Network Working Group
Internet-Draft
Expires: November 19, 2006

R. Sayre

May 18, 2006

2-Way RSS

draft-sayre-2-way-rss-05

Status of this Memo

Copyright Notice

Abstract

This memo presents a protocol that uses XML and HTTP to publish and edit Web resources.

Table of Contents

1. Introduction

2-Way RSS uses HTTP [RFC2616] and XML [XML 1.0] to publish and edit Web resources.

1.1. Editor's Note

To discuss this draft, please join the 2-Way RSS mailing list [1]. Membership is open to all.

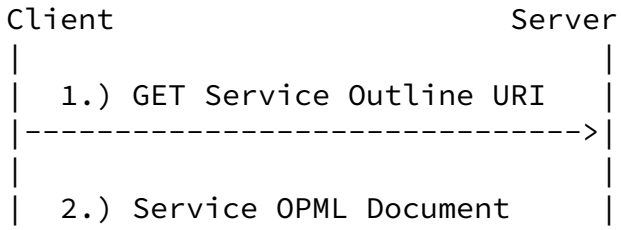
2. The 2-Way RSS Model

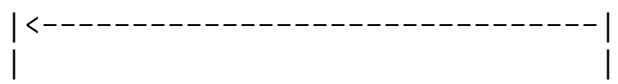
2-Way RSS uses HTTP to operate on collections of Web resources represented by RSS feeds [RSS]. In 2-Way RSS, individual RSS items have URIs [RFC3986]. This section illustrates the editing cycle for RSS items.

- GET is used to retrieve an item or perform a read-only query.
- POST is used to create a new item.
- PUT is used to update an existing item.
- DELETE is used to remove an item.

3. Discovery

To discover the location of the feeds exposed by a 2-way RSS service, the client must locate and request the Service Outline, an OPML 2.0 document [OPML2].

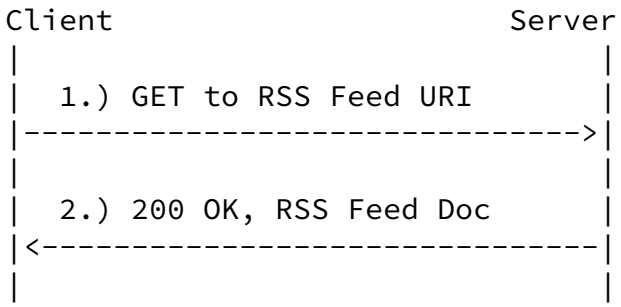




1. The client sends a GET request to the Service Outline URI.
2. The server responds with an OPML Document containing the locations of feeds provided by the service. The content of this document can vary based on aspects of the client request, including, but not limited to, authentication credentials.

4. Listing

Once the client has discovered the location of a feed in the outline, it can request a listing of the feed's items. However, a feed might contain an extremely large number of items, so servers are likely to list a small subset of them by default.

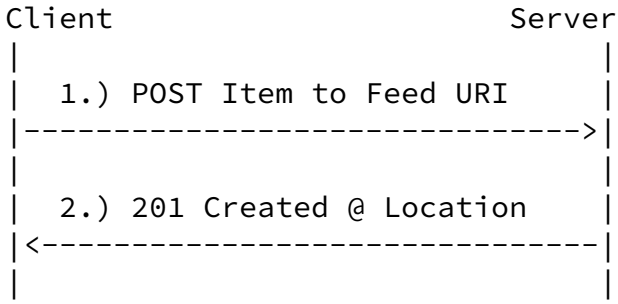


1. The client sends a GET request to the RSS Feed's URI.
2. The server responds with an RSS Feed Document containing a full or partial listing of the feed's membership.

5. Authoring

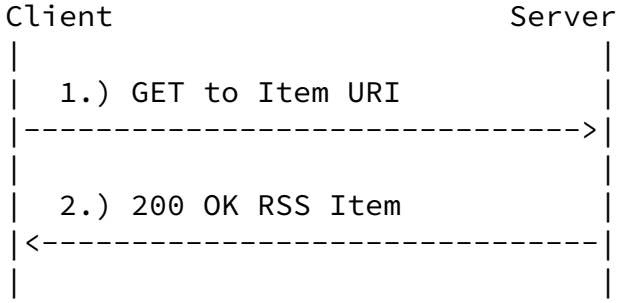
After locating a feed, a client can add entries by sending a POST request to the feed; other changes are accomplished by sending HTTP requests to each item.

5.1. Create



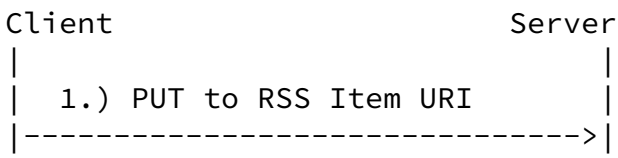
1. The client sends an RSS item to the server via HTTP POST. The Request URI is that of the RSS Feed.
2. The server responds with a response of "201 Created" and a "Location" header containing the URI of the newly-created RSS item.

5.2. Read



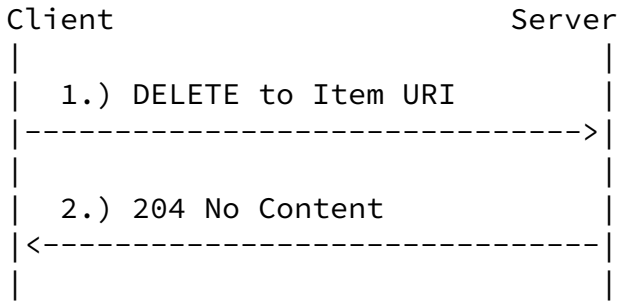
1. The client sends a GET request to the item's URI.
2. The server responds with an RSS item.

5.3. Update



1. The client PUTs an updated RSS item to the item's URI.
2. The server responds with a successful status code.

5.4. Delete



1. The client sends a DELETE request to the item's URI.
2. The server responds with successful status code.

5.5. Success and Failure

HTTP defines classes of response. 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 for more detailed definitions of each status code.

6. 2-Way RSS Feeds

6.1. GET

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

simple GET request represent a server-determined subset of the items in the feed.

An example 2-Way RSS feed:

```
<rss xmlns:r="http://example.org/2006/02/2WayRSS">
  <channel>
    <title>The Baron in the Trees</title>
    <link>http://example.org/trees.html</link>
    <description>Recent posts.</description>
    <!-- 0 or more item elements follow -->
    <item>
      <title>Chapter One</title>
      <description>It was on the fifteenth of June, 1767,
      that Cosimo Piovasco di Rond&#x00F2;, my brother,
      sat among us for the last time.</description>
      <guid>uuid:941e12b4-6eeb-4753-959d-0cbc51875387</guid>
      <link>http://example.org/chapter1.html</link>
      <r:link rel="edit" href="./item7.rss"/>
    </item>
  </channel>
</rss>
```

Each member item is represented by an <item> element, but those items are not an editable representation of the each item. To retrieve the source representation of the item, clients send a GET request to the URI found in each item's edit link, an 'r:link' element Section 9.1 with an 'edit' relation.

6.2. POST

A 2-Way RSS feed also accepts POST requests. The client POSTs a new item to the RSS feed. Some feeds only accept POST requests with

certain media-types, so a POST request could result in a response with a status code of 415 ("Unsupported Media Type"). In the case of a successful creation, the status code is 201 ("Created").

Example HTTP request creating a new item in a feed:

```
POST /wall HTTP/1.1
Host: example.org
```

```
User-Agent: Cosimo/1.0
Content-Type: text/xml
Content-Length: nnnn

<item xmlns:r="http://example.org/2006/02/2WayRSS">
  <title>Chapter One</title>
  <description>It started out simple...</description>
  <guid>uuid:941e12b4-6eeb-4753-959d-0cbc51875387</guid>
  <link>http://example.org/chapter1.html</link>
  <r:link rel="edit" href="./item7.rss"/>
</item>
```

Example response.

```
HTTP/1.1 201 Created
Date: Mon, 21 Mar 2005 19:20:19 GMT
Server: CountBasic/1.0
ETag: "4c083-268-423f1dc6"
Location: http://example.org/items/foo13241234.xml
```

7. 2-Way Media RSS Feeds

The items within 2-way Media RSS Feeds do not represent uniform types of content. For example, they might contain podcasts, JPEG images, text documents, MPEG movies, or any other type of resource the server allows.

7.1. GET

2-Way Media RSS Feeds return an RSS feed much like the textual 2-Way RSS feeds described above, but with a few additions. The entries also contain an <enclosure> element with a 'url' attribute pointing to the media object. This URL can be used to edit the uploaded media object, using PUT and DELETE. Such items may contain edit links used to edit the item metadata.

An example 2-Way Media RSS Feed:

```
<rss>
  <channel>
    <title>My Pics</title>
    <link>http://example.org/pics</link>
    <description>Recent photos.</description>
    <!-- 0 or more item elements follow -->
    <item>
      <title>beach25</title>
      <link>http://example.org/beach-pic1.html</link>
      <guid>uuid:941e12b4-6eeb-4753-959d-0cbc51875387</guid>
      <description>This was awesome.</description>
      <enclosure url="http://example.org/beach.tiff"
                 length="15234"
                 type="image/tiff" />
    </item>
  </channel>
</rss>
```

Implementations require that each such item contain either a <title> or <description> element. The value for the <title> element will likely be provided by the client, as a way for users to associate their local objects with those they have uploaded to the server (see POST below).

7.2. POST

To add an item to a 2-Way Media RSS feed, clients POST the resource to the Media feed's URL. Clients should provide a 'Title' request header [OBJECT] to provide the server with a short string identifying the object to users.

Clients may include a 'Content-Description' header [RFC2045] providing a more complete description of the content. Data gleaned from other entity headers, such as 'Keywords' [RFC2822] and 'From' [RFC2616], may also be exposed in the RSS feed.

Servers may inspect the POSTed entity for additional metadata to be exposed in an <item> element when listed in a 2-Way Media RSS feed. For example, the server might inspect a JPEG file for EXIF headers containing creator data.

An example request:

```
POST /pics HTTP/1.1
Host: example.org
User-Agent: Cosimo/1.0
Content-Type: image/tiff
Content-Length: nnnn
Title: A trip to the beach
From: Bobby <bobby@example.org>
Keywords: beach, digginit, tags, mrpibb, redvines
Content-Description: It was so fun.

...binary data...
```

An example response:

```
HTTP/1.1 201 Created
Date: Mon, 21 Mar 2005 19:20:19 GMT
Server: CountBasic/2.0
ETag: "4c083-268-423f1dc6"
Location: http://example.org/stuff/beach.tiff

<item>
  <title>A trip to the beach</title>
  <link>http://example.org/beach.jpg</link>
  <guid>uuid:4019de8a-7d08-4ca7-aee4-1a7cb92f3173</guid>
  <enclosure url="http://example.org/beach.tiff"
             length="15234"
             type="image/tiff" />
</item>
```

The server's response contains a 'Location' header that gives the URI of the created media resource. The body of the response shows the created item. The enclosure element contains more detailed information about the created media resource.

8. Service Outlines

Many 2-Way RSS applications require a basic resource layout in order to ease configuration requirements. Servers use Service Outline OPML 2.0 documents to convey information about related groups of 2-Way RSS feeds. On a blogging service, for example, each group might represent a distinct blog and associated resources. Normal RSS feeds

have a type attribute of 'rss'. 2-Way RSS feeds have a type attribute of '2-way rss'. Read-only media feeds have a type attribute of

'media rss', and 2-Way media feeds have a type attribute of '2-way media rss'.

Example Service Outline document:

```
<opml version="2.0">
  <head>
    <title>My Blogs</title>
  </head>
  <body>
    <outline text="Pretending to be Excited" type="2-way rss"
             xmlUrl="http://example.org/pretend.rss"/>
    <outline text="Quick Links" type="2-way rss"
             xmlUrl="http://example.org/side.rss"/>
  </body>
</opml>
```

Servers are not required to expose a Service Outline OPML document, but experimental deployment experience has shown that a single document which signals some basic information about the server's configuration can greatly simplify client implementations. The simplest useful Service Outline OPML document shows the location of a single feed:

```
<opml version="2.0">
  <head>
    <title>Flickr</title>
  </head>
  <body>
    <outline text="My Pics" type="2-way media rss"
             xmlUrl="http://example.org/pics.rss"/>
  </body>
</opml>
```

If another 2-Way RSS feed is added, another element is added to the Service Outline.

```
<opml version="2.0">
  <head>
```

```
    <title>Flickr</title>
  </head>
  <body>
    <outline text="Pics" type="2-way media rss"
             xmlUrl="http://example.org/pics.rss"/>
    <outline text="Overwraught Prose" type="2-way rss"
             xmlUrl="http://example.org/blog.rss"/>
  </body>
</opml>
```


More extensive services could require some amount of hierarchical grouping.

```
<opml version="2.0">
  <head>
    <title>Flickr</title>
  </head>
  <body>
    <outline text="Pics" type="2-way media rss"
             xmlUrl="http://example.org/pics.rss"/>
    <outline text="Crazy Delicious">
      <outline text="Overwraught Prose" type="2-way rss"
               xmlUrl="http://example.org/blog.rss"/>
      <outline text="Still More Overwraught Prose" type="2-way rss"
               xmlUrl="http://example.org/blog2.rss"/>
    </outline>
  </body>
</opml>
```

Many publishing systems include a categorization system. An outline element with a type attribute value of 'categories' can be used to list the available categories. In this example, the category feeds are read-only.

```
<opml version="2.0">
  <head>
    <title>Flickr</title>
  </head>
  <body>
    <outline text="Pics" type="2-way media rss"
             xmlUrl="http://example.org/pics.rss"/>
    <outline text="Categories" type="categories">
```

```
          <outline text="Category 1" type="rss"
                   xmlUrl="http://example.org/cat/1"/>
          <outline text="Category 2" type="rss"
                   xmlUrl="http://example.org/cat/2"/>
        </outline>
      </body>
    </opml>
```

9. The 2-Way RSS Namespace

The 2-Way RSS namespace URI is 'http://example.org/2006/02/2WayRSS' [@ will update]. The examples in this specifcation use the prefix 'r', but any prefix will do in practice.

9.1. The 'r:link' Element

The syntax and semantics of the 'r:link' element are identical to the XHTML link element [XHTML], except for the namespace, and one other exception: the r:link element allows arbitrary attributes.

10. References

[OBJECT] Berners-Lee, T., "Object Header lines in HTTP", 1992, <http://www.w3.org/Protocols/HTTP/Object_Headers.html>.

[OPML2] Winer, D., "OPML 2.0 Specification", March 2006, <http://www.opml.org/spec2>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[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.

[RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

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

[RSS] Cadenhead, et al., "Really Simple Syndication -- RSS Advisory Board Announcements", <http://www.rssboard.org/>.

[RSS091] Libby, D., "RSS 0.91 Spec, revision 3", June 1997, <http://my.netscape.com/publish/formats/rss-spec-0.91.html>.

[RSS092] Winer, D., "RSS 0.92", December 2005, <http://backend.userland.com/rss092>.

[RSS2] Winer, D., "RSS 2.0", July 2003, <http://blogs.law.harvard.edu/tech/rss>.

[XHTML] Pemberton, S., "XHTML. 1.0 The Extensible HyperText Markup Language (Second Edition)", W3C REC REC-xhtml1-20020801, August 2002, <http://www.w3.org/TR/2002/REC-xhtml1-20020801>.

[XML 1.0] Bray, T., Sperberg-McQueen, C., Paoli, J., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C REC REC-xml-20040204, February 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

[1] <http://groups.google.com/group/2-Way-RSS>

Appendix A. Acknowledgements

[@ will be updated prior to publication]

Author's Address

Robert Sayre

Email: rfsayre@boswijck.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to

pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

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