Atom Publishing Protocol
and
Nokia Share Online

Copyright © 2009 Randy Simons
v0.9.1 20090724

Table of Contents

1 Introduction

1.1 About this document

This document describes the basics of web syndication. It also describes how the Nokia Shared Online software for Symbian mobiles also uses these web syndication techniques, to allow people to upload media directly from their phone to a weblog or photo website.

I've written this document for a course about XML technologies. My original goal was to extend this later beyond the scope of the course, and to write some PHP software that could act as a server for Nokia Share Online. But, for various reasons, explained at my website, I'm not that motivated any more to do so.

However, my website statistics do show there is interest in the workings of Share Online, and therefore I've decided to release the document in its current, unfinished state. It probably contains errors and has omissions. Still, I hope it will be of some value for developers.

1.2 Syndication and Nokia Share Online

With the advent of weblogs or blogs and news sites on the internet it became possible for people to closely follow new, events and opinions in areas of their interest. But manually visiting a list of sites to check if there's some new content (polling) is not very convenient. A push-solution, where the user is notified if, and only if, there's news is much easier.

The RSS feed protocol made this possible. It still is a polling-technique, due to the limitations of the underlying transport protocol HTTP, but the polling is automated. The user receives a notification from the feed-client if there's news, and the content of the news items is conveyed in a standardized way, much like e-mail.

This standard also made it possible for applications and websites to aggregate other feeds, combine them, and present them as one new feed. Keeping up-to-date is now easier than ever.

But the RSS-protocol has some quirks and limitations. To remedy those, the Atom-protocol was designed. A sub-part of the Atom protocol is the Atom Publishing Protocol, which provides a standardized way of publishing new items to a feed.

Atom Publishing Protocol is used by Nokia Share Online, an application for Nokia smartphones. This software allows users to easily "blog" to various services which support the Atom Publishing Protocol, and even attach photos or videos to the blog-item. The software automatically controls and adapts the media to conform to the specification of the service used. Using a downloadable configuration file users can easily add extra services.

This document is "informative". Details are often omitted, as it tries to give the reader an overview of Atom in conjunction with the Nokia Share Online software.

First, the Atom protocol is described, and then zooms in on the Atom Publish Protocol. Next, some information about the configuration files used by Nokia Share Online is provides.

The Nokia Share Online software will also be used as a guide for the examples. Next, the Atom protocol is compared to RSS in more detail. Finally some tools are discussed which support Atom Publishing Protocol.

Some basic knowledge about the concept of webfeeds is assumed.

2 Motivation

2.1 The syndication formats

The RSS protocol made it possible for people to follow a multitude of weblogs and news sites without much fuss. All that is required is a simple client, and currently even the web browsers have built-in support.

It has a few key characteristics which might have contributed to its success:

- There is no registration required. The client simply requests the feed as long as the user wants. The server doesn't need to maintain a list of subscribers.

- Basic HTTP can be used to download the feeds and items. Therefore no special configuration to firewalls or proxies are necessary.

- A simple and open protocol based on XML makes it very easy to implement both servers and clients.

However, there are some drawbacks and ambiguities in the RSS protocol which the community wanted to solve. Most notable are:

- The content of the items in the feed are limited to plain text or HTML. It's up to the client to figure out which is used. Moreover, this limits the possibility to create feeds for any other types of payloads, like images.

- There is no standard for publishing new items to the feed and editing existing items. (Although the FeedSync standard might do the trick)

- Internationalization-support is limited. A channel (feed) can be only one language.

The Internet Engineering Task Force (IETF) developed the Atom standard. It is a true vendor independent XML-based standard, with support for any type of content payload and easily extendible using XML namespaces.

The Atom-standard consists of two separate parts: the Atom Syndication Format, used by clients to read the feeds, and the Atom Publishing Protocol, used by clients to publish or edit items.

2.2 Nokia Share Online

Current Nokia Symbian-based smartphones are equipped with the Nokia Share Online software. Users can easily upload photos and videos made with the mobile phone, optionally accompanied by text and comments, to online photo albums and weblog systems, like Flickr, Vox and Nokia's own Ovi. Since these services already supported the Atom Publishing Protocol to add new entries and media to one's collection, it seems only logical for Nokia to use the Atom Publishing Protocol for Share Online.

3 Explanation

3.1 Atom namespaces

Because the Atom-protocol is divided into two separate parts, there are two namespaces. For the Atom Syndication-part the namespace is http://www.w3.org/2005/Atom. For the Atom Publishing Protocol-part the namespace is http/www.w3.org/2007/app. From here on, the atom-namespace will be used for the Syndication-part, and the app-namespace will be used for the Publishing-part.

3.2 Atom Syndication Format

Atom entries

The basic element of Atom is the `<atom:entry />`-element. This is a container for some metadata about an individual entry, like the author(s), catagories, summary, categories, etc. It can also contain the actual content of the entry. If not, an alternative link to the content must be given.

Some metadata-elements are required: `<atom:title />`, the title of this entry, `<atom:id />`, a unique id of this entry in the form of an IRI (RFC3987) and `<atom:updated />`, the time this entry was last updated in the form fo a RFC3339-timestamp, are required elements.

An `<atom:entry />` can be part of a feed, or it can be a stand-alone Entry-document, in which the entry-element is the top node of the document. A stand-alone document is used for publishing new entries to a feed, using the Atom Publishing Protocol.

Any type of data can be used as content. The `type`-attribute of the `<atom:content />`-element defines the type. There are five options:

Type value	Meaning
text	The content is in XML-escaped human readable text, and is displayed as plain text.
html	Like `text`, but the content is processed as HTML.
xhtml	Since XHTML is a well-formed XML, XHTML-code can be used as content without further encoding. However, the top-node of the XHTML-code must be a `<div />`-element. Also, all in-line XHTML-code must be in the XHTML-namespace.en
text/* or */xml or */*+xml mime-types	Much like `text` or `xhtml`. Text-based data must be XML-escaped. XML-based content can be placed inline, but be sure to specify a namespace.
Other mime-types	Binary data is encoded as base64-encoded data.

Content can also be placed on an external resource, instead of in-line. In that case, a `src`-attribute for the `<atom:content />`-element is required, which then contains an IRI of the location of the data.

Zero or more `<atom:link />`-elements can be provided. The `href`-attribute provides the target URI. The semantics of the link is based on the `rel`-attribute. The most important ones are:

Rel value	Meaning of URI
alternate	Link to an alternative form of the content. For example, if the content of the entry is forum posting, the link points to the webpage containing an online version of the posting.
edit	Provides a link to the URI to which an updated version of this entry should be posted. The current entry is then replaced by the updated entry. See the Atom Publishing Protocol.
edit-media	If the content of this entry is binary media, then this link provides the IRI to which the user can post a new version of this media; all other data is maintained. See the Atom Publishing Protocol

If the `rel`-attribute is omitted, the link-element has the same meaning as when `rel="alternate"` was given.

The mime-type of a entry-document is `application/atom+xml`.

Atom feeds

The "normal" use for the Atom Syndication Protocol is to provide a feed of entries. Users can subscribe to a feed of their wish. In fact, it's not really a subscription, as the server doesn't maintain a list of subscribers. All it means is that the user's Atom client periodically requests a feed-document from the server. The request is normally done using standard HTTP.

The feed-document is a XML-document, with a `<atom:feed />`-element as root node. Just like the `<atom:entry />`-element, the feed-element contains some metadata, which is relevant for the entire feed. There are also some required metadata-elements: `<atom:title />`, `<atom:id />` and `<atom:updated />`. Finally, the feed contains zero or more `<atom:entry />`-elements, carnying the actual entries. The list of entries is ordered by the time of last update of the entries. The newest entry is first, the oldest last.

Clients maintain a record of the time they've retrieved the feed the last time. When a client requests the feed again, all entries whose `<atom:updated />`-timestamp is newer that the time last time the feed was retrieved, are considered as "new entries".

The mime-type for a feed-document is `application/atom+xml`.

3.3   Atom Publishing Protocol

The Atom Publishing Protocol defines methods to add, modify and delete "Web Resources" using normal HTTP. It's main purpose is to control the contents of Atom feeds. The basic idea is that users request a service-document. This document provides links to one or more collections. Users can use those links to publish new items.

Collections

In the Atom Publishing Protocol a "collection" is a "Resource that contains a set of Member Resources". Those member resources can be Atom Entry documents, or other type of resources like media files accompanying a Atom entry.

For most intends and purposes, collections can bee seen as feeds: when requesting the contents of a collection, the server will provide an Atom feed document with its entries; when publishing some new content, this content is posted to a collection, after which the user can request an updated feed document.

The element for a collection, `<app:collection />`, provides only three important details:

- A `href`-attribute, which provides an URI used to add new resources.
- A `<atom:title />`-element, providing a name. (note: the atom namespace is used!)
- Zero or more `<app:accept />`-elements, which provides mime types of the resources that can be published to this collection.

When no accept-element is provided, the collection (only) accepts Atom-entries.

Atom service document

The atom service document specification is part of the Atom Publishing Protocol. An user can retrieve a service-document from the server, to see which collections are available to him.

Collections can be grouped into separate workspaces. For example: a user may have more than one weblog using one account. The collections belonging to a single weblog can be grouped into one workspace. Thus, a workspace is nothing more than a set of collections, and a title for the workspace.

The structure of a service document is simple:

- A `<app:service />`-element is always the root-node. This node contains:
  - One or more `<app:workspace />`-elements, which contains:
    - A `<atom:title />`-element, naming the workspace.
    - Zero or more `<app:collection />`-elements.

The mime-type for a service-document is `application/atomserv+xml`.

3.4 Controlling resources

The Atom Publishing Protocol provides mechanism for the following actions:

- Adding a new resource to a collection.
- Requesting an existing resource.
- Updating an existing resource.
- Deleting an existing resource.

All these actions are executed using normal HTTP requests.

Adding a normal entry

Adding a new entry to a collection is easy: construct an entry-document, and send it using HTTP POST to the URI of the collection.

The server response with code 201 (created), a `Location`-header pointing to the newly created entry, and the entry-document. This entry does not have to be identical to the posted version; some details like the id may have been changed by the server.

Adding a binary entry

Simply send the binary data to the URI of the collection using HTTP POST. Don't forget the mime type. Again, the server will respond with code 201, the location header pointing to the Entry document, and a representation of the newly added resource as entry document.

The server may provide a link to which the user can upload a new version of the binary data.

Adding an entry with attached media

Often users want to attach media to a specific entry, as is the case with Nokia Share Online.

1. Do while there are media files to upload in queue:
    1. Send the image to collection URI, using HTTP POST. Server responds with the created entry as an Atom entry document, which includes an Edit URI.
    2. Remember the URI in the `src`-attribute in the content-element of the new entry, which points to the uploaded media.
    3. Edit the new entry to add meta-data like title, summary, author, categories etc.
    4. Send the altered entry to the editing-link, using HTTP PUT.
    5. Remove image from queue.
2. Then POST a new entry, with title and text-content, with links (rel="related") to the media-files. The `href`-attributes of the links contain the values of the rembered `src`-attributes of the media-entries.

Requesting a resource

Requesting a specific resource is accomplished mostly using the normal Atom Syndication Format: the user request the feed-document from the URI of the collection. This feed contains the links to the resources in this collection. Note that the resources will always be represented in the form of an entry-document.

Deleting a resource

Instead of requesting the resource using HTTP GET, the user can send a HTTP DELETE request to the URI of the resource. The server will then delete the resource.

Limiting access

Of course, usually it's not desired to allow everybody to publish or modify collections. Atom services can use normal HTTP authentication mechanisms. But since the authentication are not specified by the protocol, other may be used as well, such as wsse (Web Services Security) or proprietary sollutions.

3.5 Nokia Share Online

Configuration file

For convenience, users can download a special XML-based configuration file for various services, which contains the following data used by the application:

- The name of the service.
- URL for sign-up, if the user has no account yet.
- URL to obtain the collection of feeds.
- An icon, in the form of a SVG-document (another open XML format!)
- The media-types and formats accepted by the service. (list of mime-types)
- The maximum dimensions and size for the media. (maximum resolution, maximum file size)
- Whether a title, caption and text is allowed.
- Number of media files are allowed in a single post.
- Privacy options. (private, public)
- The authentication-method. (wsse, http basic authentication or custom authentication methods)

The structure is mostly straight forward. Unfortunately however, Nokia didn't publish details of the configuration files, so much of the information is an educated guess.

The protocol-element defines what protocol the application should use. For Atom v1.0, the protocol -element is as follows: `<protocol plugin="Atom" version="1.0">...</protocol>`. The `<endpoint_path />`-element within the protocol elements links to a document with a service description. For Atom, this is the location where the services-document can be retrieved.

After downloading, the file is parsed by the software. The user only has to provide a user name and password for the service he chooses. From there on, the Share Online application makes sure the uploaded photos, videos and text conforms to the specification of the service. The application will automatically scale down photos and videos, to meet the demands of the service.

Share Online and Atom

The software can use the Atom Publish Protocol for publishing the media. Share Online v3.x supports both Atom Publishing Protocol v0.3 and Atom Publishing Protocol v1.0. Since v0.3 is obsolete, only v1.0 is used from here on.

The Atom services-document as provided by Flickr has three workspaces. The title of these workspaces are specified. It seems the titles of these workspaces have a special meaning for NSO:

Workspace 'posting'
> which contains one feed for posting new media and entries. The feed accepts both images and entries. It is unclear at this moment what will happen when separate feeds are used for posting entries and images, or what will happen when more than one feed is specified.

Workspace 'commenting'
> Unclear. The collection doesn't accept entries. The URL of the collection is generic, it doesn't contain a specific username or account.

Workspace 'feeds'
> Contains several feeds which allows to follow uploaded photos of your own photos, photos uploaded by contacts, and photos uploaded by everyone. Share Online lets users select one of this feeds, and then shows the images in this feed.

However, since the Share Online application is proprietary software, which supports only a limited set of services out-of-the-box, it is unclear what the true possibilities and limitations are. The application is specifically made to publish photos and videos from a mobile phone, so support for other media or text-only-entries might be limited.

Requirements for implementation

Minimum requirements; allows for uploading images; no viewing/reading the feed:

- provide a valid configuration file.
- provide services document, with a single workspace, containing a collection which accepts image/jpeg binary files.
- Process uploaded binary files, reply with code 201.

4 Examples

4.1 An Atom Entry document

A minimalistic Entry document. It contains only required elements.

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <title>Atom-Powered Robots Run Amok</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <content>Some text.</content>
</entry>
```

XHTML-content can easily be placed inline. But note that the root-node of the XHTML-content must be a `<div />`-element. Also, the XHTML-content should be in the XHTML-namespace.

```
…
<content type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">This is <strong>XHTML</strong>content.</div>
</content>
…
```

4.2 An Atom Feed document

A minimalistic feed-document, containing one entry:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Example Feed</title>
  <link href="http://example.org/" />
  <updated>2003-12-13T18:30:02Z</updated>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
  <entry>
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atom03" />
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>
</feed>
```

4.3 An Atom Services document

The following services-document is created by Flickr.com, using Nokia Share Online. Note the use of the various workspaces, and their titles.

```
<service xmlns="http/www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom">
  <workspace>
    <atom:title>posting</atom:title>
    <collection href="http://www.flickr.com/services/atom/1.0/photo/upload/">
      <atom:title>Your Photostream</atom:title>
      <accept>image/*, entry</accept>
      <link type="text/html" rel="alternate"
        href="http://www.flickr.com/photos/testpurposes/" />
      <link type="text/html" rel="http://m.flickr.com/"
      href="http://m.flickr.com/photostream.gne?id=12345" />
    </collection>
  </workspace>
  <workspace>
```

```
    <atom:title>commenting</atom:title>
    <collection href="http://www.flickr.com/services/atom/1.0/photo/comment/" />
  </workspace>
  <workspace>
    <atom:title>feeds</atom:title>
    <collection href="http://www.flickr.com/services/atom/1.0/photos/">
      <atom:title>Your Recent Photos and Videos</atom:title>
    </collection>
    <collection href="http://www.flickr.com/services/atom/1.0/photos/contacts/">
      <atom:title>Your Contacts' Photos and Videos</atom:title>
    </collection>
    <collection href="http://www.flickr.com/services/atom/1.0/photos/everyone/">
      <atom:title>Recent Uploads From Everyone</atom:title>
    </collection>
  </workspace>
</service>
```

4.4   Adding an entry to a collection

Using a HTTP POST request to add a new entry to a collection

```
POST /blog/entries HTTP/1.1
Host: www.example.org
Content-Type: application/atom+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0" ?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <title>Atom-Powered Robots Run Amok</title>
  <link href="http://example.org/2003/12/13/atom03"/>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <author><name>James</name></author>
  <summary>Some text.</summary>
</entry>
```

The server responds with code 201 (created) and the newly created entry:

```
HTTP/1.1 201 Created
Date: nnnn
Content-Type: application/atom+xml; charset=utf-8
Content-Location: /blog/entries/1
Location: /blog/entries/1
ETag: "/blog/entries/1?1"
Last-Modified: Sat, 12 Aug 2006 13:40:03 GMT

<?xml version="1.0" ?>
<entry xmlns="http://www.w3.org/2005/Atom" >
  <id>tag:example.org,2006:/blog/entries/1</id>
  <title>Atom-Powered Robots Run Amok</title>
  <link href="http://example.org/2003/12/13/atom03"/>
  <link rel="edit" href="http://example.org/blog/entries/1" />
  <updated>2006-08-12T13:40:03Z</updated>
  <author><name>James M Snell</name></author>
  <summary>Some text.</summary>
</entry>
```

4.5   Adding a binary data to a collection

Posting a binary to a collection, as is done by Nokia Share Online, when uploading a photo to
Flickr.com. Note the alternative (proprietary) authorization-mechanism. Also note the Slug-header,
which provides a name for the resource. This is not required however.

```
POST /services/atom/1.0/photo/upload/ HTTP/1.1
Host: www.flickr.com
Accept-Language: nl-NL, nl;q=0.9, *;q=0.8
Authorization: FlickrAuth {some uid}
User-Agent: ISF Share Online/3.0
Content-Length: 32561
Content-Type: image/jpeg
Slug: title-of-photo.jpg
X-Nokia-MusicShop-Bearer: WLAN
X-Nokia-MusicShop-Version: 1.0.0

{binary data}
```

The server responds with the created entry:

```
HTTP/1.1 201 Created
Date: Tue, 26 May 2009 19:54:53 GMT
P3P: policyref="..."
Location: http://www.flickr.com/services/atom/1.0/photo/12345/edit/
Content-Length: 831
Connection: close
Content-Type: application/atom+xml; charset=utf-8


<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>tag:flickr.com:photo:12345</id>
  <title>some-photo.jpg</title>
  <content type="image/jpg"
      src="http://farm4.static.flickr.com/12345/12345_12345_o.jpg" />
  <summary></summary>
  <published>2009-05-26T12:54:54PDT</published>
  <updated>2009-05-26T12:54:54PDT</updated>
  <link type="text/html" rel="alternate"
      href="http://www.flickr.com/photos/johndoe/12345/" />
  <link type="text/html" rel="http://m.flickr.com/"
      href="http://m.flickr.com/photo.gne?id=12345" />
  <author>
    <name>John Doe</name>
    <uri>http://www.flickr.com/services/atom/1.0/photos/12345@N04/</uri>
  </author>
  <link rel="edit" href="http://www.flickr.com/services/atom/1.0/photo/12345/edit/"
type="application/x.atom+xml" />
</entry>
```

4.6 A Nokia Share Online configuration file

The exact specifications of this file is unknown. However, most of it is clear and strait forward. Most important for Atom is the protocol-description at the bottom. The endpoint_path-element is a link to a Atom services document. Apart from the proprietary Flickr-authentication, Share Online at least also supports WSSE.

The configuration-file also provides an icon for the service. This is included as a base64-encoded SVG-file.

The correct content type must be specified for the configuration-file, so make sure you provide a `Content-type: application/isf.sharing.config` header when using HTTP.

```
<?xml version="1.0"?>
<configure_file service_id="flickr" category_id="1" coding_id="2" version="3.1">
  <provider>
    <configure_file_URL>http://www.flickr.com/services/atom/1.0/flickr_configuration.cfg<
/configure_file_URL>
    <browserView reltype="http://m.flickr.com/" />
    <signup_URL />
```

```
    </provider>
  <laf>
    <title>Flickr</title>
    <context_pane_icon file="flickr.svg">flickr.svg</context_pane_icon>
    <selection_list_icon file="flickr.svg">flickr.svg</selection_list_icon>
    <icon file="flickr.svg" encoding="BASE64">[base64 encoded SVG]</icon>
  </laf>
  <media_options>
    <format_list>
      <format>image/jpg</format>
      <format>image/jpeg</format>
      <format>image/bmp</format>
      <format>image/png</format>
      <format>image/gif</format>
      <format>image/tiff</format>
    </format_list>
    <maximum_pixels horizontal="0" vertical="0" />
    <maximum_bytes bytes="5242880" />
  </media_options>
  <entry_options>
    <title_text present="1" required="0" />
    <caption_text present="0" required="0" />
    <body_text present="1" required="0" />
    <max_items items="6" />
    <min_items items="1" />
    <max_size size="0" />
    <tags present="1" required="0" />
    <privacy_levels present="1" required="1" />
  </entry_options>
  <location_options>
    <gps_info present="1" />
    <network_info present="0" />
  </location_options>
  <protocol_options>
    <protocol plugin="Atom" version="1.0" authentication="flickr">
      <endpoint_path>http://www.flickr.com/services/atom/1.0</endpoint_path>
      <roundtrip_editing>0</roundtrip_editing>
      <flickrauth_url>http://www.flickr.com/services/rest/</flickrauth_url>
      <flickrauth_web_url>http://m.flickr.com/services/auth/</flickrauth_web_url>
    </protocol>
  </protocol_options>
</configure_file>
```

Nokia Share Online requesting a feed

An example of Share Online requesting a (Flickr)-feed. Every photo has its own entry, even if the user uploaded more than one in one posting. Note the `<link rel="enclosure" ... />`-elements, which link to the actual photos. Share Online will request those photos for previewing. These photos appear to be scaled down by the server to fit the screen of the mobile phone.

Also note the `<nokia:can_comment>true</nokia:can_comment>` lines. Users can post comments on photos, which will probably be posted to the "commenting"-collection.

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:nokia="http://schemas.nokia.com/onlinesharing/2006">
  <id>tag:flickr:user:12345</id>
  <title>Testpurposes' Recent Photos</title>
  <updated>2009-05-31T20:19:33GMT</updated>
  <link rel="replies"  href="http://www.flickr.com/services/atom/1.0/photos/comments/" />
  <link rel="alternate" href="http://www.flickr.com/photos/testpurposes/" />
  <link rel="http://m.flickr.com/" href="http://m.flickr.com/photostream.gne?id=12345" />
  <author>
    <name>Testpurposes</name>
    <uri>http://www.flickr.com/services/atom/1.0/photos/12345/</uri>
  </author>
```

```xml
  <entry>
    <title>26-05-2009</title>
    <id>tag:flickr.com:photo:12345</id>
    <published>2009-05-26T12:55:00PDT</published>
    <updated>2009-05-26T12:55:03PDT</updated>
    <summary mode="text/plain">Multi</summary>
    <link type="image/jpeg" rel="enclosure"
    href="http://farm4.static.flickr.com/3602/12345_12345_m.jpg" />
    <link type="text/html" rel="alternate"
    href="http://www.flickr.com/photos/testpurposes/12345/" />
    <link type="text/html" rel="http://m.flickr.com/"
    href="http://m.flickr.com/photo.gne?id=12345" />
    <link type="text/html" rel="replies"
href="http://www.flickr.com/services/atom/1.0/photo/12345/comments/" />
    <nokia:can_comment>true</nokia:can_comment>
  </entry>
  <entry>
    <title>26-05-2009</title>
    <id>tag:flickr.com:photo:12345</id>
    <published>2009-05-26T12:54:54PDT</published>
    <updated>2009-05-26T12:54:55PDT</updated>
    <summary mode="text/plain">Multi</summary>
    <link type="image/jpeg" rel="enclosure"
href="http://farm4.static.flickr.com/3597/12345_12345_m.jpg" />
    <link type="text/html" rel="alternate"
    href="http://www.flickr.com/photos/testpurposes/12345/" />
    <link type="text/html" rel="http://m.flickr.com/"
    href="http://m.flickr.com/photo.gne?id=12345" />
    <link type="text/html" rel="replies"
href="http://www.flickr.com/services/atom/1.0/photo/12345/comments/" />
    <nokia:can_comment>true</nokia:can_comment>
  </entry>
  <entry>
    <title>26-05-2009</title>
    <id>tag:flickr.com:photo:12345</id>
    <published>2009-05-26T12:45:13PDT</published>
    <updated>2009-05-26T12:45:16PDT</updated>
    <summary mode="text/plain">Test</summary>
    <link type="image/jpeg" rel="enclosure"
href="http://farm4.static.flickr.com/3397/12345_12345_m.jpg" />
    <link type="text/html" rel="alternate"
href="http://www.flickr.com/photos/testpurposes/12345/" />
    <link type="text/html" rel="http://m.flickr.com/"
href="http://m.flickr.com/photo.gne?id=12345" />
    <link type="text/html" rel="replies"
href="http://www.flickr.com/services/atom/1.0/photo/12345/comments/" />
    <nokia:can_comment>true</nokia:can_comment>
  </entry>
</feed>
```

5 Evaluation

The Atom protocol solves the problems and limitations of the older RSS-protocol. It is a true open Internet standard, and uses XML to the full extend. By design, it's more flexible than RSS. However, there are "add-on" standards written for RSS, which provide extra services.

Both standards however work just fine: they make it really easy for users to follow weblogs and news. Both standards use basic HTTP, which makes firewall-traversal a non-issue.

The biggest advantage of Atom is the Atom Publishing Protocol. Thanks to this protocol, users can use a whole range of clients to publish their creations to weblogs etc. which support the Atom Publishing Protocol.

The biggest problem for Atom is the popularity of the term "RSS". RSS has become a genericized trademark. People may be using Atom feeds, but call it "RSS feeds".

I think the popularity of feeds will grow, as it are easy standards, which are well supported, and can relay information much faster than traditional mailing lists using e-mail. Subscribing and unsubscribing from feeds is also much easier, without the need to provide an e-mail address, which could be abused for spam.

There is a drawback compared to e-mail: there is no equivalent for IMAP for feeds; when a user has subscribed to a specific feed from both at work and at home, he will be notified of every new item twice: at work, and at home. A centralized server which keeps track of the items read would solve this issue.

Nokia thankfully used the Atom Publishing Protocol for their Share Online application and Ovi services. Other services than the factory-provided Flickr, Vox and Ovi can be used by users, as long as the service provide a configuration file (for Share Online v3.x). The structure of this configuration file is very easy.

6 Tool support

There are a lot of feed readers available. Most of them support both RSS and Atom. Web browsers like Opera and Internet Explorer 7+ support these protocols out-of-the-box.

Some third-party blogging and sharing platforms have added support for Nokia Share Online, such as Wordpress and zoopy.com. Even newspapers have added support for Share Online internally, to allow their journalists to quickly send in new copy using their mobile phones!

For developers, there are many libraries available for many platforms to support the Atom-protocol, both for syndication as for publishing.

7 Resources

My website: http://randysimons.nl/

Getting to know the Atom Publishing Protocol: http://www.ibm.com/developerworks/library/x-atompp1/

Atom Syndication Format: http://www.ietf.org/rfc/rfc4287.txt

Atom Publishing Protocol: http://www.ietf.org/rfc/rfc5023.txt

IRI (Internationalized Resource Identifiers): http://www.ietf.org/rfc/rfc3987.txt

Date/time format used by Atom: http://www.ietf.org/rfc/rfc3339.txt

WSSE authentication (Web Services Security): http://en.wikipedia.org/wiki/Wsse

Nokia Share Online website: http://europe.nokia.com/A4388334

Share Online server: http://randysimons.nl/pagina_140_NL.xhtml