Network Working Group	A. Bryan, Ed.
Internet-Draft	Metalinker Project
Intended status: Standards Track	January 13, 2009
Expires: July 17, 2009	

TOC

The Metalink Download Description Format
draft-bryan-metalink-05

Status of this Memo

Copyright Notice

Abstract

This document specifies Metalink Documents, an XML-based download description format.

Table of Contents

---

1. Introduction

TOC

Metalink is an XML-based document format that describes a file or lists of files to be added to a download queue. Lists are composed of a number of files, each with an extensible set of attached metadata. For example, each file can have a description, checksum, and list of URIs that it is available from.
The primary use case that Metalink addresses is the description of downloadable content in a format so download agents can act intelligently and recover from common errors with little or no user interaction necessary. These errors can include multiple servers going down and data corrupted in transmission.
Discussion of this draft should take place on discuss@apps.ietf.org or the Metalink discussion mailing list located at metalink-discussion@googlegroups.com. To join the list, visit http://groups.google.com/group/metalink-discussion .

1.1. Examples

TOC

A brief, single file Metalink Document:

```
<?xml version="1.0" encoding="UTF-8"?>
<metalink xmlns="urn:ietf:params:xml:ns:metalink">
  <files>
    <file name="example.ext">
      <resources>
        <url>ftp://ftp.example.com/example.ext</url>
        <url>http://example.com/example.ext</url>
        <metadata type="torrent">
        http://example.com/example.ext.torrent
        </metadata>
      </resources>
    </file>
  </files>
</metalink>
```

A more extensive, single file Metalink Document:

```
<?xml version="1.0" encoding="UTF-8"?>
<metalink xmlns="urn:ietf:params:xml:ns:metalink">
  <published>2008-05-15T12:23:23Z</published>
  <files>
    <file name="example.ext">
      <identity>Example</identity>
      <version>1.0</version>
      <description>A description of the example file for
download.</description>
      <verification>
        <hash type="sha-1">80bc95fd391772fa61c91ed68567f0980bb45fd9
        </hash>
        </verification>
      <resources>
        <url>ftp://ftp.example.com/example.ext</url>
        <url>http://example.com/example.ext</url>
        <metadata type="torrent">
        http://example.com/example.ext.torrent
        </metadata>
      </resources>
    </file>
  </files>
</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 (Second Edition),”

August 2006.) 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 (Fourth Edition),” August 2006.) and identified with the "application/metalink+xml" media type.

Metalink Documents MUST be well-formed XML. This specification does not define a DTD for Metalink Documents, and hence does not require them to be valid (in the sense used by XML).

Metalink allows the use of IRIs [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:base attribute [REC-xmlbase] (Marsh, J., “XML Base,” June 2001.). When xml:base is used in an Metalink Document, it serves the function described in Section 5.1.1 of [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.), establishing the base URI (or IRI) for resolving any relative references found within the effective scope of the xml:base attribute.

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 (Fourth Edition),” August 2006.) [REC-xml], Section 2.12.

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

Metalink is an extensible format. See Section 7 (Extending Metalink) of this document for a full description of how Metalink Documents can be extended.

TOC

3. Common Metalink Constructs

Many of Metalink's elements share a few 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.
Note that there MUST NOT be any white space in a Date construct or in any IRI. Some XML-emitting implementations erroneously insert white space around values by default, and such implementations will emit invalid Metalink Documents.

3.1. Text Constructs

TOC

A Text construct contains human-readable text, usually in small quantities. The content of Text constructs is Language-Sensitive.

```
metalinkTextConstruct =
   metalinkCommonAttributes,
   text
```

3.1.1. Text

TOC

Example 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, Metalink Processors MAY collapse white space (including line breaks) and display the text using typographic techniques such as justification and proportional fonts.

TOC

3.2. Date Constructs

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.), [W3C.NOTE-datetime-19980827] (Wolf, M. and C. Wicksteed, “Date and Time Formats,” August 1998.), and [W3C.REC-xmlschema-2-20041028] (Malhotra, A. and P. Biron, “XML Schema Part 2: Datatypes Second Edition,” October 2004.).
Example Date constructs:

```
<updated>2008-12-13T18:30:02Z</updated>
<updated>2008-12-13T18:30:02.25Z</updated>
<updated>2008-12-13T18:30:02+01:00</updated>
<updated>2008-12-13T18:30:02.25+01:00</updated>
```

Date values SHOULD be as accurate as possible. For example, it would be generally inappropriate for a publishing system to apply the same timestamp to several entries that were published during the course of a single day.

---

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 "metalink:files" element whose element children consist of metadata elements followed by one or more metalink:file (The "metalink:file" Element) child elements.

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

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 exactly one metalink:files (The "metalink:files" Element) element.

* If metalink:type (The "metalink:type" Element) is "dynamic", metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:origin (The "metalink:origin" Element) element.

* metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:type (The "metalink:type" Element) element.

* metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:generator (The "metalink:generator" Element) element.

* metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:published (The "metalink:published" Element) element.

* If metalink:type (The "metalink:type" Element) is "dynamic", metalink:metalink (The "metalink:metalink" Element) elements MAY contain exactly one metalink:updated (The "metalink:updated" Element) 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 advisable 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 when that element is present, and a non-empty metalink:version (The "metalink:version" Element) element, and a non-empty metalink:publisher (The "metalink:publisher" Element) element. However, the absence of metalink:description (The "metalink:description" Element) is not an error, and Metalink Processors MUST NOT fail to function correctly as a consequence of such an absence.

4.1.2. The "metalink:files" Element

TOC

The "metalink:files" element acts 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. Certain elements can be listed either under metalink:files (The "metalink:files" Element) or metalink:file (The "metalink:file" Element). If under metalink:files (The "metalink:files" Element), they apply to all files listed in each metalink:file (The "metalink:file" Element). If under metalink:file (The "metalink:file" Element), then they apply to just that specific file. If an element is listed both under metalink:files (The "metalink:files" Element) and metalink:file (The "metalink:file" Element), then the element under metalink:file (The "metalink:file" Element) has precedence and the metalink:files (The "metalink:files" Element) element does not apply to that particular file.

```
metalinkFiles =
   element metalink:files {
      metalinkCommonAttributes,
      (metalinkIdentity?
       & metalinkVersion?
       & metalinkDescription?
       & metalinkOS?
       & metalinkLogo?
       & metalinkLanguage?
       & metalinkPublisher?
       & metalinkCopyright?
       & metalinkLicense?
       & extensionElement*)
      metalinkFile
   }
```

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

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

* metalink:files (The "metalink:files" Element) elements SHOULD contain exactly one metalink:identity (The "metalink:identity" Element) element.

* metalink:files (The "metalink:files" Element) elements SHOULD contain exactly one metalink:version (The "metalink:version" Element) element.

* metalink:files (The "metalink:files" Element) elements MAY contain exactly one metalink:description (The "metalink:description" Element) element.

* metalink:files (The "metalink:files" Element) elements MAY contain exactly one metalink:os (The "metalink:os" Element) element.

* metalink:files (The "metalink:files" Element) elements MAY contain exactly one metalink:logo (The "metalink:logo" Element) element.

* metalink:files (The "metalink:files" Element) elements MAY contain exactly one metalink:language (The "metalink:language" Element) element.

* metalink:files (The "metalink:files" Element) elements MAY contain exactly one metalink:publisher (The "metalink:publisher" Element) element.

* metalink:files (The "metalink:files" Element) elements MAY contain exactly one metalink:copyright (The "metalink:copyright" Element) element.

* metalink:files (The "metalink:files" Element) elements MAY contain exactly one metalink:license (The "metalink:license" Element) element.

4.1.3. The "metalink:file" Element

TOC

The "metalink:file (The "metalink:file" Element)" element represents an individual file, acting as a container for metadata and data associated with the file.

```
metalinkFile =
   element metalink:file {
      metalinkCommonAttributes,
      attribute name { metalinkTextConstruct },
      (metalinkVerification?
       & metalinkIdentity?
       & metalinkVersion?
       & metalinkDescription?
       & metalinkSize?
       & metalinkOS?
       & metalinkLogo?
       & metalinkLanguage?
       & metalinkPublisher?
       & metalinkCopyright?
       & metalinkLicense?
       & extensionElement*)
      metalinkResources
   }
```

This specification assigns no significance to the order of metalink:file (The "metalink:file" 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 MUST contain exactly one metalink:resources (The "metalink:resources" Element) element.

* metalink:file (The "metalink:file" Element) elements SHOULD contain exactly one metalink:verification (The "metalink:verification" Element) element.

* metalink:file (The "metalink:file" Element) elements SHOULD contain exactly one metalink:identity (The "metalink:identity" Element) element.

* metalink:file (The "metalink:file" Element) elements SHOULD contain exactly one metalink:version (The "metalink:version" Element) element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:description (The "metalink:description" Element) element.

* metalink:file (The "metalink:file" Element) elements SHOULD contain exactly one metalink:size (The "metalink:size" Element) element.

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

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:logo (The "metalink:logo" Element) element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:language (The "metalink:language" Element) element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:publisher (The "metalink:publisher" Element) element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:copyright (The "metalink:copyright" Element) element.

* metalink:file (The "metalink:file" Element) elements MAY contain exactly one metalink:license (The "metalink:license" Element) element.

4.1.3.1. The "name" Attribute

TOC

metalink:file (The "metalink:file" Element) elements MUST have a "name" attribute, which contains the filename of the file downloaded.

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 be relative. The path MUST NOT begin with a "/", "./" or "../", contain "/../", or end with "/..". Metalink Processors MUST NOT allow directory traversal.
A Metalink Processor MAY alter the name of the subdirectory or file if they contain characters which are invalid in the destination filesystem.

4.1.4. The "metalink:resources" Element

TOC

The "metalink:resources" element acts as a container for metadata and data associated with the listed files. It contains one or more metalink:url (The "metalink:url" Element) child elements. It can also contain one or more metalink:metadata (The "metalink:metadata" Element) child elements.

```
metalinkResources =
   element metalink:resources {
      metalinkCommonAttributes,
      extensionElement*
      metalinkURL*
      metalinkMetadata*
   }
```

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

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

4.1.5. The "metalink:verification" Element

TOC

The "metalink:verification" element acts as a container for metadata and data associated with verifying the listed files. This information is in the form of checksums and digital signatures. Checksums are used to verify the integrity of a complete file or portion of a file to determine if the files have been transferred without any errors. Digital signatures verify that a file is from the entity that has signed it.

```
metalinkVerification =
   element metalink:verification {
      metalinkCommonAttributes,
     (metalinkHash*
      & metalinkPieces*
      & metalinkSignature?
      & extensionElement*)
   }
```

The following child elements are defined by this specification:

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

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

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

4.1.6. The "metalink:pieces" Element

TOC

The "metalink:pieces (The "metalink:pieces" Element)" element acts as a container for metadata and data associated with verifying the listed files. This information is in the form of checksums for a portion of a file.

```
metalinkPieces =
   element metalink:pieces {
      attribute length { metalinkTextConstruct },
      attribute type { metalinkTextConstruct },
      hash+
   }+,
```

4.1.6.1. The "type" Attribute

TOC

metalink:pieces (The "metalink:pieces" Element) elements MUST have a "type" attribute.
The IANA registry named "Hash Function Textual Names" defines values for hash types. Metalink Generators and Processors supporting verification SHOULD at least implement "sha-1" which is SHA1, as specified in [RFC3174] (Eastlake, D. and P. Jones, “US Secure Hash Algorithm 1 (SHA1),” September 2001.).

4.1.6.2. The "length" Attribute

TOC

metalink:pieces (The "metalink:pieces" Element) elements MUST have a "length" attribute, which is an integer that describes the length of the piece of the file in octets.

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.

```
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 description for a file.

```
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 agent used to generate a Metalink Document, for debugging and other purposes.

```
metalinkGenerator = element metalink:generator {
   metalinkCommonAttributes,
   attribute uri { metalinkUri }?,
   attribute version { text }?,
   text
}
```

The content of this element, when present, MUST be a string that is a human-readable name for the generating agent. Entities such as "&" and "<" represent their corresponding characters ("&" and "<" respectively), not markup.
The metalink:generator (The "metalink:generator" Element) element MAY have a "uri" attribute whose value MUST be an IRI reference [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” January 2005.). When dereferenced, the resulting URI (mapped from an IRI, if necessary) SHOULD produce a representation that is relevant to that agent.
The metalink:generator (The "metalink:generator" Element) element MAY have a "version" attribute that indicates the version of the generating agent.

4.2.4. The "metalink:hash" Element

TOC

The "metalink:hash (The "metalink:hash" Element)" element is a Text construct that conveys a hash for a file. All hashes are encoded in lowercase hexadecimal format.

```
metalinkHash =
   element metalink:hash {
   attribute piece { xsd:integer }?,
   attribute type { metalinkTextConstruct },
   text
   }
```

Metalinks can contain multiples hashes for a complete file, for example both SHA-1 and SHA-256.

```
...
<verification>
  <hash type="sha-1">a97fcf6ba9358f8a6f62beee4421863d3e52b080</hash>
  <hash type="sha-256">fc87941af7fd7f03e53b34af393f4c14923d74
  825f51116ff591336af4880227</hash>
</verification>
...
```

Metalinks can also contain hashes for individual pieces of a file.

```
...
<verification>
  <hash type="sha-1">a97fcf6ba9358f8a6f62beee4421863d3e52b080</hash>
  <hash type="sha-256">fc87941af7fd7f03e53b34af393f4c14923d74
  825f51116ff591336af4880227</hash>
  <pieces length="1048576" type="sha-1">
    <hash piece="0">d96b9a4b92a899c2099b7b31bddb5ca423bb9b30</hash>
    <hash piece="1">10d68f4b1119014c123da2a0a6baf5c8a6d5ba1e</hash>
    <hash piece="2">3e84219096435c34e092b17b70a011771c52d87a</hash>
    <hash piece="3">67183e4c3ab892d3ebe8326b7d79eb62d077f487</hash>
  </pieces>
</verification>
...
```

metalink:hash (The "metalink:hash" Element) elements MUST have a "type" attribute or a "piece" attribute. metalink:hash (The "metalink:hash" Element) elements with a "type" attribute contain a hash of the whole file. metalink:hash (The "metalink:hash" Element) elements with a "piece" attribute contain a hash for that specific piece or chunk of the file.

TOC

4.2.4.1. The "type" Attribute

The IANA registry named "Hash Function Textual Names" defines values for hash types. Metalink Generators and Processors supporting verification SHOULD at least implement "sha-1" which is SHA1, as specified in [RFC3174] (Eastlake, D. and P. Jones, “US Secure Hash Algorithm 1 (SHA1),” September 2001.).

4.2.4.2. The "piece" Attribute

TOC

metalink:hash (The "metalink:hash" Element) elements MAY have a "piece" attribute, only when they are a sub element of metalink:pieces (The "metalink:pieces" Element). The value of "piece" starts at "0" and increases, depending on the "length" attribute of metalink:pieces (The "metalink:pieces" Element) and the size of the file. Depending on the size of a file, the last piece may not be the same size as the others.

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. The identity of OpenOffice.org 3.0 would be "OpenOffice.org".

```
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 [ISO639-2] (International Organization for Standardization, “ISO 639-2:1998 - Codes for the representation of names of languages -- Part 2: Alpha-3 code - edition 1, 1998-11-01, 66 pages, prepared by a Joint Working Group of ISO TC46/SC4 and ISO TC37/SC2.,” 1998.).

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

4.2.7. The "metalink:license" Element

TOC

The "metalink:license (The "metalink:license" Element)" element is a Text construct that conveys a human-readable license name for a file.

```
metalinkLicense =
   element metalink:license {
   metalinkCommonAttributes,
   attribute uri { metalinkUri }?,
   attribute name { metalinkTextConstruct }?,
   }
```

The metalink:license (The "metalink:license" Element) element MAY have a "uri" attribute whose value MUST be an IRI reference [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” January 2005.). When dereferenced, the resulting URI (mapped from an IRI, if necessary) SHOULD produce a representation that is relevant to that agent.
The metalink:license (The "metalink:license" Element) element MAY have a "name" attribute that indicates the name of the license.

4.2.8. 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.

4.2.9. The "metalink:metadata" Element

TOC

The "metalink:metadata (The "metalink:metadata" Element)" element contains the IRI of metadata about a resource to download. For example, this could be the IRI of a BitTorrent .torrent file or a Metalink Document.

```
metalinkMetadata =
   element metalink:metadata {
      metalinkCommonAttributes,
      attribute preference { xsd:integer }?,
      attribute type { metalinkTextConstruct },
      metalinkUri
   }+
```

4.2.9.1. The "preference" Attribute

TOC

metalink:metadata (The "metalink:metadata" Element) elements MAY have a preference attribute, whose value MUST be a number from 1 to 100 for priority, with 100 used first and 1 used last. See the "preference" attribute of the metalink:url (The "metalink:url" Element) element for more information.

4.2.9.2. The "type" Attribute

TOC

metalink:metadata (The "metalink:metadata" Element) elements MUST have a "type" attribute that indicates the MIME type 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. Metalink Processors that do not support a specified type of metadata about a resource to download MUST ignore that metadata.

4.2.10. The "metalink:origin" Element

TOC

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

```
metalinkOrigin = element metalink:origin {
   metalinkCommonAttributes,
   (metalinkUri)
}
```

4.2.11. The "metalink:os" Element

TOC

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

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

4.2.12. 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.

4.2.13. The "metalink:publisher" Element

TOC

The "metalink:publisher (The "metalink:publisher" Element)" element indicates a group or other entity which has published the file.

```
metalinkPublisher =
   element metalink:publisher {
   metalinkCommonAttributes,
   attribute uri { metalinkUri }?,
   attribute name { metalinkTextConstruct }?,
}
```

The metalink:publisher (The "metalink:publisher" Element) element MAY have a "uri" attribute whose value MUST be an IRI reference [RFC3987] (Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” January 2005.). When dereferenced, the resulting URI (mapped from an IRI, if necessary) SHOULD produce a representation that is relevant to that agent.
The metalink:publisher (The "metalink:publisher" Element) element MAY have a "name" attribute that indicates the name of the publisher.

4.2.14. 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.

```
metalinkSignature =
   element metalink:signature {
   attribute type { "pgp" },
   metalinkTextConstruct
   }
```

4.2.14.1. The "type" Attribute

TOC

metalink:signature (The "metalink:signature" Element) elements MUST have a "type" attribute. The inital value of "type" is the string that is non-empty and matches "pgp". It may be useful to extend Metalink documents with new types of digital signatures, so unknown types are allowed.

4.2.15. The "metalink:size" Element

TOC

The "metalink:size (The "metalink:size" Element)" element indicates the length of the linked content in octets; it is a hint about 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, i.e. files with different sizes should be discarded.

```
metalinkSize =
   element metalink:size {
   metalinkTextConstruct
   }
```

4.2.16. The "metalink:type" Element

TOC

The "metalink:type (The "metalink:type" Element)" element is a Text construct that describes whether the IRI from "metalink:origin (The "metalink:origin" Element)" a Metalink will contain dynamic updated Metalinks or static content that is not updated.

```
metalinkType =
   element metalink:type {
   "static" | "dynamic"
   }
```

4.2.17. 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.

4.2.18. The "metalink:url" Element

TOC

The "metalink:url (The "metalink:url" Element)" element contains the IRI of a file. All IRIs MUST lead to identical files.

```
metalinkURL =
   element metalink:url {
      metalinkCommonAttributes,
      attribute location { xsd:string {
        minLength = "2"  maxLength="2"}
        }?,
      attribute preference { xsd:integer }?,
      metalinkUri
   }+
```

4.2.18.1. The "preference" Attribute

TOC

metalink:url (The "metalink:url" Element) elements MAY have a preference attribute, whose value MUST be a number from 1 to 100 for priority, with 100 used first and 1 used last. Multiple metalink:url (The "metalink:url" Element) elements can have the same preference, i.e. ten mirrors could have preference="100". A Metalink Processor MAY download different segments of a file from more than one IRI simultaneously, and when doing so SHOULD first use the highest priority IRIs and then use lower ones.
When one or more metalink:url (The "metalink:url" Element) elements have a preference attribute value of "100", other metalink:url (The "metalink:url" Element) elements SHOULD NOT be used, unless the elements with a preference of 100 cannot be processed (e.g. if they are of a metalink:metadata (The "metalink:metadata" Element) element type which is not supported by the Metalink Processor, such as BitTorrent, or if the servers are unavailable).
Any metalink:url (The "metalink:url" Element) elements with a preference attribute value of "1" SHOULD NOT be used unless all other metalink:url (The "metalink:url" Element) elements cannot be processed (e.g. if they are of a metalink:metadata (The "metalink:metadata" Element) element type which is not supported by the Metalink Processor, such as BitTorrent, or if the servers are unavailable).

4.2.18.2. The "location" Attribute

TOC

metalink:url (The "metalink:url" Element) elements MAY have a "location" attribute, which is a [ISO3166] (International Organization for Standardization, “ISO 3166:1988 (E/F) - Codes for the representation of names of countries - The International Organization

for Standardization, 3rd edition, 1988-08-15.," 1988.) alpha-2 two letter country code for the geographical location of the physical server an IRI is used to access.

4.2.19. 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 OpenOffice.org 3.0 would be "3.0".

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

---

5. Client Implementation Considerations

TOC

Metalink Processors that support HTTP MUST support transparent content negotiation with HTTP [RFC2295] (Holtman, K. and A. Mutz, "Transparent Content Negotiation in HTTP," March 1998.). Transparent content negotiation is accomplished by adding the Metalink media type to the Accept request header. Metalink Processors MUST check the returned content type, and if the Metalink media type is used, it MUST process the Metalink. If the content type does not match the Metalink media type, then Metalink Processors SHOULD handle the response as a normal response. Metalink Processors MUST NOT add the Metalink media type to Accept when requesting a URI from a metalink:url (The "metalink:url" Element) element, thus avoiding loops. Metalink Processors SHOULD handle external redirects that might lead to a Metalink.
When multiple hash types methods are provided, a Metalink Processor MAY verify using more than one of these hash types. Metalink Processors are encouraged to check all hash types given which they are able to verify.

---

6. Securing Metalink Documents

TOC

Because Metalink is an XML-based format, existing XML security mechanisms can be used to secure its content.
Producers of Metalinks 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 encypt Metalinks that list digital songs that have been purchased. Of course, many other examples exist as well. The algorithm requirements in this section pertain to the Metalink Processor. They require that a recipient, at a minimum, be able to handle messages that use the specified cryptographic algorithms. These requirements do not limit the algorithms that the sender can choose. Metalink Processors that verify signed Metalink Documents MUST at least support XML-Signature and Syntax Processing (Solo, D., Reagle, J., and D. Eastlake, "XML-Signature Syntax and Processing," February 2002.) [REC-xmldsig-core].

---

7. Extending Metalink

TOC

7.1. Extensions from Non-Metalink Vocabularies

TOC

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

7.2. 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".

7.3. Processing Foreign Markup

TOC

Metalink Processors that encounter foreign markup in a location that is legal according to this specification 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.
When unknown foreign markup is encountered in a Text Construct, software SHOULD ignore the markup and process any text content of foreign elements as though the surrounding markup were not present.

7.4. Extension Elements

TOC

Metalink allows foreign markup anywhere in an 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. Child elements of Person constructs are considered to apply to the construct. The role of other foreign markup is undefined by this specification.

7.4.1. 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.

TOC

7.4.2. Structured Extension Elements

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.

---

8. IANA Considerations

TOC

8.1. 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.

8.2. application/metalink+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:
metalink+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: .metalink

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

---

9. Security Considerations

TOC

Publishers are encouraged to offer Metalink documents via authenticated HTTP under TLS as specified in [RFC2818] (Rescorla, E., "HTTP Over TLS," May 2000.). Publishers are also encouraged to include digital signatures of the files within the Metalink Documents if they are available.

9.1. 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.

9.2. Spoofing

TOC

Metalink Processors should be aware of the potential for spoofing attacks where the attacker publishes Metalinks with false information. Malicious publishers might create Metalink Documents containing inaccurate information anywhere in the document. At best, this could deceive unaware downloaders that they are downloading a malicious or worthless file. At worst, malicious publishers could attempt a distributed denial of service attack by inserting unrelated IRIs into Metalink Documents.

9.3. 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 match.
Metalink Generators and Processors supporting verification SHOULD at least implement "sha-1" which is SHA1, as specified in [RFC3174] (Eastlake, D. and P. Jones, “US Secure Hash Algorithm 1 (SHA1),” September 2001.).

9.4. Signing

TOC

Metalink Documents SHOULD be signed using [REC-xmldsig-core] (Solo, D., Reagle, J., and D. Eastlake, “XML-Signature Syntax and Processing,” February 2002.) and are subject to the security considerations implied by its use. This addresses the issue of spoofing.
Digital signatures provide authentication, message integrity, and non-repudiation with proof of origin.

---

10. References

TOC

10.1. Normative References

TOC

[BITTORRENT]	Cohen, B., "The BitTorrent Protocol Specification," BITTORRENT 11031, February 2008.
[ISO3166]	International Organization for Standardization, "ISO 3166:1988 (E/F) - Codes for the representation of names of countries - The International Organization for Standardization, 3rd edition, 1988-08-15.," ISO Standard 3166, 1988.
[ISO639-2]	International Organization for Standardization, "ISO 639-2:1998 - Codes for the representation of names of languages -- Part 2: Alpha-3 code - edition 1, 1998-11-01, 66 pages, prepared by a Joint Working Group of ISO TC46/SC4 and ISO TC37/SC2.," ISO Standard 639-2, 1998.
[REC-xml]	Yergeau, F., Paoli, J., Bray, T., Sperberg-McQueen, C., and E. Maler, "Extensible Markup Language (XML) 1.0 (Fourth Edition)," World Wide Web Consortium Recommendation REC-xml-20060816, August 2006.
[REC-xml-infoset]	Cowan, J. and R. Tobin, "XML Information Set (Second Edition)," World Wide Web Consortium Recommendation REC-xml-infoset-20040204, February 2004.
[REC-xml-names]	Hollander, D., Bray, T., Tobin, R., and A. Layman, "Namespaces in XML 1.0 (Second Edition)," World Wide Web Consortium Recommendation REC-xml-names-20060816, August 2006.
[REC-xmlbase]	Marsh, J., "XML Base," W3C REC W3C.REC-xmlbase-20010627, June 2001.
[REC-xmldsig-core]	Solo, D., Reagle, J., and D. Eastlake, "XML-Signature Syntax and Processing," World Wide Web Consortium Recommendation REC-xmldsig-core-20020212, February 2002.
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.
[RFC2295]	Holtman, K. and A. Mutz, "Transparent Content Negotiation in HTTP," RFC 2295, March 1998.
[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.
[RFC3174]	Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, September 2001.
[RFC3339]	Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps," RFC 3339, July 2002.
[RFC3688]	Mealling, M., "The IETF XML Registry," BCP 81, RFC 3688, January 2004.
[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.

10.2. 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.
[RELAX-NG]	Clark, J., "RELAX NG Compact Syntax," December 2001.
[RFC4287]	Nottingham, M. and R. Sayre, "The Atom Syndication Format," RFC 4287, December 2005.
[W3C.NOTE-datetime-19980827]	Wolf, M. and C. Wicksteed, "Date and Time Formats," W3C NOTE NOTE-datetime-19980827, August 1998.
[W3C.REC-xmlschema-2-20041028]	Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition," W3C REC REC-xmlschema-2-20041028, October 2004.

Appendix A. Contributors

TOC

The layout and content 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 following people contributed to preliminary versions of this document: Paul Burkhead, Kristian Weston, Darius Liktorius, Michael Burford, Giorgio Maone, Manuel Subredu, Tatsuhiro Tsujikawa, A. Bram Neijt, Max Velasques, Manolo Valdes, Urs Wolfer, Frederick Cheung, Nils Maier, Hampus Wessman, Neil McNab, Hayden Legendre, Danny Ayers, Nick Dominguez, Rene Leonhardt, Per Oyvind Karlsen, Gary Zellerbach, James Clark, Daniel Stenberg, Peter Poeml, Matt Domsch, Chris Newman, Lisa Dusseault, Ian Macfarlane, Dave Cridland, Julian Reschke, Barry Leiba, Uri Blumenthal, and Paul Hoffman. The content and concepts within are a product of the Metalink community.

The Metalink community has dozens of contributors who proposed ideas and wording for this document, or contributed to the evolution of Metalink, including:

Nicolas Alvarez, Patrick Ruckstuhl, Mike Wells, Sebastien Willemijns, Micah Cowan, Dan Fandrich, Francis Giannaros, Yazsoft, Lukas Appelhans, KGet developers, FDM Team, Orbit Team, Arne Babenhauserheide, Mathias Berchtold, Xienzhenyu and TheWorld Browser Team, Xi Software, Bridget and Ethan Fletcher, Ruben Kerkhof, Agostino Russo, Gervase Markham, Salvatore and Robin Musumeci, Steve and Rachel Eshelman, Lucas Hewett, Ryan and Darren 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, John Sowder, Sandra Amisano, Tom Mainville, Janie Wargo, Jason Hansen, Tim Bray, Dan Brickley, Markus Hofmann, Dan Connolly, Tim Berners-Lee, Harry Chen, Adrien Macneil, Louis Suarez-Potts, Ross Smith, Rahul Sundaram, Jesse Keating, Michal Bentkowski, Andrew Pantyukhin, Judd Vinet, Charles Landemaine, Pascal Bleser, Jeff@BLAG, Yuichiro Nakada, Jereme Hancock, Marcel Hauser, Jeff Covey, Doug Lang, Seth Brown, Alexander Lazic, Mayank Sharma, Robin Heggelund Hansen, Steve Langasek, Federico Parodi, Stefano Verna, Jason Green, James Linden, Matt Nederlanden, Aren Olsen, Dag Odenhall, Troy Sobotka, Corey Farwell, Ed Lee, Shawn Wilsher, Mike Connor, Anand Muttagi, Dedric Carter, Debi Goulding, the Anthony Family, the Bryan Family, Juanita Anthony and Zimmy Bryan.

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 7.2 (Extensions to the Metalink Vocabulary) and 7.3 (Processing Foreign Markup).

```
# -*- rnc -*-
# RELAX NG Compact Syntax Grammar for the
# Metalink Format Specification Version 2

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


   # Common attributes

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

   # Text Constructs

   metalinkTextConstruct =
      metalinkCommonAttributes,
      text

   # Date Construct

   metalinkDateConstruct =
      metalinkCommonAttributes,
      xsd:dateTime


start =
  element metalink:metalink {
    element metalink:generator {
      attribute uri { metalinkUri }?,
      attribute version { text }?,
      metalinkTextConstruct
    }
    element metalink:origin { metalinkUri }?,
    element metalink:type { "static" | "dynamic"  }?,
    element metalink:published { metalinkDateConstruct }?,
    element metalink:updated { metalinkDateConstruct }?,
    element metalink:files {
      element metalink:file {
        attribute name { metalinkTextConstruct },
        element metalink:identity { metalinkTextConstruct }?,
        element metalink:version { metalinkTextConstruct }?,
        element metalink:size { xsd:integer }?,
        element metalink:description { metalinkTextConstruct }?,
        element metalink:license {
          attribute uri { metalinkUri }?,
          attribute name { metalinkTextConstruct }?,
```

```
        }?,
        element metalink:logo { metalinkUri }?,
        element metalink:publisher {
          attribute uri { metalinkUri }?,
          attribute name { metalinkTextConstruct }?,
        }?,
        element metalink:language { metalinkTextConstruct }?,
        element metalink:copyright { metalinkTextConstruct }?,
        element metalink:license { metalinkTextConstruct }?,
        element metalink:os { metalinkTextConstruct }?,
        element metalink:verification {
          hash+,
          element metalink:pieces {
            attribute length { metalinkTextConstruct },
            attribute type { metalinkTextConstruct },
            hash+
          }+,
          element metalink:signature {
           attribute type { "pgp" },
           text
           }+
        }?,
        element metalink:resources {
          element metalink:metadata {
            attribute preference { xsd:integer }?,
            attribute type { metalinkTextConstruct },
            metalinkUri
          element metalink:url {
            attribute location { xsd:string {
            minLength = "2"  maxLength="2"}
            }?,
            attribute preference { xsd:integer }?,
            metalinkUri
          }+
        }
      }+
    }
  }
hash =
  element metalink:hash {
    attribute piece { metalinkTextConstruct }?,
    attribute type { metalinkTextConstruct },
    text
  }

# As defined in RFC 3066
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:base | xml:lang | local:*) { text }

undefinedContent = (text|anyForeignElement)*

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

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

# EOF
```

Index

TOC

	identity XML element
L	
	language XML element
	license XML element
	logo XML element
M	
	Media Type
	application/metalink+xml
	metadata XML element
	metalink XML element
	metalinkCommonAttributes grammar production
	metalinkCopyright grammar production
	metalinkDateConstruct grammar production
	metalinkDescription grammar production
	metalinkFile grammar production
	metalinkFiles grammar production
	metalinkGenerator grammar production
	metalinkHash grammar production
	metalinkIdentity grammar production
	metalinkLanguage grammar production
	metalinkLicense grammar production
	metalinkLogo grammar production
	metalinkMetalink grammar production
	metalinkOrigin grammar production
	metalinkOS grammar production
	metalinkPieces grammar production
	metalinkPublished grammar production
	metalinkPublisher grammar production
	metalinkResources grammar production
	metalinkSignature grammar production
	metalinkSize grammar production
	metalinkTextConstruct grammar production
	metalinkType grammar production
	metalinkUpdated grammar production
	metalinkURL grammar production 1, 2
	metalinkVerification grammar production
	metalinkVersion grammar production
O	
	origin XML element
	os XML element
P	
	pieces XML element
	published XML element
	publisher XML element
R	
	resources XML element

Author's Address

TOC

	Anthony Bryan (editor)
	Metalinker Project

Email:	anthonybryan@gmail.com
URI:	http://www.metalinker.org