Metalink 3.0 Specification (Second Edition)

Namespace Description, Client Behavior, Examples

2007-07-07

http://www.metalinker.org/

Anthony Bryan (anthonybryan@gmail.com) [Author]

Implementations by:

Michael Burford <http://www.getright.com/> [Code: GetRight]
Darius Liktorius <http://www.netcorpinc.com/> [Code: Metalink Generator]
Manuel Subredu <http://simba.packages.ro/> [Code: RoPkg::Metalink]
Tatsuhiro Tsujikawa <http://aria2.sourceforge.net/> [Code: aria2]
Yazsoft <http://www.yazsoft.com/> [Code: Speed Download]
Max Velasques <http://dfast.sourceforge.net/> [Code: wxDownload Fast]
A. Bram Neijt <http://prog.infosnel.nl/metalinks/> [Code: metalink tools]
FDM Team <http://www.freedownloadmanager.org/> [Code: Free Download Manager]
Orbit Team <http://www.orbitdownloader.com/> [Code: Orbit Downloader]
Nils Maier <http://www.downthemall.net> [Code: DownThemAll]
Arne Babenhauserheide <http://www.phex.org> [Code: Phex]
Mathias Berchtold <http://www.smartftp.com> [Code: SmartFTP]
Hampus Wessman <http://hampus.vox.nu/metalink/> [Code: Metalink Editor, Metadl]
Nils Maier <http://www.downthemall.net/> [Code: DownThemAll!]
Neil McNab <http://www.nabber.org/projects/appupdater/> [Code: Appupdater]
Ruben Kerkhof <http://www.rubenkerkhof.com/> [Code: Mirror Manager]
Danny Ayers <http://dannyayers.com/> [Code: GRDDL Transformation]
Patrick Ruckstuhl <http://origo.ethz.ch/> [Code: Origo]

Status of this Specification

This document specifies a protocol for the Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited. The English version of this specification is the only normative version.

Copyright Notice

Abstract

This standard specifies a way to describe location(s) and content for a faster and simpler downloading experience by automating typically advanced features.

Table of Contents

0.0 Acknowledgements

Thanks to the following for their support and encouragement in small and large ways: Darius Liktorius, Michael Burford, Giorgio Maone, Manuel Subredu, Tatsuhiro Tsujikawa, Yazsoft, Max Velasques, Manolo Valdes, Urs Wolfer, A. Bram Neijt, FDM Team, Orbit Team, KGet developers, Nils Maier, Arne Babenhauserheide, Mathias Berchtold, Hampus Wessman, Neil McNab, Danny Ayers, Kristian Weston, Nick Dominguez, Paul Burkhead, Bridget & Ethan Fletcher, Ruben Kerkhof, Hayden Legendre, Matt Domsch, Gary Zellerbach, Gervase Markham, Ryan Cronin, Kees Cook, Josh Colbert, Steve Kleisath, Erin Solari, John Sowder, Tom Mainville, Janie Wargo, Markus Hofmann, James Clark, Tim Bray, Dan Fandrich, Dan Brickley, Dan Connolly, Tim Berners-Lee, Harry Chen, Daniel Stenberg, Adrien Macneil, Louis Suarez-Potts, Patrick Ruckstuhl, Ross Smith, Rahul Sundaram, Jesse Keating, Michał Bentkowski, Andrew Pantyukhin, Judd Vinet, Charles Landemaine, Francis Giannaros, Pascal Bleser, Jeff@BLAG, Yuichiro Nakada, Jereme Hancock, Marcel Hauser, Jeff Covey, Doug Lang, Seth Brown, Alexander Lazic, Jason Green, James Linden, Sebastien Willemijns, Nicolas Alvarez, & especially Juanita Anthony & Zimmy Bryan.

1.0 Introduction

1.1 What is a Metalink?

Metalinks bundle the various ways (FTP/HTTP/P2P) to acquire files into one format for easy content distribution/Electronic Software Distribution (ESD). This can increase download performance, reliability, usability, & efficiency. Any program that downloads files could potentially use it. A Metalink is a Multi Method Metalinker (MMM or M3talinker) document. MMM is a dialect of XML and MMM files must conform to the XML 1.0 specification. "MMM" might also look like "WWW" mirrored.

1.2 Benefits over traditional download methods

- Combines FTP and HTTP with Peer-to-peer (P2P, shared bandwidth).
- Standard unified format that collects links for automatic accelerated (segmented) downloads from multiple sources.
- Automatic load balancing distributes traffic so individual servers are under less strain.
- No Single Point of Failure (SPOF) like FTP or HTTP URLs. More fault tolerant.
- Useful for automatic updating programs when new versions are released.
- No long confusing list of possibly outdated Mirrors and P2P links.
- Makes the download process simpler for users (automatic selection of language, Operating System, location, etc).
- Stores more descriptive and useful information for ESD.
- No separate MD5/SHA-1 file or manual process for verification.
- Uniquely identifies files, so even if all references to it in the Metalink stop working, the same exact file can be found via a P2P or Web search.

- Can finish P2P downloads even if no full seeds are shared.
- For FTP/HTTP, an updated client is needed, but not a separate client like some P2P (For example, BitTorrent (Official client) is a 6.5 megabyte download).

1.3 Goals

- Remain backward compatible with the Web architecture.
- Make downloads faster and cheaper by making mirror/P2P use automatic and integrated.
- Make downloads more reliable and more likely to finish. Higher availability.
- Simplify the download process: Less options and user interaction by default.
- Improve the way that large multi-located files are distributed.
- Make the use of a unique identifier common across many file sharing networks and search engines so it is easy to find the exact same file.
- Keep the Metalink file format as simple as possible, while also storing information that may be useful for content distribution.

1.4 Possible Uses

Metalink was designed for describing the locations of large files that are multi-located (shared via many mirrors and with P2P) so all locations can be used. Support for Metalink would be useful in download managers and Web browsers. It is useful for communities or companies who distribute content over the Internet with multiple servers and methods. Possible uses include software distribution (Operating Systems, games, software updates, etc) and large video or audio file distribution (iTunes, DVDs, lossless audio, compressed audio). It is also useful for making the download process simpler, so the user does not need to select or decide which Operating System, language, or download location they require and also for downloading many related files with one click.

1.5 Version History

This (3.0) is the third version of Metalink and the first "public" version. The first version was similar to a simple text file. The second version from 1998 used XML. The third version added P2P besides the original FTP and HTTP hyperlinks. The fourth version (2.0) lacks some features of chronologically earlier versions, but was much simpler.

NOTE: Metalink 3.0 supports multiple files. Metalink 2.0 does not.

1.6 Implementation

Darius Liktorius (http://www.netcorpinc.com) was the first to implement Metalink 2.0. He modified FlashGot (http://www.flashgot.net), a Firefox extension, to support Metalinks in GetRight (http://www.getright.com). Giorgio Maone of FlashGot provided guidance and direction for modifying his code, and for this he deserves many thanks. Michael Burford added native support for Metalink 3.0 in GetRight 6 (Beta 6+04).

Without GetRight and FlashGot, Metalink would have had to start from scratch. Tatsuhiro Tsujikawa added Metalink 3.0 support to aria2 (http://aria2.sourceforge.net/), the first Unix client. Speed Download (http://www.yazsoft.com/) is the first Mac client to support Metalink 3.0. wxDownload Fast (http://dfast.sourceforge.net/) is the first cross platform (Mac/Unix/Windows) client to support Metalink 3.0. Free Download Manager (Windows) also supports Metalink. So does Orbit Downloader (Windows). Phex was the first Gnutella client with Metalink export. SmartFTP (Windows) was the first FTP client, which supports Metalink for adding files to a transfer queue and verifies their checksum. It is also the first native 64 bit Windows client. Nils Maier has added support to DownThemAll!, a great Firefox extension.

Darius Liktorius also wrote the first Metalink Generator. Manuel Subredu created the first automatic Metalink Generator (http://metalink.packages.ro/) with updates.

Manuel Subredu wrote RoPkg::Metalink and Simba for automating the creation of metalinks. Bram Nejit made metalink tools, the first command line tools for automating metalink generation for the average person, and also patched Bouncer. Hampus Wessman wrote a cross platform Metalink Editor with a GUI. Ruben Kerkhof patched Fedora's MirrorManager.

Dan Brickley did the initial RDF work and Danny Ayers has continued it along with GRDDL.

The World Browser is the first web browser with native Metalink support.

We invite other Download Managers, Web browsers, and P2P (BitTorrent, ed2k, magnet link) clients to support Metalink. We also invite comments from developers and users on ways to improve Metalink. We also look forward to more support on other platforms and more Open Source implementations.

Metalink Announcements (metalink-announce@googlegroups.com) and Metalink Discussion (metalink-discussion@googlegroups.com) are mailing lists for staying up to date, getting involved, and collaborating.

2.0 Metalink Implementation Requirements

2.1 Client

Requirements for integration into Web browsers and download managers:

- **RECOMMENDED:** Multi-threaded (segmented) downloads. (But, almost any program can benefit from multiple URLs, which increase availability).
- Optional: Configurable options/settings (or detection of): language, location, operating system, etc.

- Optional: For <verification>: Verify MD5, SHA-1, SHA-256 checksums. (Optional: OpenPGP).
- Optional: BitTorrent (**Recommended** latest version in the stable branch). (e.g. GetRight, which can download a file with BitTorrent AND FTP/HTTP links).
- Optional: Magnet Links, ed2k links, and other P2P networks.

2.2 Server

A server component to Metalink is unnecessary but would be very useful. This part will keep mirrors up to date, files verified, and will ensure that there are no broken links.

3.0 User, Client, & Site Behavior

3.1 User activity

User clicks on a link to a .metalink file on a Web page, or opens one in an email attachment or locally. This invokes the client program. Client automatically downloads file using as many methods from `<resources>` as is efficient, making the file download faster. This, however, is all transparent to the user and requires no extra input. (More options would of course be available to advanced users, for instance by right clicking the .metalink file). Checksum verification is automated.

3.2 Client (Download Manager) activity

After a user has clicked on a .metalink file, the client parses the file. If *type="dynamic"*, client checks *origin* location for newer *refreshdate*. Client refers to options the user has set indicating language, operating system, and location, and downloads that version of the file if it is specified. (For instance if the user's settings were "pt-BR" for language, "Solaris (SPARC)" for operating system, and "jp" for Japan for location, the client would default to downloading the Brazilian Portuguese version for Solaris (SPARC), using any Japanese mirrors to begin with). The client checks `<resources>` and for `<url>`s with *location* and *preference* (aka priority), and downloads segments from the highest preference first while also testing download speeds. If the client has been set to not download in segments, it will download the file from one server. If that server goes down, it can use other servers listed in the .metalink. File automatically downloads using as many methods from `<resources>` as is efficient or by the programs defaults or settings, making the file download faster. Client can prefer P2P downloads over other methods unless specified. If the client does not natively support a P2P network/method it could pass these links to an external program. Client uses .torrent `<url>` or extracts .torrent info embedded in XML. Alternatively, the client could download the .torrent from a URL specified, if the creator did not wish to embed it. If a BitTorrent download is in progress, but there are no more seeds or no seeds at 100%, then the client will automatically finish the download with another location from `<resources>`. Once a BitTorrent download has completed, it will continue seeding to others until manually stopped or until a pre-configured time. Once download is complete, client uses

<verification> such as <hash> or <signature> to verify file authenticity - verification is already built into BitTorrent, but not regular downloads.

3.2.1 Client (Download Manager) Quirks

All clients support verifying MD5 and SHA-1 checksums. Only aria2 and DownThemAll! support SHA-256.

aria2 and Metadl support partial file checksums aka chunk checksums/repair information.

GetRight only uses the first 16 URLs listed in a .metalink.

Free Download Manager does not support .metalinks listing multiple files.

aria2 and GetRight support FTP, HTTP, and BitTorrent.

aria2 supports filtering downloads from a .metalink by "location", OS, or language by command line options.

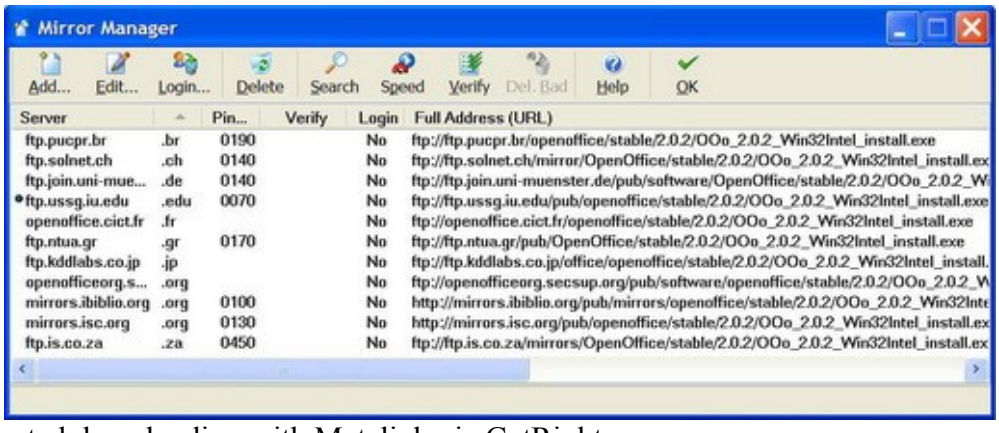
aria2 also supports directory creation.

DownThemAll! And Speed Download support filtering downloads from a .metalink by OS or language by a GUI popup where you are presented with a list of all files and you can select which ones you desire.

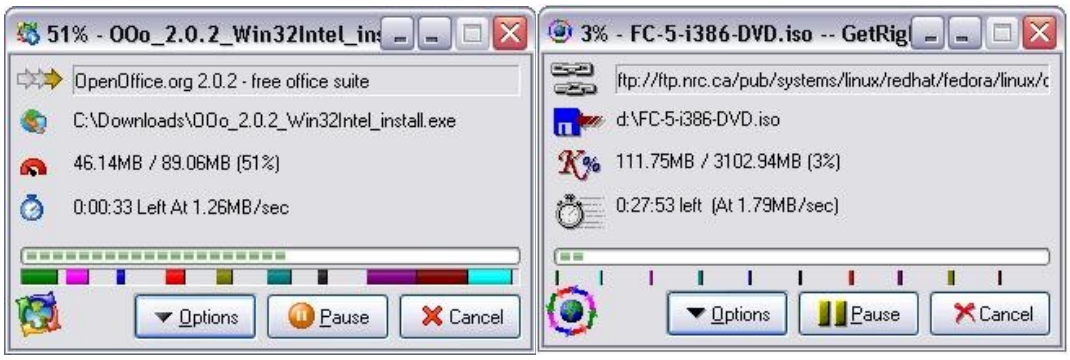
No clients yet support rsync, or ed2k.

3.2.2 Metalink in action (Figure 1 and 2)

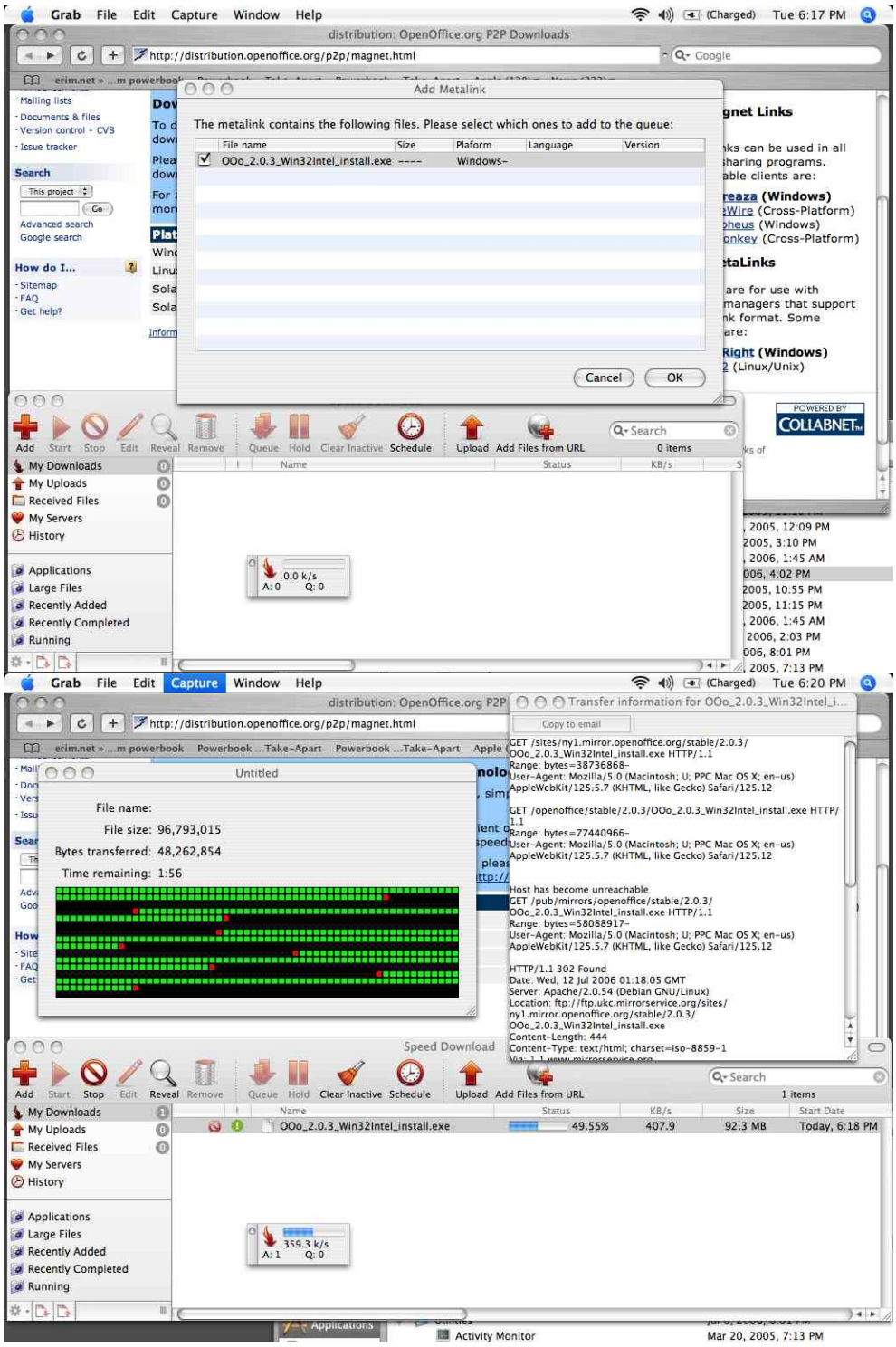
If you are familiar with GetRight and its Mirror Manager, this is what it looks like after being auto-populated by a Metalink. Instead of searching and manually adding these mirror locations, they are automatically used.



Segmented downloading with Metalink via GetRight:



Segmented downloading with a Metalink in Speed Download 4.



3.3 File Distribution Sites

Currently, OpenOffice.org uses Metalink to distribute its free office suite. They are located on the P2P Downloads page. These are automatically generated by Simba.

The following Linux and BSD distributions use Metalink to distribute their ISOs: BLAG Linux, StartCom Linux, Berry Linux, Ubuntu Christian Edition, DesktopBSD, Arch Linux, redWall Firewall, and PC-BSD.

File Distribution sites like http://www.download.com/, http://fileforum.betanews.com/, http://freshmeat.net/, & other aggregating sites like http://filemirrors.com/ could subscribe to an official syndicated feed (Atom or RSS) from software producers that alerts them to new releases of programs or files & the Metalink file that describes their locations.

Sites that serve Metalinks need to add a MIME type for .metalink files of "application/metalink+xml" on their web servers. If you use an older version of Apache, filename.tar.gz.metalink may give an error in browsers. You need to comment out these lines in /etc/apache/http.conf:

```
# AddEncoding x-compress Z
# AddEncoding x-gzip gz tgz
```

4.0 Metalink Element Definitions

4.1 Required Elements

4.1.1 Required Header Element: <metalink>

```
<metalink version="3.0" xmlns="http://www.metalinker.org/">
```

This identifies the XML namespace & differentiates it from other XML vocabularies with possibly similar tags.

4.1.2 Required Body Elements

4.1.2.1 <files> Element, required sub-element of <metalink>

`<metalink>` MUST contain one and only one `<files>`.

4.1.2.2 <file> Element, required sub-element of <files>

`<files>` MUST contain one `<file>` and MAY contain multiple `<file>`s. (In general, Metalink clients will download each <file> listed in a .metalink, but some clients like aria2, DownThemAll!, and Speed Download allow you to manually filter which files you will download).

`<file>` MUST have the attribute *name* containing the file name, as in:


```
<file name="example.ext">.
```

The value used for `name` will be the filename created locally by the download client. This means it will be the output filename, and the names of files listed in the URLs do not need to be related to it.

Directory information can also be contained in the .metalink, as in:

```
<file name="debian-amd64/dists/sarge/Contents-amd64.gz">
```

In this example, a subdirectory `debian-amd64/dists/sarge/` will be created and a file named `Contents-amd64.gz` will be created inside it.

For .metalinks containing multiple files, each <file> should have a unique name attribute. If dealing with files that are identically named at source URLs but have different content (such as http://foo1.com/test and http://foo2.com/test), each file can be given a unique `name` (such as `foo1test` and `foo2test`) or placed in a separate directory (such as `foo1/test` and `foo2/test`).

4.1.2.3 <resources> Element, required sub-element of <file>

`<file>` MUST contain one and only one `<resources>`. *maxconnections* is an optional attribute of `<resources>`, which determines the number of connections used at once for the transfer of that file. A value of "1", "2", "3", etc means that many connections can be opened to that resource. *maxconnections="1"* would turn off segmented downloading. *maxconnections* can also be used as an optional attribute of `<url>` to limit the number of connections to a particular server.

4.1.2.4 <url> Element, required sub-element of <resources>

`<resources>` MUST contain one `<url>` and MAY contain multiple `<url>`s. `<url>` has values of `"ftp"`, `"ftps"`, `"http"`, `"https"`, `"rsync"`, `"bittorrent"`, `"magnet"`, `"ed2k"` for the `type` attribute. The optional *location* attribute is a 2 letter country code for the location of the mirror, as in *location="uk"*. The optional *preference* attribute is a number from 1 to 100 for priority, with 100 used first and 1 used last. Different `<url>`s can have the same preference, i.e. ten mirrors could have *preference="100"* set. `<url>` elements do not need to be in any order. (NOTE: Magnet links must be escaped or the XML will not validate).

4.1.2.5 Example

```
<files>
  <file name="example.ext">
    <resources>
      <url type="ftp"/>
      <url type="http"/>
      <url type="bittorrent"/>
```



```
        <url type="magnet"/>
        <url type="ed2k"/>
      </resources>
    </file>
  </files>
```

A simple example with a minimum of one, but hopefully multiple methods for getting a `<file>` in `<resources>`:

```
<metalink version="3.0" xmlns="http://www.metalinker.org/">
<files>
  <file name="example.ext">
    <resources>
      <url type="ftp">ftp://ftp.example.com/example.ext</url>
      <url type="http">http://www.example.com/example.ext</url>
      <url type="bittorrent">
      http://www.example.net/file.torrent
      </url>
      <url type="magnet"/>
      <url type="ed2k"/>
    </resources>
  </file>
</files>
<metalink>
```

For simplicity also allow:

```
<metalink version="3.0" xmlns="http://www.metalinker.org/">
<files>
  <file name="example.ext">
    <resources>
      <url>http://example.com/example.ext</url>
      <url>http://example2.com/example.ext</url>
      <url>http://example3.com/example.ext.torrent</url>
    </resources>
  </file>
</files>
<metalink>
```

By scanning the beginning of a URL, it can be determined if it is FTP (ftp://), HTTP (http://), rsync (rsync://), magnet (magnet:), ed2k (ed2k://). By examining the end of a URL, you can tell if it is for BitTorrent (.torrent).

4.2 Recommended Elements

4.2.1 Recommended Header Attributes

4.2.1.1 origin Attribute

origin = The original location of this .metalink file. If type is "dynamic" then this is the location where an updated version of the .metalink file will be found.

4.2.1.2 type Attribute

type = *"dynamic"* or *"static"*. Static .metalink files are not updated. Dynamic .metalinks can be expected to contain an updated list of mirrors or resources that the file is available from.

4.2.1.3 pubdate Attribute

pubdate = Original date and time of publishing of the .metalink file. All Metalink date-times conform to the Date and Time Specification of RFC 822, with the exception that the year may be expressed with two characters or four characters (four preferred). Example: Mon, 15 May 2006 00:00:01 GMT

4.2.1.4 refreshdate Attribute

refreshdate = The date and time when a "*dynamic"* .metalink file has been updated

4.2.1.5 generator Attribute

generator = The application used to generate the .metalink file.

4.2.1.6 Header Example

```
<metalink version="3.0" xmlns="http://www.metalinker.org/"
 origin="http://www.example.com/m3/file.metalink"
 type="dynamic" pubdate="Mon, 15 May 2006 00:00:01 GMT"
 refreshdate="Mon, 15 May 2006 12:00:01 GMT"
 generator="manual">
```

4.2.2 Recommended Body Elements

4.2.2.1 <identity> Element, sub-element of <file>

This is the basic identity of the file. For OpenOffice.org 2.0, this would be:

```
<identity>OpenOffice.org</identity>
```

4.2.2.2 <version> Element, sub-element of <file>

This is the version of the file. For OpenOffice.org 2.0, this would be:


```
<version>2.0</version>
```

4.2.2.3 <verification> Element, sub-element of <file>

This contains `<hash>` and `<signature>` elements which are used to verify a file. Hashes can be used to uniquely identify a file. A signature verifies who a file is from.

The valid values for the *type* attribute for `<hash>` are: *md4 md5 sha1 sha256 sha384 sha512 rmd160 tiger crc32.* All hashes are in lowercase hexadecimal format. *md5* and *sha1* are the most commonly used hashes, and the main ones that Metalink clients currently support. More than one hash can be used:

```
<verification>
  <hash type="md5">example-md5-hash</hash>
  <hash type="sha1">example-sha1-hash</hash>
</verification>
```

Besides the above full file checksums, partial file checksums can also be used. These are also called chunk checksums or repair information. They allow corrupted downloads to be fixed and downloads in progress to be auto-corrected. They can also be used to see if some mirrors are providing false information, and then those mirrors can be banned or discarded. It is good to include a full file checksum as well, for clients that only support them and to compare against "official" checksums. Currently, the GUI Metalink Editor and the metalink tools support this with the "-d sha1pieces" option. The format for chunk checksums is:

```
<verification>
  <hash type="sha1">example-sha1-hash</hash>
  <pieces type="sha1" length="262144">
    <hash piece="0">example-sha1-hash</hash>
    <hash piece="1">example-sha1-hash</hash>
  </pieces>
</verification>
```

Also, PGP signatures can be embedded with `<signature type="pgp">` and can contain an optional `file` attribute which references another file (for example, `<file name="linux.sign">`) listed in the Metalink as so:

```
<verification>
  <signature type="pgp" file="linux.sign">
  -----BEGIN PGP SIGNATURE-----
  Version: GnuPG v1.4.2.2 (GNU/Linux)
  Comment: See http://www.kernel.org/signature.html for info

  iD8DBQBElLc9yGugalF9Dw4RAsplAJ9Vhjp8IgkMBdGdiYygXtYBcKZ6GwCffyTu

  +gY8wsoFmSdAU6UiqakOcyo=
  =hcew
  -----END PGP SIGNATURE-----
  </signature>
```



```
</verification>
```

4.2.2.4 <size> Element, sub-element of <file>

This is the size of the file in bytes. If a mirror reports a file size that is different from the size reported in the .metalink, it should not be used. If multiple sources are used, all file sizes should match.

4.3 Optional Body Elements

4.3.1 <url> Element attributes

4.3.1.1 preference Attribute

"*preference*" is an optional attribute of `<url>` under `<resources>` , the priority of a URL. Valid values are numbers from 1-100 that calls for one resource to be used before another. The higher the number, the more preferable that resource is, that is, resources with a higher preference number will be used first. For instance, we want to use P2P or a certain mirror first, so we set it at 100. Or we may want to pull from an FTP/HTTP mirror more for increased speed if P2P is slow. If more than one resource has the same "*preference*", they can all be used. If "*preference*" is unspecified, it is assumed to have the value of 1, that is, the lowest preference.

4.3.1.2 location Attribute

"*location*" is an optional attribute of `<url>` under `<resources>`, which contains the ISO 3166-1 alpha-2 two letter country code for where a server is located. These are similar to two letter top level domain country codes, but are not always the same.

4.3.1.3 maxconnections Attribute

"maxconnections" is an optional attribute of `<url>` under `<resources>`, which determines the number of connections to a mirror server. A value of "1", "2", "3", etc means that many connections can be opened to that resource. *"maxconnections"* is also an optional attribute of `<resources>` (`<resources maxconnections="1">` would turn off segmented downloads. By default, if *maxconnections* is not included, it is limited by the client's default settings or what the user has set them to.

4.3.2 <description> Element, sub-element of <file>

This is a text description of the file.

4.3.3 <logo> Element, sub-element of <file>

This is a location for a graphic logo for the file or program.

4.3.4 <tags> Element, sub-element of <file>

Tags that describe the file in a few words. Tags are separated by commas.

4.3.5 <language> Element, sub-element of <file>

The language the file is in, per ISO-639/3166. "en-US" for Standard American English, "en-GB" for British English, "fr" for French, "de" for German, "zh-Hans" for Chinese (Simplified), "zh-Hant" for Chinese (Traditional), etc. By default, a client will all files listed in a .metalink. In the future, they should only download files in the user's language (set as an option in the client or detected by it). But, there should be options for advanced users to download other files.

4.3.6 <os> Element, sub-element of <file>

This contains information on the required Operating System and architecture, if the file is an application. For example: Source, BSD-x86, BSD-x64, Linux-x86, Linux-x64, Linux-ia64, Linux-alpha, Linux-arm, Linux-hppa, Linux-m68k, Linux-mips, Linux-mipsel, Linux-PPC, Linux-PPC64, Linux-s390, Linux-SPARC, MacOSX-PPC, MacOSX-Intel, MacOSX-UB, Solaris-SPARC, Solaris-x86, Windows-x86, Windows-x64, Windows-ia64. By default, a client will download all files listed in a .metalink. In the future, they should only download files for the user's Operating System (set as an option in the client or detected by it). There should be options for advanced users to download other files though.

4.3.7 <mimetype> Element, sub-element of <file>

MIME type of the file.

4.3.8 <relations> Element, sub-element of <file>

This lists other files that are closely related to this file, such as a version in another language, for another Operating System, or the same file compressed with a different program.

4.3.9 <releasedate> Element, sub-element of <file>

This describes when the file was released (not when the .metalink file was created).

4.3.10 <changelog> Element, sub-element of <file>

This lists the changes between this version of the file and the last.

4.3.11 <publisher> Element, sub-element of <file> and/or <metalink>

This is the publisher of the file the Metalink points to.


```
<publisher>
  <name>Arch Linux</name>
  <url>http://www.archlinux.org/</url>
</publisher>
```

4.3.12 <copyright> Element, sub-element of <file>

This contains the Copyright of the file.

4.3.13 <license> Element, sub-element of <file>

The license the file was released under. Such as: Shareware, Commercial, GPL, BSD, Creative Commons, etc.

```
<license>
  <name>GPL</name>
  <url>http://www.gnu.org/copyleft/gpl.html</url>
</license>
```

4.3.14 <multimedia> Element, sub-element of <file>

For Audio and Audio/Video files. Optional sub-elements include <audio>, <video> for listing <bitrate>, <duration>, <codec>, <resolution>, <artist>, <album> for example:

```
<multimedia>
  <audio>
    <bitrate>320</bitrate>
    <codec>MP3</codec>
  </audio>
  <video>
    <codec>DivX 5.1</codec>
    <duration>55:23</duration>
    <resolution>512 x 384</resolution>
  </video>
</multimedia>
```

4.3.15 <screenshot> Element, sub-element of <file>

Contains `<url>` for screenshots of the application.

4.3.16 <upgrade> Element, sub-element of <file>

The action to be performed when a previous version is already installed. Some programs need older versions uninstalled before installing new ones, and some do not. Could be "install", "uninstall, reboot, install", or "uninstall, install".

4.3.17 <bittorrent> encoding/decoding information

There are two ways to deal with BitTorrent: externally linked, or extracted and placed in XML tags. (1) If the creator does not wish to embed the torrent, it can be linked with <url>http://example.com/example.torrent</url> to an external torrent. (2) Information can also be extracted from the .torrent & stored in the Metalink such as <tracker>, <hash>, <size>, <pubdate>, <piecelength>, and <pieces>. This makes BitTorrent information much easier to manipulate. Method (1) is preferred, because (2) has not been implemented in clients. If BitTorrent is used with Metalink, it is beneficial to use the same chunk size and hash with both.

Examples:

(1) Externally linked:

```
<resources>
  <url type="bittorrent">http://www.example.com/example.torrent</url>
</resources>
```

(2) .torrent information extracted and placed in XML tags:

```
<bittorrent preference="100">
  <announce>http://www.example.com:6969/announce</announce>
  <file name="filename.ext">
  <size>262144</size>
  <verification>
    <hash type="sha1">example-sha1-hash</hash>
    <pieces type="sha1" length="262144">
      <hash piece="0">example-sha1-hash</hash>
      <hash piece="1">example-sha1-hash</hash>
    </pieces>
  </verification>
  </file>
</bittorrent>
```

5.0 Guidelines & Implementation Checklist

5.1 Guidelines

These are not rules that have to be followed, but just general ideas that might not hurt. Please suggest more, or feel free to disagree.

Drop dead/inactive/incomplete links.
Use most preferable and hopefully fastest mirrors based on "location", "preference", and speed.
Rely on P2P if available, unless download speed goes below a threshold
Don't use multiple connections on slow servers and respect maxconnections.
Preferably use one segment per mirror (Max: 2).

Switch to faster mirrors, if speed goes below threshold.
If partial checksums (chunk checksums) are available, check those coming from FTP/HTTP to identify bad Mirrors (even if you don't download from BitTorrent).
Compare checksums. If they match, do nothing. If not, give short error message/warning (or if your program lists completed downloads, display that the checksums don't match) & maybe give choice to re-download.
If MD5 and SHA-1 checksums are present, use SHA-1.

5.2 Implementation Checklist

None of these are required but they are recommended.

Does your application support segmented downloads?
Does it support verifying full file checksums once the download is complete?
Does it support verifying partial checksums for segments downloaded from FTP/HTTP?
Does it support PGP signatures?
Does it support checking the origin URL in the Metalink header for finding updated .metalinks?
Does it process .metalink files that it downloads automatically?
Does it recognize all files contained in a .metalink? Directory information?
Does it allow using metalinks that are on the local filesystem?
Does your application associate .metalink files with itself, so when they are double clicked they will start downloading?
Does your application show a list of "Finished Downloads"? If so, it might be useful to have some type of small graphical indicator (like an orange "!") to show that a downloaded file's checksum does not match the expected checksum.

Appendix A : Example .metalink files

Updated example .metalinks are available from http://www.metalinker.org/samples.html and Metalink @ Packages Resources.

A.1 Linux Kernel 2.6.16.19: linux-2.6.16.19.tar.bz2.metalink

This was generated by Manuel Subredu's Metalink generating code. It originally contained all kernel.org mirrors, but was edited for space reasons to only contain a few.

```
<?xml version="1.0" encoding="UTF-8"?>
<metalink version="3.0" xmlns="http://www.metalinker.org/"
  origin="http://metalink.packages.ro/download/download/kernel/linux-2.
6.16.19.tar.bz2.metalink"
  type="static" pubdate="2006-06-09-18:56:57"
  generator="Metalink Gen - http://metalink.packages.ro"
  refreshdate="2006-06-09-18:56:57">

  <publisher>
    <name>Package resources</name>
    <url>http://www.packages.ro</url>
  </publisher>
  <license>
    <name>GPL</name>
    <url>http://www.gnu.org/copyleft/gpl.html</url>
  </license>
  <description>Linux kernel</description>
  <tags>kernel, linux</tags>
  <identity>linux-2.6.16.19.tar.bz2</identity>
  <version>2.6.16.19</version>

  <files>
   <file name="linux-2.6.16.19.tar.bz2">

     <os>Linux-x86</os>

    <size>40836905</size>
    <verification>
     <hash type="md5">b1e3c65992b0049fdbee825eb2a856af</hash>
    </verification>
    <resources>
      <url type="http"
           location="RO"
           preference="10">
       http://ftp.roedu.net/mirrors/ftp.kernel.org/pub/linux/kernel/v2.
6/linux-2.6.16.19.tar.bz2
      </url>
      <url type="http"
           location="AT"
           preference="10">
       http://ftp.at.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.19.t
ar.bz2
      </url>
      <url type="http"
           location="al"
```



```
           preference="10">
       http://ftp.al.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.19.t
ar.bz2
      </url>
      <url type="http"
           location="ad"
           preference="10">
       http://ftp.ad.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.19.t
ar.bz2
      </url>
      <url type="http"
           location="aq"
           preference="10">
       http://ftp.aq.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.19.t
ar.bz2
      </url>
      <url type="http"
           location="ag"
           preference="10">
       http://ftp.ag.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.19.t
ar.bz2
      </url>
      <url type="http"
           location="ar"
           preference="10">
       http://ftp.ar.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.19.t
ar.bz2
      </url>
      <url type="http"
           location="am"
           preference="10">
       http://ftp.am.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.19.t
ar.bz2
      </url>
    </resources>
   </file>
  </files>
</metalink>
```

A.2 Nero Ultra Edition 7: nero-7.0.1.4b.metalink

NOTE: If this is not the current version of Nero, these files may be unavailable.

This example is for the popular CD/DVD authoring program called Nero. This includes BitTorrent links, multiple FTP, and HTTP hyperlinks for the English, Chinese (Simplified), and German versions for Windows. The client will only download the version for the user's language (which will be specified in the client) with this single click Metalink. For example, a German user clicks the Metalink and the client automatically downloads only the German version, or a Chinese (Simplified) user clicks it and it only downloads the Chinese (Simplified) version, etc.

The current Nero download process is at http://www.nero.com/nero7/enu/nero7-up.php . Mirrors are available from another link at http://www.nero.com/nero7/enu/mirror.php?pak_lang=eng

and http://www.hspeed.net/en/nero-mirror.php (which also has BitTorrent links). Information on the separate MD5 verification is at http://www.nero.com/nero7/enu/Nero_MD5_Verifier.html .

```
<?xml version="1.0" encoding="UTF-8"?>
<metalink version="3.0" xmlns="http://www.metalinker.org/"
  origin="http://www.nero.com/mmm/nero-7.0.1.4b.metalink"
  type="static" pubdate="2005-12-22-22:04:25"
  refreshdate="2005-12-23-03:24:18">

<publisher>
  <name>Nero AG</name>
  <url>http://www.nero.com/</url>
</publisher>
<copyright>Copyright 2006 Nero AG / Nero Inc.</copyright>
<releasedate>2005-12-22</releasedate>
<description>Nero Ultra Edition 7 - CD/DVD Authoring suite
</description>
<tags>Nero, CD, DVD, burning, authoring</tags>
<identity>Nero Ultra Edition</identity>


<files>
  <file name="Nero-7.0.1.4b_eng.exe">
    <version>7.0.1.4b</version>
    <language>en-US</language>
    <os>Windows-x86</os>
    <size>106797808</size>
    <verification>
      <hash type="md5">b86eaee3dc7f511c7b93cddb1f1bcaac</hash>
    </verification>
    <resources>
      <url type="bittorrent" preference="100">
ftp://nero-
mirror.hspeed.net/software/Nero7/Nero-7.0.1.4b_eng.exe.torrent
      </url>
      <url type="ftp" location="us" preference="80">
ftp://ftp2.usw.nero.com/software/nero7/Nero-7.0.1.4b_eng.exe
      </url>
      <url type="ftp" location="de" preference="40">
ftp://nero-mirror.com/software/Nero7/Nero-7.0.1.4b_eng.exe
      </url>
      <url type="ftp" location="de" preference="40">
ftp://nero-mirror.hspeed.net/software/Nero7/Nero-7.0.1.4b_eng.exe
      </url>
      <url type="http" location="us" preference="80">
http://httpdl2.usw.nero.com/software/nero7/Nero-7.0.1.4b_eng.exe
      </url>
    </resources>
  </file>

<file name="Nero-7.0.1.4b_chs.exe">
    <version>7.0.1.4b</version>
    <language>zh-Hans</language>
    <os>Windows-x86</os>
    <size>112296416</size>
    <verification>
```



```
      <md5>cccd7f891ff81b30b9152479d2efcda2</md5>
    </verification>
    <resources>
      <url type="bittorrent" preference="100">
ftp://nero-
mirror.hspeed.net/software/Nero7/Nero-7.0.1.4b_chs.exe.torrent
      </url>
      <url type="ftp" location="de" preference="40">
ftp://nero-mirror.hspeed.net/software/Nero7/Nero-7.0.1.4b_chs.exe
      </url>
    </resources>
  </file>

<file name="Nero-7.0.1.4b_deu.exe">
    <identity>Nero Ultra Edition</identity>
    <version>7.0.1.4b</version>
    <language>de</language>
    <os>Windows-x86</os>
    <size>112422536</size>
    <verification>
      <hash type="md5">44b04c2b0a49ec59da26706dfb969158</hash>
    </verification>
    <resources>
      <url type="bittorrent" preference="100">
ftp://nero-
mirror.hspeed.net/software/Nero7/Nero-7.0.1.4b_deu.exe.torrent
      </url>
      <url type="ftp" location="us" preference="80">
ftp://ftp2.usw.nero.com/software/nero7/Nero-7.0.1.4b_deu.exe
      </url>
      <url type="http" location="us" preference="80">
http://httpdl2.usw.nero.com/software/nero7/Nero-7.0.1.4b_deu.exe
      </url>
    </resources>
  </file>

</files>
</metalink>
```


Appendix B : Metalink Schema

Note: This schema does not contain information for chunk checksums.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.metalinker.org"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.metalinker.org" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="metalink" type="metalinkType">
		<xs:annotation>
			<xs:documentation>Comment describing your root
element</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:complexType name="filesType">
		<xs:sequence>
			<xs:element name="file" type="fileType"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="metalinkType">
		<xs:sequence>
			<xs:element name="files" type="filesType"/>
		</xs:sequence>
		<xs:attribute name="origin" type="xs:string"/>
		<xs:attribute name="type">
			<xs:simpleType>
				<xs:restriction base="xs:string">
					<xs:enumeration value="dynamic"/>
					<xs:enumeration value="static"/>
				</xs:restriction>
			</xs:simpleType>
		</xs:attribute>
		<xs:attribute name="pubdate" type="xs:dateTime"/>
		<xs:attribute name="refreshdate" type="xs:dateTime"/>
		<xs:attribute name="generator" type="xs:string"/>
	</xs:complexType>
	<xs:complexType name="fileType">
		<xs:sequence>
			<xs:element name="resources" type="resourcesType"/>
			<xs:element name="identity" type="xs:string"/>
			<xs:element name="version" type="xs:string"/>
			<xs:element name="verification">
				<xs:complexType>
					<xs:sequence>
```



```
                                                <xs:element name="hash"
maxOccurs="unbounded">
                                                        <xs:complexType>
                                                                <xs:attribute
name="type">
        <xs:simpleType>
        <xs:restriction base="xs:string">
                <xs:enumeration value="md4"/>
                <xs:enumeration value="md5"/>
                <xs:enumeration value="sha1"/>
                <xs:enumeration value="sha256"/>
                <xs:enumeration value="sha384"/>
                <xs:enumeration value="sha512"/>
                <xs:enumeration value="rmd160"/>
                <xs:enumeration value="tiger"/>
                <xs:enumeration value="crc32"/>
        </xs:restriction>
        </xs:simpleType>
        </xs:attribute>
                                                        </xs:complexType>
                                                </xs:element>
                                                <xs:element name="signature">
                                                        <xs:complexType>
                                                                <xs:attribute
name="type" type="xs:string"/>
                                                                <xs:attribute
name="file" type="xs:string"/>
                                                        </xs:complexType>
                                                </xs:element>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
```



```
                        <xs:element name="size" type="xs:string"/>
                        <xs:element name="description" type="xs:string"/>
                        <xs:element name="logo" type="xs:string"/>
                        <xs:element name="tags" type="xs:string"/>
                        <xs:element name="lenguage" type="xs:string"/>
                        <xs:element name="os" type="xs:string"/>
                        <xs:element name="mimetype" type="xs:string"/>
                        <xs:element name="relations" type="xs:string"/>
                        <xs:element name="releasedate" type="xs:dateTime"/>
                        <xs:element name="changelog" type="xs:string"/>
                        <xs:element name="publisher" type="publisherType"/>
                        <xs:element name="license" type="licenseType"/>
                        <xs:element name="multimedia" type="multimediaType"/>
                        <xs:element name="screenshot" type="xs:string"/>
                        <xs:element name="upgrade" type="xs:string"/>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string"/>
        </xs:complexType>
        <xs:complexType name="resourcesType">
                <xs:sequence>
                        <xs:element name="url" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:attribute name="type">
                                                <xs:simpleType>
                                                        <xs:restriction
base="xs:string">

        <xs:enumeration value="ftp"/>

        <xs:enumeration value="ftps"/>

        <xs:enumeration value="http"/>

        <xs:enumeration value="https"/>

        <xs:enumeration value="rsync"/>

        <xs:enumeration value="bittorrent"/>

        <xs:enumeration value="magnet"/>

        <xs:enumeration value="ed2k"/>
                                                        </xs:restriction>
                                                </xs:simpleType>
                                        </xs:attribute>
                                        <xs:attribute name="location">
```



```
                                                    <xs:simpleType>
                                                        <xs:restriction
base="xs:string">

            <xs:maxLength value="2"/>
                                                        </xs:restriction>
                                                    </xs:simpleType>
                                                </xs:attribute>
                                                <xs:attribute name="preference"
type="xs:int"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                                <xs:complexType name="licenseType">
                                    <xs:sequence>
                                        <xs:element name="name" type="xs:string"/>
                                        <xs:element name="url" type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                                <xs:complexType name="publisherType">
                                    <xs:sequence>
                                        <xs:element name="name" type="xs:string"/>
                                        <xs:element name="url" type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                                <xs:complexType name="audioType">
                                    <xs:sequence>
                                        <xs:element name="bitrate" type="xs:string"/>
                                        <xs:element name="codec" type="xs:string"/>
                                        <xs:element name="duration" type="xs:string"/>
                                        <xs:element name="resolution" type="xs:string"/>
                                        <xs:element name="artist" type="xs:string"/>
                                        <xs:element name="album" type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                                <xs:complexType name="videoType">
                                    <xs:sequence>
                                        <xs:element name="bitrate" type="xs:string"/>
                                        <xs:element name="codec" type="xs:string"/>
                                        <xs:element name="duration" type="xs:string"/>
                                        <xs:element name="resolution" type="xs:string"/>
                                        <xs:element name="artist" type="xs:string"/>
                                        <xs:element name="album" type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
```



```
        <xs:complexType name="multimediaType">
                <xs:sequence>
                        <xs:element name="audio" type="audioType"
maxOccurs="unbounded"/>
                        <xs:element name="video" type="videoType"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

Appendix C : Full Copyright Statement