Network Working Group	A. Bryan	
Internet-Draft	T. Tsujikawa	
Intended status: Standards Track	N. McNab	
Expires: August 16, 2010		
	P. Poeml	
	Novell, Inc.	
	February 12, 2010	

TOC

The Metalink Download Description Format
draft-bryan-metalink-28

Abstract

This document specifies Metalink, an XML-based download description format. Metalink describes download locations (mirrors), cryptographic hashes, and other information. Clients can transparently use this information to reliably transfer files.

Status of this Memo

Copyright Notice

Table of Contents

---

1. Introduction

TOC

Metalink is a document format based on Extensible Markup Language (XML) that describes a file or list of files to be downloaded from a server. Metalinks can list a number of files, each with an extensible set of attached metadata. Each listed file can have a description, multiple cryptographic hashes, and a list of Uniform Resource Identifiers (URIs) that it is available from.

Often, identical copies of a file are accessible in multiple locations on the Internet over a variety of protocols, such as File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), and Peer-to-Peer (P2P). In some cases, users are shown a list of these multiple download locations (mirror servers) and must manually select one based on geographical location, priority, or bandwidth. This is done to distribute the load across multiple servers, and to give human users the opportunity to choose a download location that they expect to work best for them.

At times, individual servers can be slow, outdated, or unreachable, but this can not be determined until the download has been initiated. This can lead to the user canceling the download and needing to restart it. During downloads, errors in transmission can corrupt the file. There are no easy ways to repair these files. For large downloads this can be especially troublesome. Any of the number of problems that can occur during a download lead to frustration on the part of users, and bandwidth wasted with retransmission.

Knowledge about availability of a download on mirror servers can be acquired and maintained by the operators of the origin server, or by a third party. This knowledge, together with cryptographic hashes, digital signatures, and more, can be stored in a machine-readable

Metalink file. The Metalink file can transfer this knowledge to the user agent, which can peruse it in automatic ways or present the information to a human user. User agents can fall back to alternate mirrors if the current one has an issue. Thereby, clients are enabled to work their way to a successful download even under adverse circumstances. All this can be done transparently to the human user and the download is much more reliable and efficient. In contrast, a traditional HTTP redirect to one mirror conveys only comparatively minimal information - a referral to a single server, and there is no provision in the HTTP protocol to handle failures.
Other features that some clients provide include multi-source downloads, where chunks of a file are downloaded from multiple mirrors (and optionally, Peer-to-Peer) simultaneously, which frequently results in a faster download. Metalinks can leverage HTTP, FTP and Peer-to-Peer protocols together, because regardless over which protocol the Metalink was obtained, it can make a resource accessible through other protocols. If the Metalink was obtained from a trusted source, included verification metadata can solve trust issues when downloading files from replica servers operated by third parties. Metalinks also provide structured information about downloads that can be indexed by search engines.
[[Discussion of this draft should take place on apps-discuss@ietf.org. Past discussion has gone on at the Metalink discussion mailing list located at metalink-discussion@googlegroups.com / http://groups.google.com/group/metalink-discussion .]]

1.1. Examples

TOC

A brief, Metalink Document that describes a single file:

```
<?xml version="1.0" encoding="UTF-8"?>
<metalink xmlns="urn:ietf:params:xml:ns:metalink">
  <file name="example.ext">
    <size>14471447</size>
    <url>ftp://ftp.example.com/example.ext</url>
    <url>http://example.com/example.ext</url>
    <metaurl mediatype="torrent">
    http://example.com/example.ext.torrent</metaurl>
  </file>
</metalink>
```

A more extensive, Metalink Document that describes two files:

```
<?xml version="1.0" encoding="UTF-8"?>
<metalink xmlns="urn:ietf:params:xml:ns:metalink">
  <published>2009-05-15T12:23:23Z</published>
  <file name="example.ext">
    <size>14471447</size>
    <identity>Example</identity>
    <version>1.0</version>
    <language>en</language>
    <description>
    A description of the example file for download.
    </description>
    <hash type="sha-256">f0ad929cd259957e160ea442eb80986b5f01...</hash>
    <url location="de"
         priority="1">ftp://ftp.example.com/example.ext</url>
    <url location="fr"
         priority="1">http://example.com/example.ext</url>
    <metaurl mediatype="torrent"
         priority="2">http://example.com/example.ext.torrent</metaurl>
  </file>
  <file name="example2.ext">
    <size>14471447</size>
    <identity>Example2</identity>
    <version>1.0</version>
    <language>en</language>
    <description>
    Another description for a second file.
    </description>
    <hash type="sha-256">2f548ce50c459a0270e85a7d63b2383c5523...</hash>
    <url location="de"
         priority="1">ftp://ftp.example.com/example2.ext</url>
    <url location="fr"
         priority="1">http://example.com/example2.ext</url>
    <metaurl mediatype="torrent"
         priority="2">http://example.com/example2.ext.torrent</metaurl>
  </file>
</metalink>
```

1.2. Namespace and Version

TOC

The XML Namespaces URI [REC-xml-names] (Hollander, D., Bray, T., Tobin, R., and A. Layman, “Namespaces in XML 1.0 (Third Edition),” December 2009.) for the XML data format described in this specification is:
urn:ietf:params:xml:ns:metalink

For convenience, this data format may be referred to as "Metalink", which this specification uses internally.

1.3. Notational Conventions

TOC

This specification describes conformance of Metalink Documents. Additionally, it places some requirements on Metalink Processors. This specification uses the namespace prefix "metalink:" for the Namespace URI identified in Section 1.2 (Namespace and Version), above. Note that the choice of namespace prefix is arbitrary and not semantically significant.
Metalink is specified using terms from the XML Infoset [REC-xml-infoset] (Cowan, J. and R. Tobin, "XML Information Set (Second Edition)," February 2004.). However, this specification uses a shorthand for two common terms: the phrase "Information Item" is omitted when naming Element Information Items and Attribute Information Items. Therefore, when this specification uses the term "element," it is referring to an Element Information Item in Infoset terms. Likewise, when it uses the term "attribute," it is referring to an Attribute Information Item.
Some sections of this specification are illustrated with fragments of a non-normative RELAX NG Compact schema [RELAX-NG] (Clark, J., "RELAX NG Compact Syntax," December 2001.). However, the text of this specification provides the definition of conformance. A complete schema appears in Appendix B (RELAX NG Compact Schema).
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.), as scoped to those conformance targets.

---

2. Metalink Documents

TOC

This specification describes Metalink Documents.
A Metalink Document describes a file or group of files, how to access them, and metadata that identifies them. Its root is the metalink:metalink (The "metalink:metalink" Element) element.

```
namespace metalink = "urn:ietf:params:xml:ns:metalink"
start = metalinkMetalink
```

Metalink Documents are specified in terms of the XML Information Set, serialized as XML 1.0 [REC-xml] (Yergeau, F., Paoli, J., Bray, T., Sperberg-McQueen, C., and E. Maler, "Extensible Markup Language (XML)

1.0 (Fifth Edition),” November 2008.) and identified with the "application/metalink4+xml" media type.
Metalink Documents MUST be well-formed XML. This specification does not define a Document Type Definition (DTD) for Metalink Documents, and hence does not require them to be valid (in the sense used by XML).
Metalink allows the use of Internationalized Resource Identifiers (IRIs), encoded according to [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” January 2005.). Every URI [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.) is also an IRI, so a URI may be used wherever below an IRI is named. There is one special consideration: when an IRI that is not also a URI is given for dereferencing, it MUST be mapped to a URI using the steps in Section 3.1 of [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” January 2005.).
Any element defined by this specification MAY have an xml:lang attribute, whose content indicates the natural language for the element and its descendents. The language context is only significant for elements and attributes declared to be "Language-Sensitive" by this specification. Requirements regarding the content and interpretation of xml:lang are specified in XML 1.0 (Yergeau, F., Paoli, J., Bray, T., Sperberg-McQueen, C., and E. Maler, “Extensible Markup Language (XML) 1.0 (Fifth Edition),” November 2008.) [REC-xml], Section 2.12.

```
metalinkCommonAttributes =
   attribute xml:lang { metalinkLanguageTag }?,
   undefinedAttribute*
```

All leading and trailing whitespace is part of the element content, and MUST NOT be ignored. Consequently, it is disallowed for elements where the defined type does not allow whitespace, such as dates, integers, or IRIs. Some XML-generating implementations erroneously insert white space around values by default, and such implementations will generate invalid Metalink Documents.
Metalink Documents that do not follow this specification are invalid and SHOULD NOT be used by Metalink Processors.
Metalink is an extensible format. See Section 5 (Extending Metalink) of this document for a full description of how Metalink Documents can be extended.

---

3. Common Metalink Constructs

TOC

Many Metalink elements share common structures. This section defines those structures and their requirements for convenient reference by the appropriate element definitions.

When an element is identified as being a particular kind of construct, it inherits the corresponding requirements from that construct's definition in this section.

3.1. Text Constructs

TOC

A Text construct contains human-readable text, usually short in length.

```
metalinkTextConstruct =
   metalinkCommonAttributes,
   text
```

For example, a metalink:description (The "metalink:description" Element) with text content:

```
...
<description>
A description of the example file for download.
</description>
...
```

The content of the Text construct MUST NOT contain child elements. Such text is intended to be presented to humans in a readable fashion. Thus, white space could be collapsed (including line breaks) and text could be displayed using typographic techniques such as justification and proportional fonts.

3.2. Date Constructs

TOC

A Date construct is an element whose content MUST conform to the "date-time" production in [RFC3339] (Klyne, G. and C. Newman, “Date and Time on the Internet: Timestamps,” July 2002.). In addition, an uppercase "T" character MUST be used to separate date and time, and an uppercase "Z" character MUST be present in the absence of a numeric time zone offset.

```
metalinkDateConstruct =
   metalinkCommonAttributes,
   xsd:dateTime
```

Such date values happen to be compatible with the following specifications: [ISO.8601.1988] (International Organization for Standardization, “Data elements and interchange formats - Information interchange - Representation of dates and times,” June 1988.),

[NOTE-datetime-19980827] (Wolf, M. and C. Wicksteed, “Date and Time Formats,” August 1998.), and [REC-xmlschema-2-20041028] (Malhotra, A. and P. Biron, “XML Schema Part 2: Datatypes Second Edition,” October 2004.).
Example Date constructs:

```
...
<updated>2009-05-15T18:30:02Z</updated>
...
<updated>2009-05-15T18:30:02.25Z</updated>
...
<updated>2009-05-15T18:30:02+01:00</updated>
...
<updated>2009-05-15T18:30:02.25+01:00</updated>
...
```

---

4. Metalink Element Definitions

TOC

4.1. Container Elements

TOC

4.1.1. The "metalink:metalink" Element

TOC

The "metalink:metalink" element is the document (i.e., top-level) element of a Metalink Document, acting as a container for metadata and data associated with the listed files. It contains one or more metalink:file (The "metalink:file" Element) child elements which consist of metadata elements.

```
metalinkMetalink =
   element metalink:metalink {
      metalinkCommonAttributes,
      (metalinkFile+
       & metalinkGenerator?
       & metalinkOrigin?
       & metalinkPublished?
       & metalinkUpdated?
       & extensionElement*)
   }
```

The following child elements are defined by this specification (note that the presence of some of these elements is required):

* metalink:metalink (The "metalink:metalink" Element) elements MUST contain one or more metalink:file (The "metalink:file" Element) elements.

* metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:generator (The "metalink:generator" Element) element and MUST NOT contain more than one such element.

* metalink:metalink (The "metalink:metalink" Element) elements SHOULD contain exactly one metalink:origin (The "metalink:origin" Element) element and MUST NOT contain more than one such element.

* metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:published (The "metalink:published" Element) element and MUST NOT contain more than one such element.

* metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:updated (The "metalink:updated" Element) element and MUST NOT contain more than one such element.

4.1.1.1. Providing Textual Content

TOC

Experience teaches that downloads providing textual content are in general more useful than those that do not. Some applications (one example is full-text indexers) require a minimum amount of text to function reliably and predictably. Metalink publishers should be aware of these issues. It is RECOMMENDED that each metalink:file (The "metalink:file" Element) element contain a non-empty metalink:description (The "metalink:description" Element) element, a non-empty metalink:identity (The "metalink:identity" Element) element, a non-empty metalink:version (The "metalink:version" Element) element, and a non-empty metalink:publisher (The "metalink:publisher" Element) element when these elements are present. However, the absence of metalink:description (The "metalink:description" Element), metalink:identity (The "metalink:identity" Element), metalink:version (The "metalink:version" Element), and metalink:publisher (The "metalink:publisher" Element) is not an error, and Metalink Processors MUST NOT fail to function correctly as a consequence of such an absence.

TOC

4.1.2. The "metalink:file" Element

The "metalink:file (The "metalink:file" Element)" element represents an individual file, acting as a container for metadata and data associated with the file. Each unique file described in a Metalink Document MUST have its own metalink:file (The "metalink:file" Element) element. All metalink:url (The "metalink:url" Element) elements contained in each metalink:file (The "metalink:file" Element) element SHOULD lead to identical files. That is, each metalink:url (The "metalink:url" Element) element should be an alternative location for the same file and each metalink:metaurl (The "metalink:metaurl" Element) element should provide metadata to retrieve the same file in another way, such as a peer to peer network. Refer to Section 4.2.16 (The "metalink:url" Element) and Section 4.2.8 (The "metalink:metaurl" Element) for more information.

```
metalinkFile =
   element metalink:file {
      metalinkCommonAttributes,
      attribute name { text },
      (metalinkCopyright?
       & metalinkDescription?
       & metalinkHash*
       & metalinkIdentity?
       & metalinkLanguage*
       & metalinkLogo?
       & metalinkMetaURL*
       & metalinkURL*
       & metalinkOS*
       & metalinkPieces*
       & metalinkPublisher?
       & metalinkSignature?
       & metalinkSize?
       & metalinkVersion?
       & extensionElement*)
   }
```

This specification assigns no significance to the order of metalink:file (The "metalink:file" Element) elements or to the order of metalink:url (The "metalink:url" Element) or metalink:metaurl (The "metalink:metaurl" Element) elements. Significance is determined by the value of the "priority" attribute of the metalink:url (The "metalink:url" Element) or metalink:metaurl (The "metalink:metaurl" Element) elements.
The following child elements are defined by this specification (note that it requires the presence of some of these elements):

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:copyright (The "metalink:copyright" Element) element and MUST NOT contain more than one such element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:description (The "metalink:description" Element) element and MUST NOT contain more than one such element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:identity (The "metalink:identity" Element) element and MUST NOT contain more than one such element.

* metalink:file (The "metalink:file" Element) elements MAY contain one or more metalink:hash (The "metalink:hash" Element) elements.

* metalink:file (The "metalink:file" Element) elements MAY contain one or more metalink:language (The "metalink:language" Element) elements.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:logo (The "metalink:logo" Element) element and MUST NOT contain more than one such element.

* metalink:file (The "metalink:file" Element) elements MAY contain one or more metalink:os (The "metalink:os" Element) element.

* metalink:file (The "metalink:file" Element) elements MUST contain at least one metalink:url (The "metalink:url" Element) element or at least one metalink:metaurl (The "metalink:metaurl" Element) element. Typically, metalink:file (The "metalink:file" Element) elements contain more than one metalink:url (The "metalink:url" Element) element to provide multiple download sources.

* metalink:file (The "metalink:file" Element) elements MAY contain one or more metalink:pieces (The "metalink:pieces" Element) elements.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:publisher (The "metalink:publisher" Element) element and MUST NOT contain more than one such element.

* metalink:file (The "metalink:file" Element) elements MAY contain one or more metalink:signature (The "metalink:signature" Element) elements.

* metalink:file (The "metalink:file" Element) elements SHOULD contain exactly one metalink:size (The "metalink:size" Element) element and MUST NOT contain more than one such element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:version (The "metalink:version" Element) element and MUST NOT contain more than one such element.

4.1.2.1. The "name" Attribute

TOC

metalink:file (The "metalink:file" Element) elements MUST have a "name" attribute, which contains the local filename that the downloaded file will be written to. Hence, if a Metalink Document contains multiple metalink:file (The "metalink:file" Element) elements, the value of the "name" attribute MUST be unique for each.
Directory information can also be contained in a "path/file" format only, as in:

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

In this example, a subdirectory "debian-amd64/sarge/" will be created and a file named "Contents-amd64.gz" will be created inside it. The path MUST NOT contain any directory traversal directives or information. The path MUST be relative. The path MUST NOT begin with a "/", "./" or "../", contain "/../", or end with "/..".

4.1.3. The "metalink:pieces" Element

TOC

The "metalink:pieces (The "metalink:pieces" Element)" element acts as a container for a list of cryptographic hashes of non-overlapping pieces of the file. The cryptographic hashes MUST be listed in the same order as the corresponding pieces appear in the file, starting at the beginning of the file.

```
metalinkPieces =
   element metalink:pieces {
      attribute length { xsd:positiveInteger },
      attribute type { text },
      metalinkHash+
   }
```

TOC

4.1.3.1. The "type" Attribute

metalink:pieces (The "metalink:pieces" Element) elements MUST have a "type" attribute.
The Internet Assigned Numbers Authority (IANA) registry named "Hash Function Textual Names" defines values for hash types. See Section 7.4 (Cryptographic Hashes) for security implications.

4.1.3.2. The "length" Attribute

TOC

metalink:pieces (The "metalink:pieces" Element) elements MUST have a "length" attribute, which is a positive integer that describes the length of the pieces of the file in octets. The whole file is divided into non-overlapping pieces of this length, starting from the beginning of the file. That is, every piece MUST be the same size, apart from the last piece which is the remainder. The last piece extends to the end of the file, and therefore MAY be shorter than the other pieces.

4.2. Metadata Elements

TOC

4.2.1. The "metalink:copyright" Element

TOC

The "metalink:copyright (The "metalink:copyright" Element)" element is a Text construct that conveys a human-readable copyright for a file. It is Language-Sensitive.

```
     metalinkCopyright =
        element metalink:copyright {
          metalinkTextConstruct
        }
```

4.2.2. The "metalink:description" Element

TOC

The "metalink:description (The "metalink:description" Element)" element is a Text construct that conveys a human-readable file description. It is Language-Sensitive.

```
metalinkDescription =
   element metalink:description {
     metalinkTextConstruct
   }
```

4.2.3. The "metalink:generator" Element

TOC

The "metalink:generator (The "metalink:generator" Element)" element's content identifies the generating agent name and version used to generate a Metalink Document, for debugging and other purposes.

```
metalinkGenerator =
   element metalink:generator {
     metalinkTextConstruct
   }
```

The metalink:generator (The "metalink:generator" Element) element's content is defined below in ABNF notation [RFC5234] (Crocker, D., Ed. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” January 2008.).

```
token          = 1*<any CHAR except CTLs or separators>
separators     = "(" / ")" / "<" / ">" / "@"
               / "," / ";" / ":" / "\" / <">
               / "/" / "[" / "]" / "?" / "="
               / "{" / "}" / SP / HTAB
agent         = token ["/" agent-version]
agent-version = token
```

Examples:

```
...
<generator>MirrorBrain/2.11</generator>
...
<generator>MirrorManager/1.2.11</generator>
...
<generator>metalinktools/0.3.6</generator>
...
<generator>MetalinkEditor/1.2.0</generator>
...
```

Although any token character MAY appear in an agent-version, this token SHOULD only be used for a version identifier (i.e., successive versions of the same agent SHOULD only differ in the agent-version portion of the agent value).

4.2.4. The "metalink:hash" Element

TOC

The "metalink:hash (The "metalink:hash" Element)" element is a Text construct that conveys a cryptographic hash for a file. All hashes are encoded in lowercase hexadecimal format. Hashes are used to verify the integrity of a complete file or portion of a file to determine if the file has been transferred without any errors.

```
metalinkHash =
   element metalink:hash {
     attribute type { text }?,
     text
   }
```

Metalink Documents MAY contain one or multiples hashes of a complete file. metalink:hash (The "metalink:hash" Element) elements with a "type" attribute MUST contain a hash of the complete file. In this example, both SHA-1 and SHA-256 hashes of the complete file are included.

```
...
  <hash type="sha-1">a97fcf6ba9358f8a6f62beee4421863d3e52b080</hash>
  <hash type="sha-256">fc87941af7fd7f03e53b34af393f4c14923d74...</hash>
...
```

Metalink Documents MAY also contain hashes for individual pieces of a file. metalink:hash (The "metalink:hash" Element) elements that are inside a metalink:pieces (The "metalink:pieces" Element) container element have a hash for that specific piece or chunk of the file, and are of the same hash type as the metalink:pieces (The "metalink:pieces" Element) element they are contained in.
metalink:hash (The "metalink:hash" Element) elements without a "type" attribute MUST contain a hash for that specific piece or chunk of the file and MUST be listed in the same order as the corresponding pieces appear in the file, starting at the beginning of the file. The size of the piece is equal to the value of the "length" attribute of the metalink:pieces (The "metalink:pieces" Element) element, apart from the last piece which is the remainder. See Section 4.1.3.2 (The "length" Attribute) for more information on the size of pieces.
In this example, SHA-1 and SHA-256 hashes of the complete file are included, along with four SHA-1 piece hashes.

```
...
  <hash type="sha-1">a97fcf6ba9358f8a6f62beee4421863d3e52b080</hash>
  <hash type="sha-256">fc87941af7fd7f03e53b34af393f4c14923d74...</hash>
  <pieces length="1048576" type="sha-1">
    <hash>d96b9a4b92a899c2099b7b31bddb5ca423bb9b30</hash>
    <hash>10d68f4b1119014c123da2a0a6baf5c8a6d5ba1e</hash>
    <hash>3e84219096435c34e092b17b70a011771c52d87a</hash>
    <hash>67183e4c3ab892d3ebe8326b7d79eb62d077f487</hash>
  </pieces>
...
```

4.2.4.1. The "type" Attribute

TOC

metalink:hash (The "metalink:hash" Element) elements MUST have a "type" attribute, if and only if it contains a hash of the complete file. The IANA registry named "Hash Function Textual Names" defines values for hash types. See Section 7.4 (Cryptographic Hashes) for security implications.

4.2.5. The "metalink:identity" Element

TOC

The "metalink:identity (The "metalink:identity" Element)" element is a Text construct that conveys a human-readable identity for a file. For example, the identity of Firefox 3.5 would be "Firefox".

```
metalinkIdentity =
   element metalink:identity {
     metalinkTextConstruct
   }
```

4.2.6. The "metalink:language" Element

TOC

The "metalink:language (The "metalink:language" Element)" element is a Text construct that conveys a code for the language of a file, per [RFC5646] (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.).
Multiple metalink:language (The "metalink:language" Element) elements are allowed, for instance, to describe a file such as an binary installation program that provides multiple language options, or a

movie with multiple language tracks, or a document in multiple languages.

```
metalinkLanguage =
   element metalink:language {
     metalinkTextConstruct
   }
```

4.2.7. The "metalink:logo" Element

TOC

The "metalink:logo (The "metalink:logo" Element)" element's content is an IRI reference [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” January 2005.) that identifies an image that provides visual identification for a file.

```
metalinkLogo =
   element metalink:logo {
     metalinkCommonAttributes,
     (metalinkUri)
   }
```

The image SHOULD have an aspect ratio of one (horizontal) to one (vertical) and SHOULD be suitable for presentation at a small size.

4.2.8. The "metalink:metaurl" Element

TOC

The "metalink:metaurl (The "metalink:metaurl" Element)" element contains the IRI of a metadata file, also known as a metainfo file, about a resource to download. For example, this could be the IRI of a BitTorrent .torrent file, a Metalink Document, or other type of metadata file. Note that the information in the metalink:hash (The "metalink:hash" Element) element does not apply to these metadata files, but to the files that are described by them.

```
metalinkMetaURL =
   element metalink:metaurl {
      metalinkCommonAttributes,
      attribute priority { xsd:positiveInteger {
         maxInclusive = "999999"}}?,
      attribute mediatype { text },
      attribute name { text }?,
      (metalinkUri)
   }
```

4.2.8.1. The "priority" Attribute

TOC

metalink:metaurl (The "metalink:metaurl" Element) elements MAY have a priority attribute. Values MUST be positive integers between 1 and 999999. Lower values indicate a higher priority. metalink:metaurl (The "metalink:metaurl" Element) elements without a priority attribute are considered to have the lowest priority, i.e. 999999. The priority values of metalink:metaurl (The "metalink:metaurl" Element) and metalink:url (The "metalink:url" Element) elements are compared and those with the lowest values, starting with 1, are used first. Multiple metalink:metaurl (The "metalink:metaurl" Element) and metalink:url (The "metalink:url" Element) elements MAY have the same priority, i.e. one BitTorrent .torrent file and three FTP URIs could have priority="1". See also the "priority" attribute of the metalink:url (The "metalink:url" Element) element.

4.2.8.2. The "mediatype" Attribute

TOC

metalink:metaurl (The "metalink:metaurl" Element) elements MUST have a "mediatype" attribute that indicates the Multipurpose Internet Mail Extensions (MIME) media type [RFC4288] (Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures," December 2005.) of the metadata available at the IRI. In the case of BitTorrent as specified in [BITTORRENT] (Cohen, B., "The BitTorrent Protocol Specification," February 2008.), the value "torrent" is REQUIRED. Types without "/" are reserved. Currently, "torrent" is the only reserved value.
Values for this attribute are defined below in ABNF notation [RFC5234] (Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," January 2008.).

```
media-type = (type-name "/" subtype-name) / media-reserved
media-reserved = "torrent"
type-name = <Defined in section 4.2 of RFC 4288>
subtype-name = <Defined in section 4.2 of RFC 4288>
```

4.2.8.3. The "name" Attribute

TOC

metalink:metaurl (The "metalink:metaurl" Element) elements MAY have a "name" attribute that indicates a specific file in a BitTorrent .torrent file or a Metalink Document that describes multiple files. Directory information can also be contained in a "path/file" format only, as in:

```
<metaurl
     mediatype="torrent" name="debian-amd64/sarge/Contents-amd64.gz">
```

In this example, a file named "Contents-amd64.gz" is indicated, in a "debian-amd64/sarge/" subdirectory. The path MUST NOT contain any directory traversal directives or information. The path MUST be relative. The path MUST NOT begin with a "/", "./" or "../", contain "/../", or end with "/..".

4.2.9. The "metalink:origin" Element

TOC

The "metalink:origin (The "metalink:origin" Element)" element is an IRI where the Metalink Document was originally published. If the dynamic attribute of metalink:origin (The "metalink:origin" Element) is "true", then updated versions of the Metalink can be found at this IRI.

```
metalinkOrigin =
   element metalink:origin {
     metalinkCommonAttributes,
     attribute dynamic { xsd:boolean }?,
     (metalinkUri)
   }
```

TOC

4.2.9.1. The "dynamic" Attribute

The metalink:origin (The "metalink:origin" Element) element MAY have a "dynamic" attribute, set to "true" or "false", which tells if a Metalink at the origin IRI will contain dynamic updated information or if it is static and not likely to be updated.

4.2.10. The "metalink:os" Element

TOC

The "metalink:os (The "metalink:os" Element)" element is a Text construct that conveys an Operating System for a file. The IANA registry named "Operating System Names" defines values for OS types.

```
metalinkOS =
   element metalink:os {
     metalinkTextConstruct
   }
```

4.2.11. The "metalink:published" Element

TOC

The "metalink:published (The "metalink:published" Element)" element is a Date construct indicating an instant in time associated with an event early in the life cycle of the entry.

```
metalinkPublished =
   element metalink:published {
     metalinkDateConstruct
   }
```

Typically, metalink:published (The "metalink:published" Element) will be associated with the initial creation or first availability of the resource. The metalink:updated (The "metalink:updated" Element) element is used when a Metalink Document has been updated after initial publication.

4.2.12. The "metalink:publisher" Element

TOC

The "metalink:publisher (The "metalink:publisher" Element)" element contains a human-readable group or other entity which has published the file described in the Metalink Document and an IRI for more information.

```
metalinkPublisher =
   element metalink:publisher {
     metalinkCommonAttributes,
     attribute name { text },
     attribute url { metalinkUri }?
   }
```

The metalink:publisher (The "metalink:publisher" Element) element MUST have a "name" attribute that indicates the human-readable name of the publisher.
The metalink:publisher (The "metalink:publisher" Element) element MAY have a "url" attribute whose value MUST be an IRI reference [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” January 2005.).

4.2.13. The "metalink:signature" Element

TOC

The "metalink:signature (The "metalink:signature" Element)" element is a Text construct that conveys a digital signature for a file described in a Metalink Document. Digital signatures verify that a file is from the entity that has signed it.
Support in Metalink Processors for digital signatures included in this element is OPTIONAL. Note that the signing of Metalink Documents, as opposed to a digital signature of a file described in a Metalink Document, is covered in Section 7.1 (Digital Signatures).

```
metalinkSignature =
   element metalink:signature {
     attribute mediatype { text },
     metalinkTextConstruct
   }
```

Example with an OpenPGP signature [RFC4880] (Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, “OpenPGP Message Format,” November 2007.):

```
<signature mediatype="application/pgp-signature">
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.10 (GNU/Linux)

iEYEABECAAYFAkrxdXQACgkQeOEcayedXJHqFwCfd1p/HhRf/iDvYhvFbTrQPz+p
p3oAoO9lKHoOqOE0EMB3zmMcLoYUrNkg
=ggAf
-----END PGP SIGNATURE-----
</signature>
```

4.2.13.1. The "mediatype" Attribute

TOC

metalink:signature (The "metalink:signature" Element) elements MUST have a "mediatype" attribute that indicates the MIME media type [RFC4288] (Freed, N. and J. Klensin, “Media Type Specifications and Registration Procedures,” December 2005.) of the included digital signature.
Values for this attribute are defined below in ABNF notation [RFC5234] (Crocker, D., Ed. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” January 2008.).

```
media-type = type-name "/" subtype-name
type-name = <Defined in section 4.2 of RFC 4288>
subtype-name = <Defined in section 4.2 of RFC 4288>
```

4.2.14. The "metalink:size" Element

TOC

The "metalink:size (The "metalink:size" Element)" element indicates the length of the linked content in octets. This is the content length of the representation returned when the IRI is mapped to a URI and dereferenced. Note that the "metalink:size (The "metalink:size" Element)" element MUST override the actual content length of the representation as reported by the underlying protocol, and those that do not match will be discarded by Metalink Processors. This value MUST be a non-negative integer.

```
metalinkSize =
   element metalink:size {
     xsd:nonNegativeInteger
   }
```

4.2.15. The "metalink:updated" Element

TOC

The "metalink:updated (The "metalink:updated" Element)" element is a Date construct indicating the most recent instant in time when a Metalink was modified in a way the publisher considers significant. Therefore, not all modifications necessarily result in a changed metalink:updated (The "metalink:updated" Element) value.

```
metalinkUpdated =
   element metalink:updated {
     metalinkDateConstruct
   }
```

Publishers MAY change the value of this element over time.

4.2.16. The "metalink:url" Element

TOC

The "metalink:url (The "metalink:url" Element)" element contains a file IRI. Most metalink:file (The "metalink:file" Element) container elements will contain multiple metalink:url (The "metalink:url" Element) elements, and each one SHOULD be a valid alternative to download the same file.
The metalink:url (The "metalink:url" Element) elements SHOULD be resolvable and, if resolvable, SHOULD lead to identical files.
Metalink Processors MUST filter out invalid files obtained from "metalink:url (The "metalink:url" Element)" elements by using information in the metalink:size (The "metalink:size" Element) element and metalink:hash (The "metalink:hash" Element) elements.

```
metalinkURL =
   element metalink:url {
      metalinkCommonAttributes,
      attribute location { xsd:string {
         minLength = "2"  maxLength="2"}
      }?,
      attribute priority { xsd:positiveInteger {
         maxInclusive = "999999"}}?,
      (metalinkUri)
   }
```

4.2.16.1. The "priority" Attribute

TOC

metalink:url (The "metalink:url" Element) elements MAY have a priority attribute. Values MUST be positive integers between 1 and 999999. Lower values indicate a higher priority. metalink:url (The "metalink:url" Element) elements without a priority attribute are considered to have the lowest priority, i.e. 999999. Multiple metalink:url (The "metalink:url" Element) elements can have the same priority, i.e. ten different mirrors could have priority="1".

4.2.16.2. The "location" Attribute

TOC

metalink:url (The "metalink:url" Element) elements MAY have a "location" attribute, which is a [ISO3166-1] (International Organization for Standardization, “ISO 3166-1:2006. Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes,” November 2006.) alpha-2 two letter country code for the geographical location of the physical server an IRI is used to access.

4.2.17. The "metalink:version" Element

TOC

The "metalink:version (The "metalink:version" Element)" element is a Text construct that conveys a human-readable version for a file. The version of Firefox 3.5 would be "3.5".

```
     metalinkVersion =
        element metalink:version {
          metalinkTextConstruct
        }
```

5. Extending Metalink

TOC

5.1. Extensions from Non-Metalink Vocabularies

TOC

This specification describes Metalink's XML markup vocabulary. Markup from other vocabularies ("foreign markup") can be used in a Metalink Document.

5.2. Extensions to the Metalink Vocabulary

TOC

The Metalink namespace is reserved for future forward-compatible revisions of Metalink. Future versions of this specification could add new elements and attributes to the Metalink markup vocabulary. Software written to conform to this version of the specification will not be able to process such markup correctly and, in fact, will not be able to

distinguish it from markup error. For the purposes of this discussion, unrecognized markup from the Metalink vocabulary will be considered "foreign markup".

5.3. Processing Foreign Markup

TOC

Metalink Processors that encounter foreign markup in a location that is legal according to this specification MUST ignore such foreign markup, in particular they MUST NOT stop processing or signal an error. It might be the case that the Metalink Processor is able to process the foreign markup correctly and does so. Otherwise, such markup is termed "unknown foreign markup".
When unknown foreign markup is encountered as a child of metalink:file (The "metalink:file" Element), metalink:metalink (The "metalink:metalink" Element), Metalink Processors MAY bypass the markup and any textual content and MUST NOT change their behavior as a result of the markup's presence.

5.4. Extension Elements

TOC

Metalink allows foreign markup anywhere in a Metalink document, except where it is explicitly forbidden. Child elements of metalink:file (The "metalink:file" Element) and metalink:metalink (The "metalink:metalink" Element) are considered Metadata elements and are described below. The role of other foreign markup is undefined by this specification.

5.4.1. Simple Extension Elements

TOC

A Simple Extension element MUST NOT have any attributes or child elements. The element MAY contain character data or be empty. Simple Extension elements are not Language-Sensitive.

```
simpleExtensionElement =
   element * - metalink:* {
      text
   }
```

The element can be interpreted as a simple property (or name/value pair) of the parent element that encloses it. The pair consisting of the namespace-URI of the element and the local name of the element can be interpreted as the name of the property. The character data content

of the element can be interpreted as the value of the property. If the element is empty, then the property value can be interpreted as an empty string.

5.4.2. Structured Extension Elements

TOC

The root element of a Structured Extension element MUST have at least one attribute or child element. It MAY have attributes, it MAY contain well-formed XML content (including character data), or it MAY be empty. Structured Extension elements are Language-Sensitive.

```
structuredExtensionElement =
   element * - metalink:* {
      (attribute * { text }+,
         (text|anyElement)*)
    | (attribute * { text }*,
       (text?, anyElement+, (text|anyElement)*))
   }
```

The structure of a Structured Extension element, including the order of its child elements, could be significant.
This specification does not provide an interpretation of a Structured Extension element. The syntax of the XML contained in the element (and an interpretation of how the element relates to its containing element) is defined by the specification of the Metalink extension.

---

6. IANA Considerations

TOC

6.1. XML Namespace Registration

TOC

This document makes use of the XML registry specified in [RFC3688] (Mealling, M., “The IETF XML Registry,” January 2004.). Accordingly, IANA has made the following registration:
Registration request for the Metalink namespace:
URI: urn:ietf:params:xml:ns:metalink
Registrant Contact: See the "Author's Address" section of this document.
XML: None. Namespace URIs do not represent an XML specification.

6.2. application/metalink4+xml MIME type

TOC

A Metalink Document, when serialized as XML 1.0, can be identified with the following media type:

> **MIME media type name:** application
>
> **MIME subtype name:** metalink4+xml
>
> **Mandatory parameters:** None.
>
> **Optional parameters:**
>
> > **"charset":** This parameter has semantics identical to the charset parameter of the "application/xml" media type as specified in [RFC3023] (Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.).
>
> **Encoding considerations:** Identical to those of "application/xml" as described in [RFC3023] (Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.), Section 3.2.
>
> **Security considerations:** As defined in this specification.
>
> In addition, as this media type uses the "+xml" convention, it shares the same security considerations as described in [RFC3023] (Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.), Section 10.
>
> **Interoperability considerations:** There are no known interoperability issues.
>
> **Published specification:** This specification.
>
> **Applications that use this media type:** No known applications currently use this media type.

Additional information:

> **Magic number(s):** As specified for "application/xml" in [RFC3023] (Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.), Section 3.2.
>
> **File extension:** .meta4
>
> **Fragment identifiers:** As specified for "application/xml" in [RFC3023] (Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.), Section 5.

Base URI:

As specified in [RFC3023] (Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” January 2001.), Section 6.

Macintosh File Type code: TEXT

Person and email address to contact for further information:
Anthony Bryan <anthonybryan@gmail.com>

Intended usage: COMMON

Author/Change controller: IESG

---

7. Security Considerations

TOC

Because Metalink is an XML-based format, existing XML security mechanisms can be used to secure its content.
Publishers of Metalink Documents may have sound reasons for signing otherwise-unprotected content. For example, a merchant might digitally sign a Metalink that lists a file download to verify its origin. Other merchants may wish to sign and encrypt Metalink Documents that list digital songs that have been purchased. Of course, many other examples are conceivable as well.
Publishers are encouraged to offer Metalink documents via authenticated HTTP under TLS (Transport Layer Security) as specified in [RFC2818] (Rescorla, E., “HTTP Over TLS,” May 2000.). The choice of a secure content layer is entirely possible for content providers.
Publishers are also encouraged to include digital signatures of the files within the Metalink Documents, if they are available, as described in Section 4.2.13 (The "metalink:signature" Element).
Normally, a publisher is in the best position to know how strong the protective signing ought to be on their content. Thus, a publisher can choose weak or strong cryptography, and a Metalink Processor would normally accept that. There MAY be applications where the Metalink Processor chooses to reject weak cryptography, but that is not envisioned as the common use case.

7.1. Digital Signatures

TOC

The root of a Metalink Document (i.e., metalink:metalink (The "metalink:metalink" Element)) or any metalink:file (The "metalink:file" Element) element MAY have an Enveloped Signature, as described by XML-Signature and Syntax Processing (Solo, D., Reagle, J., and D. Eastlake,

“XML-Signature Syntax and Processing (Second Edition),” June 2008.)
[REC-xmldsig-core].
Although signing and verifying signatures are both OPTIONAL, an implementation that supports either feature SHOULD implement RSA with a minimum key size of 2048 with SHA-256.
Metalink Processors that support verifying signatures MUST reject Metalink Documents with invalid signatures.
Metalink Processors MUST NOT reject a Metalink Document containing such a signature because they are not capable of verifying it; they MUST continue processing and MAY inform the user of their failure to validate the signature.
In other words, the presence of an element with the namespace URI "http://www.w3.org/2000/09/xmldsig#" and a local name of "Signature" as a child of the document element MUST NOT cause a Metalink Processor to fail merely because of its presence.
Other elements in a Metalink Document MUST NOT be signed unless their definitions explicitly specify such a capability.
Section 6.5.1 of [REC-xmldsig-core] (Solo, D., Reagle, J., and D. Eastlake, “XML-Signature Syntax and Processing (Second Edition),” June 2008.) requires support for Canonical XML [REC-xml-c14n] (Boyer, J., “Canonical XML Version 1.0,” March 2001.). However, many implementers do not use it because signed XML documents enclosed in other XML documents have their signatures broken. Thus, Metalink Processors that verify signed Metalink Documents MUST be able to canonicalize with the exclusive XML canonicalization method identified by the URI "http://www.w3.org/2001/10/xml-exc-c14n#", as specified in Exclusive XML Canonicalization [REC-xml-exc-c14n] (Eastlake, D., Boyer, J., and J. Reagle, “Exclusive XML Canonicalization Version 1.0,” July 2002.).
Section 4.4.2 of [REC-xmldsig-core] (Solo, D., Reagle, J., and D. Eastlake, “XML-Signature Syntax and Processing (Second Edition),” June 2008.) requires support for DSA signatures and recommends support for RSA signatures. However, because of the much greater popularity in the market of RSA versus DSA, Metalink Processors that verify signed Metalink Documents MUST be able to verify RSA signatures, but do not need be able to verify DSA signatures. Due to security issues that can arise if the keying material for message authentication code (MAC) authentication is not handled properly, Metalink Documents SHOULD NOT use MACs for signatures.

7.2. URIs and IRIs

TOC

Metalink Processors handle URIs and IRIs. See Section 7 of [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.) and Section 8 of [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource

Identifiers (IRIs)," January 2005.) for security considerations related to their handling and use.

7.3. Spoofing

TOC

There is potential for spoofing attacks where the attacker publishes Metalink Documents with false information. Malicious publishers might create Metalink Documents containing inaccurate information anywhere in the document. Unaware downloaders could be deceived into downloading a malicious or worthless file. Malicious publishers could attempt a distributed denial of service attack by inserting unrelated IRIs into Metalink Documents.
Digital signatures address the issue of spoofing.

7.4. Cryptographic Hashes

TOC

Currently, some of the hash types defined in the IANA registry named "Hash Function Textual Names" are considered insecure. These include the whole Message Digest family of algorithms which are not suitable for cryptographically strong verification. Malicious people could provide files that appear to be identical to another file because of a collision, i.e. the weak cryptographic hashes of the intended file and a substituted malicious file could match.
Metalink Generators and Processors MUST support "sha-256" which is SHA-256, as specified in [FIPS-180-3] (National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," October 2008.), and MAY support stronger hashes.
If a Metalink Document contains hashes, it SHOULD include "sha-256" which is SHA-256, or stronger. It MAY also include other hashes from the IANA registry named "Hash Function Textual Names".

8. References

TOC

8.1. Normative References

TOC

[BITTORRENT]	Cohen, B., "The BitTorrent Protocol Specification," BITTORRENT 11031, February 2008.
[FIPS-180-3]	

	National Institute of Standards and Technology (NIST), “Secure Hash Standard (SHS),” FIPS PUB 180-3, October 2008.
[ISO3166-1]	International Organization for Standardization, “ISO 3166-1:2006. Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes,” November 2006.
[REC-xml]	Yergeau, F., Paoli, J., Bray, T., Sperberg-McQueen, C., and E. Maler, “Extensible Markup Language (XML) 1.0 (Fifth Edition),” W3C REC-xml-20081126, November 2008.
[REC-xml-c14n]	Boyer, J., “Canonical XML Version 1.0,” W3C REC REC-xml-c14n-20010315, March 2001.
[REC-xml-exc-c14n]	Eastlake, D., Boyer, J., and J. Reagle, “Exclusive XML Canonicalization Version 1.0,” W3C REC REC-xml-exc-c14n-20020718, July 2002.
[REC-xml-infoset]	Cowan, J. and R. Tobin, “XML Information Set (Second Edition),” W3C REC-xml-infoset-20040204, February 2004.
[REC-xml-names]	Hollander, D., Bray, T., Tobin, R., and A. Layman, “Namespaces in XML 1.0 (Third Edition),” W3C REC-xml-names-20091208, December 2009.
[REC-xmldsig-core]	Solo, D., Reagle, J., and D. Eastlake, “XML-Signature Syntax and Processing (Second Edition),” W3C REC-xmldsig-core-20080610, June 2008.
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997.
[RFC2818]	Rescorla, E., “HTTP Over TLS,” RFC 2818, May 2000.
[RFC3023]	Murata, M., St. Laurent, S., and D. Kohn, “XML Media Types,” RFC 3023, January 2001.
[RFC3339]	Klyne, G. and C. Newman, “Date and Time on the Internet: Timestamps,” RFC 3339, July 2002.
[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” STD 66, RFC 3986, January 2005.
[RFC3987]	Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” RFC 3987, January 2005.
[RFC4288]	Freed, N. and J. Klensin, “Media Type Specifications and Registration Procedures,” BCP 13, RFC 4288, December 2005.
[RFC5234]	Crocker, D., Ed. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” STD 68, January 2008.
[RFC5646]	Phillips, A. and M. Davis, “Tags for Identifying Languages,” BCP 47, RFC 5646, September 2009.

8.2. Informative References

TOC

[ISO.8601.1988]	International Organization for Standardization, “Data elements and interchange formats - Information interchange - Representation of dates and times,” ISO Standard 8601, June 1988.
[NOTE-datetime-19980827]	Wolf, M. and C. Wicksteed, “Date and Time Formats,” W3C NOTE-datetime-19980827, August 1998.
[REC-xmlschema-2-20041028]	Malhotra, A. and P. Biron, “XML Schema Part 2: Datatypes Second Edition,” W3C REC-xmlschema-2-20041028, October 2004.
[RELAX-NG]	Clark, J., “RELAX NG Compact Syntax,” December 2001.
[RFC3688]	Mealling, M., “The IETF XML Registry,” BCP 81, RFC 3688, January 2004.
[RFC4287]	Nottingham, M. and R. Sayre, “The Atom Syndication Format,” RFC 4287, December 2005.
[RFC4880]	Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, “OpenPGP Message Format,” RFC 4880, November 2007.

Appendix A. Acknowledgements and Contributors

TOC

The layout and shape of this document relies heavily on work pioneered in the Atom Syndication Format as specified in [RFC4287] (Nottingham, M. and R. Sayre, “The Atom Syndication Format,” December 2005.).
The content and concepts within are a product of the Metalink community. Key contributors provided early implementations: A. Bram Neijt, Hampus Wessman, Darius Liktorius, Manuel Subredu, Michael Burford, Giorgio Maone, Nils Maier, Max Velasques, Manolo Valdes, Hayden Legendre, Frederick Cheung, Rene Leonhardt, Per Oyvind Karlsen, Matt Domsch, Yazsoft, KGet developers, Free Download Manager developers, Orbit developers, Arne Babenhauserheide, Mathias Berchtold, Xienzhenyu and TheWorld Browser developers, Xi Software, Agostino Russo, and James Antill.
The Metalink community has dozens of contributors who contributed to the evolution of Metalink or proposed ideas and wording for this document, including:
Paul Burkhead, Kristian Weston, Nicolas Alvarez, Urs Wolfer, Bridget and Ethan Fletcher, Patrick Ruckstuhl, Sebastien Willemijns, Micah Cowan, Ruben Kerkhof, Danny Ayers, Nick Dominguez, Gary Zellerbach, James Clark, Daniel Stenberg, John and Sandra Sowder, Salvatore Musumeci, Steve Eshelman, Lucas Hewett, Ryan Cronin, Dave Winquist, Bob

Denison, Wes Shelton, Kees Cook, Josh Colbert, Steve Kleisath, Chad Neptune, Nick Carrabba, Chris Carrabba, Erin Solari, Derick Cordoba, Ryan Alexander, Tom Mainville, Janie Wargo, Jason Hansen, Tim Bray, Dan Brickley, Markus Hofmann, Dan Connolly, Tim Berners-Lee, Louis Suarez-Potts, Ross Smith, Jeff Covey, Ed Lee, Shawn Wilsher, Mike Connor, Johan Svedberg, Dedric Carter, and Debi Goulding. We also thank the Anthony Family, the Bryan Family, Juanita Anthony and Zimmy Bryan.
We also thank the following contributors for assistance and review: Eran Hammer-Lahav, Lisa Dusseault, Mark Nottingham, Peter Saint-Andre, Julian Reschke, Chris Newman, Ian Macfarlane, Dave Cridland, Barry Leiba, Uri Blumenthal, Paul Hoffman, Felix Sasaki, Matthias Fuchs, Mark Baker, Scott Cantor, Brian Carpenter, Alexey Melnikov, Lars Eggert, Pasi Eronen, Tim Polk, and Dan Romascanu.

Appendix B. RELAX NG Compact Schema

TOC

This appendix is informative.
The Relax NG schema explicitly excludes elements in the Metalink namespace that are not defined in this revision of the specification. Requirements for Metalink Processors encountering such markup are given in Sections 5.2 (Extensions to the Metalink Vocabulary) and 5.3 (Processing Foreign Markup).

```
# -*- rnc -*-
# RELAX NG Compact Syntax Grammar for the
# Metalink Format Specification Version 4
# Based on RFC 4287 schema

namespace local = ""
namespace metalink = "urn:ietf:params:xml:ns:metalink"
namespace xsd = "http://www.w3.org/2001/XMLSchema"

# Common attributes

metalinkCommonAttributes =
   attribute xml:lang { metalinkLanguageTag }?,
   undefinedAttribute*

# Text Constructs

metalinkTextConstruct =
   metalinkCommonAttributes,
   text

# Date Construct

metalinkDateConstruct =
   metalinkCommonAttributes,
   xsd:dateTime

start = metalinkMetalink

metalinkMetalink =
   element metalink:metalink {
      metalinkCommonAttributes,
      (metalinkFile+
       & metalinkGenerator?
       & metalinkOrigin?
       & metalinkPublished?
       & metalinkUpdated?
       & extensionElement*)
   }

metalinkFile =
   element metalink:file {
      metalinkCommonAttributes,
      attribute name { text },
      (metalinkCopyright?
       & metalinkDescription?
       & metalinkHash*
       & metalinkIdentity?
       & metalinkLanguage*
```

```
        & metalinkLogo?
        & metalinkMetaURL*
        & metalinkURL*
        & metalinkOS*
        & metalinkPieces*
        & metalinkPublisher?
        & metalinkSignature?
        & metalinkSize?
        & metalinkVersion?
        & extensionElement*)
   }

metalinkPieces =
   element metalink:pieces {
      attribute length { xsd:positiveInteger },
      attribute type { text },
      metalinkHash+
   }

metalinkCopyright =
   element metalink:copyright {
      metalinkTextConstruct
   }

metalinkDescription =
   element metalink:description {
      metalinkTextConstruct
   }

metalinkGenerator =
   element metalink:generator {
      metalinkTextConstruct
   }

metalinkHash =
     element metalink:hash {
       attribute type { text }?,
       text
     }

metalinkIdentity =
   element metalink:identity {
      metalinkTextConstruct
   }

metalinkLanguage =
   element metalink:language {
      metalinkTextConstruct
   }
```

```
metalinkLogo =
   element metalink:logo {
      metalinkCommonAttributes,
      (metalinkUri)
   }

metalinkMetaURL =
   element metalink:metaurl {
      metalinkCommonAttributes,
      attribute priority { xsd:positiveInteger {
         maxInclusive = "999999"}}?,
      attribute mediatype { text },
      attribute name { text }?,
      (metalinkUri)
   }

metalinkOrigin =
   element metalink:origin {
      metalinkCommonAttributes,
      attribute dynamic { xsd:boolean }?,
      (metalinkUri)
   }

metalinkOS =
   element metalink:os {
      metalinkTextConstruct
   }

metalinkPublished =
   element metalink:published {
      metalinkDateConstruct
   }

metalinkPublisher =
   element metalink:publisher {
      metalinkCommonAttributes,
      attribute name { text },
      attribute url { metalinkUri }?
   }

metalinkSignature =
   element metalink:signature {
      attribute mediatype { text },
      metalinkTextConstruct
   }

metalinkSize =
   element metalink:size {
      xsd:nonNegativeInteger
```

```
   }

metalinkUpdated =
   element metalink:updated {
      metalinkDateConstruct
   }

metalinkURL =
   element metalink:url {
      metalinkCommonAttributes,
      attribute location { xsd:string {
         minLength = "2"  maxLength="2"}
      }?,
      attribute priority { xsd:positiveInteger {
         maxInclusive = "999999"}}?,
      (metalinkUri)
   }

metalinkVersion =
   element metalink:version {
      metalinkTextConstruct
   }

# As defined in RFC 3066 and compatible with RFC 5646
metalinkLanguageTag = xsd:string {
   pattern = "[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*"
}

# Unconstrained; it's not entirely clear how IRI fit into
# xsd:anyURI so let's not try to constrain it here
metalinkUri = text

# Simple Extension

simpleExtensionElement =
   element * - metalink:* {
      text
   }

# Structured Extension

structuredExtensionElement =
   element * - metalink:* {
      (attribute * { text }+,
         (text|anyElement)*)
    | (attribute * { text }*,
       (text?, anyElement+, (text|anyElement)*))
   }

# Other Extensibility
```

```
extensionElement =
   simpleExtensionElement | structuredExtensionElement

undefinedAttribute =
  attribute * - (xml:lang | local:*) { text }

undefinedContent = (text|anyForeignElement)*

anyElement =
   element * {
      (attribute * { text }
       | text
       | anyElement)*
   }

anyForeignElement =
   element * - metalink:* {
      (attribute * { text }
       | text
       | anyElement)*
}

# EOF
```

Appendix C. Document History (to be removed by RFC Editor before publication)

TOC

[[to be removed by the RFC editor before publication as an RFC.]]
Updated versions can be found at http://tools.ietf.org/html/draft-bryan-metalink with frequent updates in Subversion at http://metalinks.svn.sourceforge.net/viewvc/metalinks/internetdraft/
Known issues concerning this draft:

*Waiting on: MIME type review.

-28 : February xx, 2010.

*Address IESG Comments and Discuss: Tim Polk.

-27 : January 28, 2010.

*Address IESG Comments and Discuss: Pasi Eronen and Dan Romascanu.

*Remove xml:base.

-26 : January 23, 2010.

*Address IESG Comments and Discuss: Alexey Melnikov, Lars Eggert.

-25 : January 11, 2010.

*Julian Reschke XML issues.

*Generator ABNF and reference. Remove license element.

*Update IPR to "trust200902".

*dynamic element changed to dynamic attribute of origin element.

-24 : December 08, 2009.

*Eran Hammer-Lahav, Document Shepherd review changes.

*Example XML indentation.

*Baseline file hash: SHA-256.

-23 : November 26, 2009.

*Lisa Dusseault, Apps Area AD review changes, Change RFC3688 from Normative to Informative Reference.

*Schema: integer changed to positiveInteger or nonNegativeInteger where fitting.

-22 : November 09, 2009.

*Clarifications.

-21 : October 13, 2009.

*Update author details.

-20 : October 12, 2009.

*RFC 5646 updates RFC 4646.

-19 : October 5, 2009.

*Remove organization for independent authors.

-18 : October 4, 2009.

*File extension: .meta4

*Hashes clarification, modified to allow multiple metalink:os elements, add size element to example.

-17 : September 28, 2009.

*Typo correction.

-16 : August 31, 2009.

*Clarifications.

-15 : August 26, 2009.

*Rename "preference" attribute of metaurl and url elements to "priority", where lower values indicate higher priority.

-14 : August 24, 2009.

*Update abstract and introduction.

-13 : August 21, 2009.

*Remove files, resources, verification container elements.

*MIME type: application/metalink4+xml

-12 : August 18, 2009.

*Remove "piece" attribute from hash elements in pieces container elements.

*Rename "uri" attribute of license and publisher elements to "url".

-11 : August 08, 2009.

*Renamed type element (static or dynamic values) to dynamic element (true or false values).

*Removed metadata inheritance and most other elements from files element.

-10 : July 28, 2009.

*Schema fixes.

*Rename metadata element to metaurl, add name attribute to it similar to file element's name attribute.

*Update REC-xmldsig-core reference to second edition.

-09 : July 11, 2009.

*Replace ISO639-2 references with RFC 4646.

*Add ISO3166-1.

-08 : July 04, 2009.

*Clarifications.

*Remove "uri" and "version" attributes from generator element.

-07 : June 18, 2009.

*This ID describes the Metalink document format/schema.

*Remove "Client Implementation Considerations" section.

*Expand "Known issues" section of Document History.

-06 : March 3, 2009.

*Add authors and this Document History section.

-05 : January 13, 2009.

*Clarifications.

-04 : December 31, 2008.

*New IPR notice as required by IETF.

*Correct "metalink:pieces" Element text.

*Add hash examples.

*Slim down "Securing Metalink Documents" section.

*Recommend at least SHA-1.

-03 : September 19, 2008.

*New namespace - urn:ietf:params:xml:ns:metalink

*Use the IANA registry named "Operating System Names" to define values for OS types.

*Add "Client Implementation Considerations" section, which includes Content Negotiation.

-02 : September 4, 2008.

*Use the IANA registry named "Hash Function Textual Names" for hash types.

*metadata Element for listing .torrent, .metalink, etc.

*Remove type attribute for url Element.

-01 : August 28, 2008.

*Clarify directory info in name attribute, hash types, add text for preference attribute.

-00 : August 23, 2008.

*Initial draft; Text largely based on RFC 4287, ideas from Metalink 3.0 specification.

Index

TOC

	version

Authors' Addresses

TOC

	Anthony Bryan
	Pompano Beach, FL
	USA
Email:	anthonybryan@gmail.com
URI:	http://www.metalinker.org
	Tatsuhiro Tsujikawa
Email:	tatsuhiro.t@gmail.com
URI:	http://aria2.sourceforge.net
	Neil McNab
Email:	neil@nabber.org
URI:	http://www.nabber.org
	Peter Poeml
	Novell, Inc.
Email:	poeml@mirrorbrain.org
URI:	http://www.mirrorbrain.org/