry contains a child Feed, which can be any logical Feed. The kind child Feed’s metadata identifies the kind of new Entries that it can accept. The child Feed is created when the master Entry is created and the child Feed is removed when the master Entry is removed. A master Feed MAY itself be a child of another master Feed.

A master Feed can only accept a new master Entry. Each of its Entry contains a "detail" atom:link for a logical Feed.

atom:feed
  |  
  o- atom:link@rel="next" (0..1)
  o- atom:link@rel="prev" (0..1)
  o- atom:link@rel="first" (0..1)
  o- atom:link@rel="last" (0..1)
  o- atom:link@rel="self" (1..1)
  o- atom:link@rel="master" (0..1)
  o- app:collection (1..1)
     | 
     o- app:accept@h:role="master" (1..1)

A detail Feed MUST be a child of a master Entry.
A detail Feed can only accept a new detail Entry. It provides an atom:link back to its "master" Entry. None of its Entries contain a "detail" atom:link.

atom:feed
 |-- atom:link@rel="next" (0..1)
 |-- atom:link@rel="prev" (0..1)
 |-- atom:link@rel="first" (0..1)
 |-- atom:link@rel="last" (0..1)
 |-- atom:link@rel="master" (1..1)
 |-- atom:link@rel="self" (1..1)
 |  |-- app:collection (1..1)
 |  |  |-- app:accept@h:role="detail" (1..1)
 |  |-- atom:entry (0..n)
 |     |-- atom:link@rel="self" (1..1)
 |     |-- atom:link@rel="edit" (1..1)

3.2. Protocol Operations

AtomPub protocol governs the server behavior for operations involving master-detail feeds. This section illustrates how additional behavior results from the semantics of master-detail relations among feeds.

No special importance should be attached to the status codes shown in the illustrations below, and a server is entitled to using any HTTP status code that adequately represents the result of the requested operation.

3.2.1. Creating a Collection

Client Server

1.) POST to master Collection URI
    -------------------------------------------------->

2.) 201 Created
    Master Entry Document
<-------------------------------------------------->

1. The client sends a POST request containing an Atom Entry to the URI of the master Collection.
2. The server responds with an Atom Entry Document containing the IRIs of the newly created master Entry and the child Feed.

3.2.2. Editing a Master Entry

Client

<table>
<thead>
<tr>
<th>1.) PUT to master Entry URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Entry Document</td>
</tr>
</tbody>
</table>

Server

<table>
<thead>
<tr>
<th>2.) 200 OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Entry Document</td>
</tr>
</tbody>
</table>

1. The client sends a PUT request containing an Atom Entry to the URI of the master Entry.
2. If the request is successful, the server responds with a 200 status code without removing the child Feed from the master Entry Document.

3.2.3. Removing a Master Entry

Client

<table>
<thead>
<tr>
<th>1.) DELETE to master Entry URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------------------------------</td>
</tr>
</tbody>
</table>

Server

<table>
<thead>
<tr>
<th>2.) 204 No Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>----------------------------------</td>
</tr>
</tbody>
</table>

1. The client sends a DELETE request to the URI of the master Entry.
2. If the request is successful, the server responds with a 204 status code and removes the master Entry and detail Feed. Future requests to the URI of the master Entry or the child Feed or any of its entries MAY produce responses with 410 Gone status code.

4. Backing Collection

Often applications need to advertise that a Feed is modifiable, or in other words that new entries can be added to a Feed. If a syndicated feed document is backed by an AtomPub Collection resource, then clients may be able to augment the Feed by interacting with its backing Collection.
Section 8.3.5 of [RFC5023] specifies the semantics of an app:collection element appearing as a child of atom:feed element. Along those lines, if a Feed is backed by a Collection, the server SHOULD identify the backing Collection using the app:collection element as a child of the atom:feed element.

If the href attribute of this app:collection element is identical to that of the parent atom:feed element’s self link relation, then the client SHOULD treat the feed to be the Collection feed.

Example: Writable Collection backing a Feed

```xml
<atom:feed>
  <atom:link rel="self" href="http://example.org/feed"/>
  <app:collection href="http://example.org/collection">
    <atom:title type="text">Writable Feed</atom:title>
    <app:accept>application/atom+xml;type=entry</app:accept>
  </app:collection>
  ...
</atom:feed>
```

Example: Read-only Collection backing a Feed

```xml
<atom:feed>
  <atom:link rel="self" href="http://example.org/feed"/>
  <app:collection href="http://example.org/collection">
    <atom:title type="text">Read-only Feed</atom:title>
    <app:accept/>
  </app:collection>
  ...
</atom:feed>
```

Example: Writable Collection Feed

```xml
<atom:feed>
  <atom:link rel="self" href="http://example.org/collection"/>
  <app:collection href="http://example.org/collection">
    <atom:title type="text">Collection Feed</atom:title>
  </app:collection>
  ...
</atom:feed>
```

5. Master Detail Relations

Master detail relations among entries and feeds are indicated using link relations.
5.1. Child feeds

Every master Entry has a child Feed and vice versa. The contents of a child Feed can be supplied in the following ways:

1. Out-of-line reference: The client can retrieve the child Feed by following a URL specified in the master Entry. This URL is specified via an atom:link element as detailed below.

2. Inline content with out-of-line reference: The client can use the embedded content and expect the server to provide the same feed document at the URL referenced in the master Entry. The server MAY embed the atom:feed directly inside the atom:link element in the master entry document, in which case the client MAY safely use the embedded feed document as the representation of the child Feed.

5.1.1. The "detail" Link

Master Entries identify the URLs of their child Feed in their own metadata. A master entry MUST contain an atom:link element with link relation of "detail" to indicate the child Feed URL. The type attribute of this link element (if present), MUST be the Atom Feed content type, i.e., application/atom+xml;type=feed.

5.1.2. The "h:count" Extension Attribute

A master Entry MAY include an optional h:count attribute with a positive integral value identifying an approximate count of the number of entries in the detail Feed.

5.1.3. Example

Example: Entry with out-of-line reference to child Feed

```xml
<atom:entry>
  <atom:title type="text">My Portfolio</atom:title>
  <atom:link rel="detail" h:count="4"
    href="/finance/feeds/default/portfolios/1/positions"/>
  <atom:link rel="edit"
    href="/finance/feeds/default/portfolios/1"/>
  ...
</atom:entry>
```
Example: Entry with inline child Feed

```xml
<atom:entry>
  <atom:link rel="detail"
    href="/finance/feeds/default/portfolios/1/positions">
    <atom:feed>
      <atom:link rel="master"
        href="/finance/feeds/default/portfolios/1"/>
      <app:collection href="/finance/feeds/default/portfolios/1/positions">
        <atom:title>Oracle Corp</atom:title>
      </app:collection>
    </atom:feed>
    <atom:link rel="self"
      href="/finance/feeds/default/portfolios/1/positions">
      ...
    </atom:link>
  </atom:link>
  <atom:link rel="edit"
    href="/finance/feeds/default/portfolios/1"/>
  ...
</atom:entry>
```

5.2. Master entries

The contents of a master Entry can be supplied in the following ways:

1. Out-of-line reference: The client can retrieve the master Entry by following a URL specified in the child Feed. This URL is specified via an atom:link element as detailed below.

2. Inline content with out-of-line reference: The client can use the embedded content and expect the server to provide the same entry document at the URL referenced in the child Feed. The server MAY embed the atom:entry directly inside the atom:link element in the child feed document, in which case the client MAY safely use the embedded entry document as the representation of the master Entry.

5.2.1. The "master" Link

Child Feeds identify the URLs of their master Entry in their own metadata. A detail Feed MUST contain an atom:link element with link relation of "master" to indicate the master Entry URL. The type attribute of this link element (if present), MUST be the Atom Entry content type, i.e., application/atom+xml;type=entry.
5.2.2. Example

Example: Feed with out-of-line reference to master Entry

```xml
<atom:feed>
  <atom:title type="text">Positions</atom:title>
  <atom:link rel="master"
    href="/finance/feeds/default/portfolios/1"/>
  <atom:link rel="self"
    href="/finance/feeds/default/portfolios/1/positions"/>
  ...
</atom:feed>
```

Example: Feed with inline master Entry

```xml
<atom:feed>
  <atom:title type="text">Positions</atom:title>
  <atom:link rel="master"
    href="/finance/feeds/default/portfolios/1"/>
  <atom:entry>
    <atom:link rel="detail"
      href="/finance/feeds/default/portfolios/1/positions"/>
    <atom:link rel="edit"
      href="/finance/feeds/default/portfolios/1">
      ...
    </atom:link>
    <atom:link rel="self"
      href="/finance/feeds/default/portfolios/1/positions"/>
    ...
  </atom:entry>
</atom:feed>
```

6. The ‘h:role’ Extension Attribute

This specification defines additional metadata for AtomPub Service documents and collection-backed Feed documents for the purpose of identifying master and detail feeds.

The "h:role" attribute MAY be added to the app:accept element of an app:collection declaration. When present, the value MUST be one of "master" or "detail". If the "h:role" attribute is present then the content type of the accept element MUST be "application/atom+xml;type=entry".

```xml
hierarchyType =
  attribute h:role { "master" | "detail" }
```

Divilly & Mehta Expires November 6, 2009 [Page 11]
6.1. The 'master' role

If the "h:role" value is "master" then POSTing an Atom entry document
to that collection MUST cause the server to do both of the following:
- create a new Atom Entry resource whose location is provided as
defined in Section 9.2.1 of [RFC5023]
- create a new AtomPub Collection resource whose location is
  provided in the master entry document

Example: Hierarchy metadata for collections

```xml
<app:collection href="http://example.org/collection">
  <title type="text">Master Feed</title>
  <app:accept
    h:role="master">application/atom+xml;type=entry
  </app:accept>
</app:collection>
```

A POST to that collection would be as per [RFC5023]:

```
POST /collection HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <title>A master entry</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-8003ffefa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <author><name>John Doe</name></author>
  <content>Some text.</content>
</entry>
```
The server produces the following response:

201 Created
Location: http://example.org/collection/master1

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
  xmlns:h="http://purl.org/atom/hierarchy/">
  <title>A master entry</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-8003ffefa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <author><name>John Doe</name></author>
  <content>Some text.</content>
  <link rel="self" href="/collection/master1"/>
  <link rel="edit" href="/collection/master1"/>
  <link rel="detail" h:count="0"
    href="/collection/master1/collection">
    <feed xmlns:app="http://www.w3.org/2007/app">
      <id>urn:uuid:1225c695-cfb8-4ebb-8003ffefa6a</id>
      <app:collection href="/collection/master1/collection">
        <app:accept h:role="detail">
          application/atom+xml;type=entry
        </app:accept>
      </app:collection>
      <link rel="self" href="/collection/master1/collection"/>
    </feed>
  </link>
</entry>

6.2. The 'detail' role

If the "h:role" value is "detail" then the server behaves as specified in Section 9.2.1 of [RFC5023] when a client POSTs an Atom entry document to the Collection.

7. Working With Master-Detail Feeds
7.1. Retrieving Master Feeds

Below is a feed for a finance portfolio tracker application. Each user has a Collection of portfolios. Each portfolio is a Collection of positions.

```xml
<feed xmlns='http://www.w3.org/2005/Atom'
     xmlns:app='http://www.w3.org/2007/app'
     xmlns:h='http://purl.org/atom/hierarchy/
     xmlns:f='http://example.com/finance'
     xml:base='http://example.com'>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaa80da344cba6a</id>
  <updated>2008-10-01T13:05:30.000Z</updated>
  <title type='text'>Portfolio Feed</title>
  <link rel='service'
        href='http://finance/feeds/default/service'/>
  <app:collection href='http://finance/feeds/default/portfolios'>
    <title type='text'>Portfolio Feed</title>
    <app:accept h:role='container'>application/atom+xml</app:accept>
  </app:collection>
  <entry>
    <id>urn:uuid:1225c695-cfb8-4ebb-aada80da344efa6a</id>
    <updated>2008-06-10T01:29:49.000Z</updated>
    <title type='text'>My Portfolio</title>
    <link rel='detail'
          href='http://finance/feeds/default/portfolios/1/positions'/>
    <link rel='edit'
          href='http://finance/feeds/default/portfolios/1'/>
    <summary>Default portfolio</summary>
    <f:portfolioData currencyCode='USD' gainPercentage='0.0'
                      return1w='0.0' return1y='0.0' return3m='0.0' return3y='0.0'
                      return4w='0.0' return5y='0.0' returnOverall='0.0' returnYTD='0.0'/>
  </entry>
</feed>
```
7.2. Creating A Master Entry

If a user wishes to add a new portfolio, they can do so by making an AtomPub POST request to the portfolio Collection identified above in the app:collection element of the master portfolio feed:

POST /finance/feeds/default/portfolios HTTP/1.1
Host: example.com
Content-Type: application/atom+xml; type=entry
Content-Length: nnn

<entry xmlns='http://www.w3.org/2005/Atom'
   xmlns:f='http://example.com/finance'>
   <id/>
   <updated>2008-06-10T23:38:01.000Z</updated>
   <title type='text'>Hanky Panky</title>
   <f:portfolioData currency='USD'/>
</entry>

The server generates the following response after creating the master entry and the detail feed.

201 Created
Location: /finance/feeds/default/portfolios/2
Content-Type: application/atom+xml; type=entry
Content-Length: nnn

<entry xmlns='http://www.w3.org/2005/Atom'
   xmlns:f='http://example.com/finance'
   xmlns:base='http://example.com'>
   <id>urn:uuid:1225c695-cfb8-4ebb-aada-80da344efa6a</id>
   <updated>2008-06-10T23:38:01.000Z</updated>
   <title type='text'>Hanky Panky</title>
   <link rel='detail'
      type='application/atom+xml;type=feed' title='positions'
      href='/finance/feeds/default/portfolios/2/positions'/>
   <link rel='detail'
      type='text/html'
      href='http://finance.example.com/portfolio/2'/>
   <link rel='edit'
      href='/finance/feeds/default/portfolios/2'/>
   <summary>Default portfolio</summary>
   <f:portfolioData currencyCode='USD' gainPercentage='0.0'
      return1w='0.0' return1y='0.0' return3m='0.0' return3y='0.0'
      return4w='0.0' return5y='0.0' returnOverall='0.0' returnYTD='0.0'/>
</entry>

This entry identifies a newly created Holdings Collection in the link
with the "detail" relation.

7.3. Creating A Detail Entry

If a user wishes to add a new position, they can do so by making an AtomPub POST request to the portfolio's Collection identified above in the app:collection element of the detail positions feed:

```http
POST /finance/feeds/default/portfolios1/positions HTTP/1.1
Host: example.com
Content-Type: application/atom+xml; type=entry
Content-Length: nnn

<entry xmlns='http://www.w3.org/2005/Atom'>
  <symbol xmlns='http://api.example.com/finance'
    exchange='NASDAQ' fullName='Siebel Systems' symbol='SEBL'/>
</entry>
```

The server generates the following response after creating the detail entry.

```
201 Created
Location: /finance/feeds/default/portfolios/1/positions/NASD:SEBL
Content-Type: application/atom+xml; type=entry
Content-Length: nnn

<entry xmlns='http://www.w3.org/2005/Atom'
  xmlns:f='http://api.example.com/finance'>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa9b</id>
  <updated>2008-10-01T23:28:01.000Z</updated>
  <category scheme='http://api.example.com/cat/2008#kind'
    term='http://api.example.com/finance/2008#position'/>
  <title type='text'>Siebel Systems</title>
  <link rel='edit'
    href='/finance/feeds/default/portfolios/1/positions/NASD:SEBL'/>
  <f:portfolioData currencyCode='USD' gainPercentage='0.0'
    return1w='0.0' return1y='0.0' return3m='0.0' return3y='0.0'
    return4w='0.0' return5y='0.0' returnOverall='0.0' returnYTD='0.0'/>
  <f:symbol exchange='NASDAQ' fullName='Google Inc'
    symbol='SEBL'/>
</entry>
```

This entry identifies a newly created SEBL position.

8. Security Considerations

Atom Publishing Protocol Hierarchy Extensions is subject to the
security considerations found in Section 8 of [RFC4287] and Section 15 of [RFC5023].

9. IANA Considerations

This specification defines the following new relations that have been added to the Link Relations registry:
- Attribute Value: detail
  - Description: A URI that refers to the child feed for a master entry.
  - Expected display characteristics: none
  - Security considerations: See this draft
- Attribute Value: master
  - Description: A URI that refers to the master entry for a child feed.
  - Expected display characteristics: none
  - Security considerations: See this draft

10. Normative References


Appendix A.  Acknowledgements

Bill de hOra and Ashish Motivala reviewed early drafts of this I-D and helped strengthen the text and make it easier to read.

Appendix B.  Revision History

00 - Initial Revision.

Authors’ Addresses

Colm Divilly
Oracle Corp.

Email: colm.divilly@oracle.com
URI: http://cdivilly.wordpress.com

Nikunj Mehta
Oracle Corp.

Email: nikunj.mehta@oracle.com
URI: http://o-micron.blogspot.com/