NOKIA

Lifeblog Posting Protocol Specification

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

Disclaimer

The information in this document is provided "as is," with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

Nokia's liability for any errors in the document is limited to the documentary correction of errors. Nokia will not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this document or the information in it.

Nokia, NOKIA logo, and Nokia Connecting People are registered trademarks of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective companies, and they are mentioned for identification purposes only.

Contents

1   Introduction

Nokia Lifeblog is a PC and mobile phone software combination that keeps a multimedia diary of the items collected with select Nokia mobile phones. Lifeblog automatically organizes photos, videos, text messages, and multimedia messages into a chronology that can be easily browsed, searched, and shared.

The core of Lifeblog is Life Logging, particularly the automatic collection, tagging, and management of multimedia content. With Lifeblog, the phone becomes the Life Recorder and the PC the Life Archive. For sharing, the PC provides an easy-to-use interface, and with the mobile, items can be kept on the phone, as a mobile subset.

Furthermore, Lifeblog 1.5 enables sharing through SMS, MMS, email, or posting to web logs via Atom. All these sharing methods are complementary and have different usage experiences, providing the Lifeblogger the freedom in how to share or post to blogs. Likewise, because Nokia is not a service provider, these sharing methods use widely established protocols to enable compatibility with existing services.

Relevant to this document, Nokia has chosen and supports Atom as a blogging protocol. In addition to enabling posting via Atom from Lifeblog 1.5 on the PC and mobile, Nokia is working with blogging companies, such as Six Apart, and the Atom community to enhance the mobile aspects the Atom API.

This document is offered as an aid to those wishing to use Lifeblog to post to their web servers. It outlines the posting protocol, as used by Lifeblog, including any special elements and tips for the server-side actions. It is offered in good faith, with the understanding that inaccuracies or omissions in the document were unintentional and that the Atom 1.0 specifications are still not finalized.

2 Protocol

Lifeblog posting is based on the Atom API publishing protocol [5], which is an application protocol that works on top of HTTP. Both PC and Series 60 versions of Lifeblog 1.5 behave similarly when it comes to posting, though there are some minor changes that are described in this document.

2.1 Atom API

At the time of releasing Lifeblog 1.5, the Atom API publishing protocol [5] and Atom Syndication Format [6] are in a PRE-DRAFT state. The specification work is ongoing in the IETF "ATOMPUB Working Group" [7], with the expected release of version 1.0 during the first half of 2005. Since the Atom standard is not yet ready, the protocol described in Atom pre-draft ([5]) will change. The future development of Lifeblog will follow the Atom standardization process and update the supported protocols accordingly, while still trying to remain compatible with the protocol described in this document.

2.1.1 Atom XML documents and MIME types

The Atom format [6] defines XML documents that the Atom API publishing protocol ([5]) uses. Atom documents can have the following registered MIME types [8]: "`application/atom+xml`" and "`application/x.atom+xml`". The first one is the official IANA registered MIME type and the second one was introduced in the Atom Syndication Format 0.3 PRE-DRAFT ([6]). Both should be supported and understood by the server.

In addition to this, Lifeblog 1.5 PC application sets the *Content-Type* header ([1] Ch. 14.17) of the HTTP requests to "`application/xml`". So for the server to be fully Lifeblog 1.5 compatible it should not reject the request as long as the body of the request is otherwise valid (as defined in this document). The erroneous *Content-Type* header will be fixed to be the registered MIME type in a future release of Lifeblog. The Series 60 Lifeblog 1.5 application uses the Content-Type "`application/x.atom+xml`".

2.2 Supported authentication methods

The only supported authentication method in Lifeblog 1.5 is called WSSE authentication, which is similar to the HTTP Digest authentication. Authentication information is included in the HTTP header. WSSE authentication provides authentication that is based on non-clear-text passwords. It also contains a nonce to prevent a replay attacks. A good article about WSSE in Atom can be found in [9].

2.2.1 Authentication header

Authentication information is embedded in HTTP header "X-WSSE":

```
X-WSSE: UsernameToken Username="bob",
PasswordDigest="quR/EWLAV4xLf9Zqyw4pDmfV9OY=",
Nonce="d36e316282959a9ed4c89851497a717f", Created="2003-12-
15T14:43:07Z"
```

The header line contains Username, Nonce, Creation timestamp and Password digest:

- **Username**
  The username of the user

- **Nonce**
  Base64 encoded (cryptographically) random string that should be different for each request.

- **Created**
  Nonce creation timestamp W3DTF format [10].

- **PasswordDigest**
  A secure digest that is calculated from *Created, Nonce, Username* and *Password* using the equation:
  ```
  PasswordDigest = Base64( SHA1(Nonce + CreationTimestamp +
  Password))
  ```

2.2.2 Authenticating the user

To authenticate the user, the server should look for X-WSSE header line. If one is not found, the server should respond with status "401 Unauthorized" containing the header line:

```
WWW-Authenticate: WSSE realm="foo", profile="UsernameToken"
```

In this case, the client should replay the request with appropriate authentication information. Lifeblog 1.5 always includes the X-WSSE header in the requests sent to Atom server, and therefore it is not strictly necessary for the server to implement this to be compatible with Lifeblog 1.5.

When a request with an X-WSSE header arrives at the server, the server should:

1. Confirm that the *Username* is valid

2. Check that the *Nonce* is not too old (*Created* timestamp)

3. Check that the *Nonce* has not been used before by this user (this is a way to protect from the replay attacks)

4. Calculate the *PasswordDigest* using the formula above and confirm that it is identical to the one in the X-WSSE header

If all of the above checks pass, the user can be considered properly authenticated. If not, then the server should respond with status "401 Unauthorized", and optionally include an error description in the response body.

2.3   Bloglist

When the user enters weblog account information in Lifeblog, Lifeblog fetches the list of blogs belonging to the user. To get the blog information, Lifeblog sends a WSSE authenticated HTTP GET request (Ch. 9.3 [1]) to a "connection URL" (entered by the user when setting up their account details in Lifeblog), and expects to get a response that has a Content-Type with a valid Atom MIME type (see 2.1.1). The response body must contain Atom <feed> top level element (Ch. 4 [6]) with Atom <link> elements (Ch. 4.4 [6]) as direct children. For each blog there should at least be links with *rel* attribute "service.post" (connection point for posting to the blog), "service.feed" (where the feed of the blog can be queried) and "alternate" with type="text/html" (HTML representation of the weblog). The *title* attribute of the <link> elements designates the name of the blog and all <link>s with same title point to the same blog.

2.3.1   Blog list example

```
<?xml version="1.0"?>
<feed xmlns="http://purl.org/atom/ns#">

<link type="application/atom+xml" rel="service.post"
href="http://www.sampleblog.com/t/atom/weblog/blog_id=66439" title="testblog"/>

<link type="application/atom+xml" rel="service.feed"
href="http://www.sampleblog.com/t/atom/weblog/blog_id=66439" title="testblog"/>

<link type="text/html" rel="alternate"
href="http://myblog.sampleblog.com/testblog/" title="testblog"/>

<link type="application/atom+xml" rel="service.post"
href="http://www.sampleblog.com/t/atom/weblog/blog_id=77518" title="blog2"/>

<link type="application/atom+xml" rel="service.feed"
href="http://www.sampleblog.com/t/atom/weblog/blog_id=77518" title="blog2"/>
```

```
<link type="text/html" rel="alternate"
href="http://myblog.sampleblog.com/blog2/" title="blog2"/>

</feed>
```

2.4 Posting

Posting in Atom is done by sending an HTTP POST request (Ch. 9.5 [1]) to the PostURI of the blog. PostURI is indicated with link with rel="service.post" in the blog list. The request body contains an Atom <entry> element (Ch. 4.13 [6]) that describes the posted item. The item itself is encoded in the <content> element (Ch. 4.13.10[6]). If the posting is accepted, the server must respond with the status "201 Created", and the response body must contain the Atom <entry> element filled with all the obligatory fields (as specified in the Atom documentation). Lifeblog requires the returned entry to contain at least the <id> of the post when posting non-post resources (see 2.4.1) and a <link> (rel="alternate", type="text/html") pointing to the html-presentation of the post when posting the body text of the post (this is often called "permalink").

Since the Atom API (as it stands in version 0.9 draft) does not support posting of non-post resources (such as non-post images) and certain item types (such as SMS), the Lifeblog posting uses some non-standard features. Care has been taken to ensure compatibility with "pure" Atom (draft) server implementations (as long as the server doesn't mind extra non-Atom elements in the <entry> element), so Lifeblog should still work with those (although with missing functionality).

To specify item captions, Lifeblog uses the Atom <summary> element (Ch. 4.13.9 [6]). When posting the body text, the summary is set to be empty (on PC) or not included at all (on Series 60).

A basic (text) blog post using Atom looks like this:

```
POST http://www.sampleblog.com/t/atom/weblog/blog_id=66439
HTTP/1.1
Accept: text/xml
Content-type: application/xml; charset=utf-8
Authorization: WSSE profile="UsernameToken"
X-WSSE: UsernameToken Username="bob",
PasswordDigest="dQQQqDGpMgxTDLyARIzRzWlP/ds=",
Nonce="Q09NQXRvbS00ODIz", Created="2004-11-16T13:04:04Z"
User-Agent: AtomClient
Host: www.sampleblog.com

<?xml version="1.0" encoding="utf-8"?>
<a:entry xmlns:a="http://purl.org/atom/ns#">
    <a:title type="" mode="escaped">Post title</a:title>
    <a:summary/>
    <a:issued/>
    <a:content type="text/plain" mode="escaped">
        Here is my sample blog post.
    </a:content>
    <a:generator url="http://www.nokia.com/lifeblog">
```

```
            Nokia Lifeblog posting component 1.0.0.7
      </a:generator>
</a:entry>
```

With a response from the server:

```
HTTP/1.1 201 Created
Date: Tue, 16 Nov 2004 13:04:04 GMT
Content-Type: application/atom+xml; charset=utf-8
Vary: Accept-Encoding,User-Agent
Transfer-Encoding: chunked

<?xml version="1.0"?>
<entry xmlns="http://purl.org/atom/ns#">
<title>Post title</title>
<summary>summary of the text</summary>
<issued>2004-11-16T13:04:04Z</issued>
<link type="text/html" rel="alternate"
href="http://myblog.sampleblog.com/testblog/postid=12345"
title="HTML"/>
<id>tag:sampleblog.com,post-12345</id>
</entry>
```

2.4.1 Posting Lifeblog items

When Lifeblog posts a weblog post containing multiple items (such as images, SMSs, MMSs, notes or videos) it breaks the whole post into smaller elements. It first posts all items one at a time, just like a normal post but with a special XML element <standalone> (from namespace "http://sixapart.com/atom/typepad#") embedded in the <entry> element. After posting each item, Lifeblog stores the <id> returned by the server for each item. It then posts the "body text" of the post with <link> elements (with rel="related") with the href attribute set to collected resource ids.

The following example should clarify this mechanism. In this example, we first post a JPEG image followed by a body text.

1. REQUEST:

```
POST http://www.sampleblog.com/t/atom/weblog/blog_id=66439 HTTP/1.1
Accept: text/xml
Content-type: application/xml; charset=utf-8
Authorization: WSSE profile="UsernameToken"
X-WSSE: UsernameToken Username="bob",
PasswordDigest="dQQQqDGpMgxTDLyARIzRzWlP/ds=", Nonce="Q09NQXRvbS00ODIz",
Created="2004-11-16T13:04:04Z"
User-Agent: AtomClient
Host: www.sampleblog.com

<?xml version="1.0" encoding="utf-8"?>
<a:entry xmlns:a="http://purl.org/atom/ns#">
<a:title type="" mode="escaped">681_plane.jpg</a:title>
<a:summary/>
<a:issued/>
```

```
<standalone xmlns="http://sixapart.com/atom/typepad#">1</standalone>
<a:content type="image/jpeg" mode="base64">
/9j/4AAQSkZJRgABAQEASABIAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRof
6Alto9NtfypSaJbT8EywStPoAsjMTKlxPgxPh+aH0yMIaa/3BXPRtTNW/D4fcp+q6eLq+Xyi
pkUV1COeIkc16POR7fdAf//Z
</a:content>
<a:generator url="http://www.nokia.com/lifeblog">Nokia Lifeblog posting
component 1.0.0.7</a:generator>
</a:entry>
```

1. RESPONSE:

```
HTTP/1.1 201 Created
Date: Tue, 16 Nov 2004 13:04:04 GMT
Content-Type: application/atom+xml; charset=utf-8
Vary: Accept-Encoding,User-Agent
Transfer-Encoding: chunked

<?xml version="1.0"?>
<entry xmlns="http://purl.org/atom/ns#">
<title>681_plane.jpg</title>
<summary>This is an image of a plane</summary>
<issued>2004-11-16T05:04:04Z</issued>
<link type="text/html" rel="alternate"
href="http://myblog.sampleblog.com/photos/uncategorized/681_planejpg.html"
title="HTML"/>
<id>tag:sampleblog.com,2003:photo-1213185</id>
</entry>
```

2. REQUEST

```
POST http://www.sampleblog.com/t/atom/weblog/blog_id=66439 HTTP/1.1
Accept: text/xml
Content-type: application/xml; charset=utf-8
Authorization: WSSE profile="UsernameToken"
X-WSSE: UsernameToken Username="bob",
PasswordDigest="0nyuRdLu+hriWNQPUXGBnRqUCfw=", Nonce="Q09NQXRvbS0wMDI5",
Created="2004-11-16T13:04:05Z"
User-Agent: AtomClient
Host: www.sampleblog.com

<?xml version="1.0" encoding="utf-8"?>
<a:entry xmlns:a="http://purl.org/atom/ns#">
<a:title type="" mode="escaped">Image posting</a:title>
<a:summary/>
<a:issued/>
<a:content type="text/html" mode="escaped">
    Here is the &amp;quotbody text&amp;quot of the post.
    &lt;br&gt;This is sent last with links pointing to the posted resource.
</a:content>
<a:generator url="http://www.nokia.com/lifeblog">Nokia Lifeblog posting
component 1.0.0.7</a:generator>
<a:link  rel="related" type="image/jpeg" href="tag:sampleblog.com,2003:photo-
1213185"/>
</a:entry>
```

2. RESPONSE

```
HTTP/1.1 201 Created
Date: Tue, 16 Nov 2004 13:04:04 GMT
Location: http://www.sampleblog.com/t/atom/weblog/blog_id=66439/entry_id=2744356
Content-Type: application/atom+xml; charset=utf-8
Vary: Accept-Encoding,User-Agent
Transfer-Encoding: chunked
```

NOKIA

```
<?xml version="1.0"?>
<entry xmlns="http://purl.org/atom/ns#">
    <title>Image posting</title>
    <summary>Here is the summary.</summary>
    <content mode="escaped">
        &lt;a
href="http://myblog.sampleblog.com/photos/uncategorized/681_planejpg.jpg"&gt;
        &lt;img class="image-full"
src="http://myblog.sampleblog.com/photos/uncategorized/681_planejpg.jpg"
        border="0" alt="This is an image of a plane" /&gt;&lt;/a&gt;
        This is an image of a plane
        &lt;br style="clear: left;" /&gt;
        Here is the &amp;quotbody text&amp;quot of the post.
        &lt;br&gt;This is sent last with links pointing to
        the posted resource.
    </content>
    <issued>2004-11-16T05:04:04Z</issued>
    <link type="text/html" rel="alternate"
href="http://myblog.sampleblog.com/test/2004/11/image_posting.html"
title="HTML"/>
    <id>tag:sampleblog.com,2003:post-2744356</id>
    <link type="application/atom+xml" rel="service.edit"
href="http://www.sampleblog.com/t/atom/weblog/blog_id=66439/entry_id=2744356"
title="Image posting"/>
</entry>
```

If Lifeblog is used to post to a server that does not understand the <standalone> tag, problems should not occur as long as the server supports posting binaries. The normal result is that each of the resources appears first as a separate blog post (with filename as its title) followed by the actual post (with the correct title).

2.4.1.1 Resources in the future

When the Atom W3C Working Group publishes version 1.0 of the Atom API, Lifeblog will implement the SimpleResourcePosting method described by the new Atom protocol.

2.5 Posting special items

In Atom, the type of the posted content is described with the standard IANA "MIME Media Type". However, in Lifeblog there are many items that do not clearly fall in one specific media type. For example both "SMS" and "Note" can be understood as text (mimetype "text/plain"), but the user clearly would expect them to be displayed differently on the weblog. Another rather difficult item is MMS, since it is a collection of files joined by a SMIL file.

To facilitate the handling of these special items, Lifeblog adds additional metadata to the <entry> element. This metadata is from the "Dublin Code Metadata Initiative" (DCMI [11]) and the elements belong to the XML namespace "http://purl.org/dc/elements/1.1/". (prefix "dc" used in below).

2.5.1 SMS

SMS messages are posted as normal <content> with mode="escaped" and type="text/plain"

Special metadata:

```
<dc:type>Text</dc:type>
<dc:format>SMS</dc:format>
```

2.5.2 Note

Notes are posted as normal <content> with mode="escaped" and type="text/plain"

Special metadata:

```
<dc:type>Text</dc:type>
<dc:format>Note</dc:format>
```

2.5.3 MMS

MMS messages are posted by packing all files that make the MMS using the ZIP algorithm. The resulting binary is sent Base64 encoded in a <content> element with mode="base64" and type="application/zip".

Special metadata:

```
<dc:type>Image</dc:type>
<dc:format>MMS</dc:format>
```

It is up to the server to decide what to do with this additional information. In case of an MMS it is suggested that the zip is extracted to a folder of its own and at least a link is given to the SMIL file it contains.

NOKIA

References

1. Hypertext Transfer Protocol (HTTP)
   http://www.w3.org/Protocols/rfc2616/rfc2616.html

2. Extensible Markup Language (XML)
   http://www.w3.org/XML/

3. RFC 3548 - The Base16, Base32, and Base64 Data Encodings
   http://www.faqs.org/rfcs/rfc3548.html

4. FIPS PUB 180-1 Secure Hash Standard
   http://www.itl.nist.gov/fipspubs/fip180-1.htm

5. The AtomAPI
   http://bitworking.org/projects/atom/draft-gregorio-09.html

6. The Atom Syndication Format 0.3 (PRE-DRAFT)
   http://www.mnot.net/drafts/draft-nottingham-atom-format-02.html

7. IETF ATOMPUB Working Group
   http://www.ietf.org/html.charters/atompub-charter.html

8. MIME Media Types
   http://www.iana.org/assignments/media-types/

9. Atom Authentication
   http://www.xml.com/pub/a/2003/12/17/dive.html

10. Date and Time Formats
    http://www.w3.org/TR/NOTE-datetime

11. The Dublin Core Metadata Initiative
    http://www.dublincore.org/

Glossary

HTTP	Hyper text transfer protocol [1]
Atom	Web standard (PRE-DRAFT) for syndicate file format and API for publishing weblog posts
Blog	Web log
XML	Extensible Markup Language [2]
Base64	Encoding used in converting binary data to text [3]
SHA-1	Algorithm for calculating cryptographically secure hashes [4]

Concepts

Item	Lifeblog object that can be attached to the weblog post (e.g. image).
Post	A single weblog entry that might contain multiple items. All pieces of a post appear under one title and publishing date
Non-post resource	Item that is sent to Atom server that is not meant to be shown as a separate post.

 LB001
Issue 1.0 Draft 1.0