Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2013

A. Bryan
T. Tsujikawa
N. McNab

P. Poeml
MirrorBrain
July 4, 2012

Metalink/XML Clients, Publishers, and Caches
draft-bryan-metalink-client-00

Abstract

This document specifies behavior for Metalink/XML clients, publishers, and proxy caches. way to get information that is usually contained in the Metalink XML-based download description format. Metalink XML files contain multiple download locations (mirrors and Peer-to-Peer), cryptographic hashes, digital signatures, and other information. Metalink clients can use this information to make file transfers more robust and reliable. Normative requirements for Metalink/XML clients, publishers, and proxy caches are described here.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft should take place on the apps-discuss mailing list (apps-discuss@ietf.org), although this draft is not a WG item.

The changes in this draft are summarized in Appendix B.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Notational Conventions 5
1.2. Terminology 5
2. Metalink/XML Clients 6
3. Metalink/XML Publishers and Generators 7
3.1. Metalink/XML Files 8
3.2. Mirror Servers 8
4. Metalink/XML Proxy Cache 8
5. Client / Server Multi-source Download Interaction 9
5.1. Error Prevention, Detection, and Correction 11
5.1.1. Error Prevention (Early File Mismatch Detection) . . . 11
5.1.2. Error Correction 11
6. IANA Considerations 12
7. Security Considerations 12
7.1. URIs and IRIs 12
7.2. Spoofing 12
7.3. Cache Poisoning 12
7.4. Cryptographic Hashes 12
7.5. Signing 13
8. References 13
8.1. Normative References 13
8.2. Informative References 14
Appendix A. Acknowledgements and Contributors 14
Appendix B. Document History 14

1. Introduction

Metalink [RFC5854] 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) from which it is available.

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 cannot 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 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 of

the protocol over which 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.

Metalink/HTTP [RFC6249] is an alternative and complementary representation of Metalink information, using HTTP header fields instead of the XML-based document format [RFC5854]. Metalink/HTTP is used to list information about a file to be downloaded. This can include lists of multiple URIs (mirrors and Peer-to-Peer information), cryptographic hashes, and digital signatures.

Identical copies of a file are frequently accessible in multiple locations on the Internet over a variety of protocols (such as FTP, HTTP, and Peer-to-Peer). In some cases, users are shown a list of these multiple download locations (mirrors) and must manually select a single one on the basis of geographical location, priority, or bandwidth. This distributes the load across multiple servers, and should also increase throughput and resilience. At times, however, individual servers can be slow, outdated, or unreachable, but this can not be determined until the download has been initiated. Users will rarely have sufficient information to choose the most appropriate server, and will often choose the first in a list which might not be optimal for their needs, and will lead to a particular server getting a disproportionate share of load. The use of suboptimal mirrors can lead to the user canceling and restarting the download to try to manually find a better source. During downloads, errors in transmission can corrupt the file. There are no easy ways to repair these files. For large downloads this can be extremely troublesome. Any of the number of problems that can occur during a download lead to frustration on the part of users.

Some popular sites automate the process of selecting mirrors using DNS load balancing, both to approximately balance load between servers, and to direct clients to nearby servers with the hope that this improves throughput. Indeed, DNS load balancing can balance long-term server load fairly effectively, but it is less effective at delivering the best throughput to users when the bottleneck is not the server but the network.

This document describes a mechanism by which the benefit of mirrors can be automatically and more effectively realized. All the information about a download, including mirrors, cryptographic hashes, digital signatures, and more can be transferred in coordinated HTTP header fields hereafter referred to as a Metalink. This Metalink transfers the knowledge of the download server (and

mirror database) to the client. Clients can fallback to other mirrors if the current one has an issue. With this knowledge, the client is enabled to work its way to a successful download even under adverse circumstances. All this can be done without complicated user interaction and the download can be much more reliable and efficient. In contrast, a traditional HTTP redirect to a mirror conveys only minimal information - one link to one server, and there is no provision in the HTTP protocol to handle failures. Furthermore, in order to provide better load distribution across servers and potentially faster downloads to users, Metalink/HTTP facilitates multi-source downloads, where portions of a file are downloaded from multiple mirrors (and optionally, Peer-to-Peer) simultaneously.

Upon connection to a Metalink/HTTP server, a client will receive information about other sources of the same resource and a cryptographic hash of the whole resource. The client will then be able to request chunks of the file from the various sources, scheduling appropriately in order to maximize the download rate.

1.1. Notational Conventions

This specification describes conformance of Metalink/HTTP.

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], as scoped to those conformance targets.

1.2. Terminology

The following terms as used in this document are defined here:

- Metalink Generator : Application that creates Metalink/XML files, and includes information about the files described in the Metalink such as locations (on Mirror servers or other methods like P2P), file sizes, and cryptographic hashes.

- Metalink Publisher : One who uses a Metalink Generator to create Metalink/XML files that are then offered to people to improve their download experience.

- Mirror server : Typically FTP or HTTP servers that "mirror" the Metalink server, as in they provide identical copies of (at least some) files that are also on the mirrored server.

- Metalink/XML : An XML file that can contain similar information to a HTTP response header Metalink, such as mirrors and cryptographic hashes.

- Metalink Processors or Clients : Applications that process Metalink/XML and use them provide an improved download experience. They support HTTP and could also support other download protocols like FTP or various Peer-to-Peer methods.

2. Metalink/XML Clients

In this context, "Metalink" refers to "Metalink/XML" refers to the XML format described in [RFC5854].

Metalink clients use the mirrors provided by a Metalink/XML file. Metalink clients SHOULD support HTTP [RFC2616] and SHOULD support FTP [RFC0959]. Metalink clients MAY support BitTorrent [BITTORRENT], or other download methods. Metalink clients SHOULD switch downloads from one mirror to another if a mirror becomes unreachable. Metalink clients MAY support multi-source, or parallel, downloads, where portions of a file can be downloaded from multiple mirrors simultaneously (and optionally, from Peer-to-Peer sources). Metalink clients SHOULD support error recovery by using the cryptographic hashes of parts of the file listed in Metalink/XML files. Metalink clients SHOULD support checking digital signatures.

Metalink/XML clients MUST sanitize directory traversal information as specified in [RFC5854] Section 4.1.2.1. Also see [RFC2183] Section 5 and [RFC6266] Section 4.3.

Metalink/XML clients MUST process metalinks available by URI. They MAY process local Metalinks.

Metalink/XML clients SHOULD recognize Metalink/XML files by MIME type. (What about misconfigured/unupdated servers that do not have correct MIME type?) SHOULD(?) client recognize metalink by file extension as well?

If Metalink/XML clients support HTTP, SHOULD(?) support "transparent metalink" usage from regular download to Metalink/XML (see Section 3.1).

If Metalink/XML clients support HTTP, MAY do Accept header transparent content negotiation. (deprecated?)

If a file with same name already exists locally, Metalink/XML clients SHOULD verify full file hash and if hash is correct, do not re-download the file. If a file exists and full file hash is incorrect, Metalink/XML clients MAY repair file if partial file hashes exist. otherwise, MAY write to other file name (file_2 or file(2) like some apps already do).

Metalink/XML clients SHOULD (or MUST?) verify full file hash after download completes. if error, MUST describe as corrupted and MAY re-download or keep download? SHOULD verify chunk hash if available and re-get error parts. SHOULD (or MAY?) be done during initial download process, MAY be done after download completed or to repair file downloaded another way?

Metalink/XML clients SHOULD(?) use BitTorrent chunk hashes with HTTP/FTP downloads to repair file if client supports torrents. (What if chunk hashes are present in torrent and metalink, should one be preferred?)

If client supports Metalink/XML AND Metalink/HTTP, which should be preferred (in case mirrors/hashes differ)?

Metalink/XML clients SHOULD make use of Metalink/XML origin element if dynamic="true" to check for updated Metalink.

Metalink/XML clients MAY make use of the [ISO3166-1] alpha-2 two-letter country code for the geographical location of the physical server the URI is used to access, in an attempt to improve the download experience.

Metalink clients SHOULD? use the location of the original Metalink in the "Referer" header field for these ranged requests.

Metalink clients MAY support the use of metainfo files (such as BitTorrent) for downloading files.

Metalink clients SHOULD support the use of OpenPGP signatures.

Metalink clients SHOULD support the use of S/MIME [RFC5751] signatures.

[[NOTE: A number of requirements of Metalink clients are also in [RFC5854]. Should these be repeated or referenced?]]

3. Metalink/XML Publishers and Generators

Metalink/XML publishers MUST use correct MIME type for metalink files

Metalink/XML publishers SHOULD advertise Metalink/XML file with Link HTTP header field from regular download for "transparent metalink" usage (see Section 3.1).

Metalink/XML publishers SHOULD publish with chunk hashes if error recovery ability is desired (and files meet certain criteria like "large enough" - no point for 10k size file).

Metalink Generators SHOULD offer Metalink/XML documents that contain cryptographic hashes of parts of the file (and other information) if error recovery is desirable.

Metalink/XML publishers SHOULD publish with size element if it refers to a specific file.

Metalink/XML publishers MAY do Accept header transparent content negotiation (deprecated?)

Metalink/XML publishers SHOULD include Metalink/XML origin element and dynamic="true" if updated metalinks will be offered.

Metalink publishers SHOULD include digital signatures, as described in [RFC5854] Section 4.2.13.

3.1. Metalink/XML Files

Metalink/XML files for a given resource MAY be provided in a Link header field [RFC5988] as shown in this example:

This example shows a brief HTTP response header with .meta4:

```
Link: <http://example.com/example.ext.meta4>; rel=describedby;
type="application/metalink4+xml"
```

Metalink/XML files are specified in [RFC5854], and they are particularly useful for providing metadata such as cryptographic hashes of parts of a file (see [RFC5854] Section 4.1.3), allowing a client to recover from errors (see Section 5.1.2). Metalink servers SHOULD provide Metalink/XML files with partial file hashes in Link header fields, and Metalink clients SHOULD use them for error recovery.

3.2. Mirror Servers

Mirror servers are typically FTP or HTTP servers that "mirror" another server. That is, they provide identical copies of (at least some) files that are also on the mirrored server. Mirror servers SHOULD support serving partial content.

4. Metalink/XML Proxy Cache

Metalink/XML proxy cache could detect and log Metalink usage.

Metalink/XML proxy cache MUST? use a whitelist for trusted sources by domain name (ie kde.org, ubuntu.com, fedoraproject.org) to prevent cache poisoning.

Metalink/XML proxy cache SHOULD use preferred mirrors (those that are most cost efficient/better/local)

Metalink/XML proxy cache MAY? repair errors or use hashes? I guess so, but the client will also be verifying hashes.

5. Client / Server Multi-source Download Interaction

Metalink clients begin with a Metalink/XML document. They parse the XML and obtain a list of ways to retrieve a file or files from FTP or HTTP mirrors or P2P.

After that, the client follows with a GET request to the desired mirrors.

From the Metalink/XML file, the client learns some or all of the following metadata about the requested object:

- o Mirror list, which can describe the mirror's priority and geographical location.

- o Whole and partial file cryptographic hash.

- o Object size.

- o Peer-to-peer information.

- o Digital signature.

Next, the Metalink client requests a Range of the object from a mirror server:

```
GET /example.ext HTTP/1.1
Host: www2.example.com
Range: bytes=7433802-
Referer: http://www.example.com/distribution/example.ext
```

Metalink clients SHOULD use partial file cryptographic hashes as described in Section 5.1.2, if available, to detect if the mirror server returned the correct data. Errors in transmission and substitutions of incorrect data on mirrors, whether deliberate or accidental, can be detected with error correction as described in Section 5.1.2.

Here, the mirror server has the correct file and responds with a 206 Partial Content HTTP status code and appropriate "Content-Length" and "Content Range" header fields. In this example, the mirror server responds, with data, to the above request:

```
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Length: 7433801
Content-Range: bytes 7433802-14867602/14867603
```

Metalink clients MAY start a number of parallel ranged downloads (one per selected mirror server other than the first) using mirrors provided by the Metalink/XML. Metalink clients MUST limit the number of parallel connections to mirror servers, ideally based on observing how the aggregate throughput changes as connections are opened. It would be pointless to blindly open connections once the path bottleneck is filled. After establishing a new connection, a Metalink client SHOULD monitor whether the aggregate throughput increases over all connections that are part of the download. The client SHOULD NOT open additional connections during this period. If the aggregate throughput has increased, the client MAY open an additional connection and repeat these steps. Otherwise, the client SHOULD NOT open a new connection until an established one closes.

The Metalink client can determine the size and number of ranges requested from each server, based upon the type and number of mirrors and performance observed from each mirror. Note that Range requests impose an overhead on servers and clients need to be aware of that and not abuse them. When dowloading a particular file, metalink clients MUST NOT make more than one concurrent request to each mirror server that it downloads from.

Metalink clients SHOULD close all but the fastest connection if any Ranged requests generated after the first request end up with a complete response, instead of a partial response (as some mirrors might not support HTTP ranges), if the goal is the fastest transfer. Metalink clients MAY monitor mirror conditions and dynamically switch between mirrors to achieve the fastest download possible. Similarly, Metalink clients SHOULD abort extremely slow or stalled range requests and finish the request on other mirrors. If all ranges have finished except for the final one, the Metalink client can split the final range into multiple range requests to other mirrors so the transfer finishes faster.

Metalink clients MUST reject individual downloads from mirrors where the file size does not match the file size as reported by the Metalink server.

If a Metalink client does not support certain download methods (such as FTP or BitTorrent) that a file is available from, and there are no available download methods that the client supports, then the download will have no way to complete.

Metalink clients MUST verify the cryptographic hash of the file once the download has completed. If the cryptographic hash offered in the Metalink/XML does not match the cryptographic hash of the downloaded file, see Section 5.1.2 for a possible way to repair errors.

If the download can not be repaired, it is considered corrupt. The client can attempt to re-download the file.

Metalink clients that support verifying digital signatures MUST verify digital signatures of requested files if they are included. Digital signatures MUST validate back to a trust anchor as described in the validation rules in [RFC3156] and [RFC5280].

5.1. Error Prevention, Detection, and Correction

Error prevention, or early file mismatch detection, is possible before file transfers with the use of file sizes provided in Metalink/XML. Error detection requires full file cryptographic hashes in the Metalink/XML to detect errors in transfer after the transfers have completed. Error correction, or download repair, is possible with partial file cryptographic hashes.

5.1.1. Error Prevention (Early File Mismatch Detection)

To verify the individual ranges of files, which might have been requested from different sources, see Section 5.1.2.

5.1.2. Error Correction

Partial file cryptographic hashes can be used to detect errors during the download. Metalink servers SHOULD provide Metalink/XML files with partial file hashes in Link header fields as specified in Section 3.1, and Metalink clients SHOULD use them for error correction.

An error in transfer or a substitution attack will be detected by a cryptographic hash of the object not matching the full file checksum from the Metalink/XML. If the cryptographic hash of the object does not match the full file checksum from the Metalink/XML, then the client SHOULD use the partial file cryptographic hashes (if available). This may contain partial file cryptographic hashes which will allow detection of which mirror server returned incorrect data. Metalink clients SHOULD use the Metalink/XML data to figure out what ranges of the downloaded data can be recovered and what needs to be fetched again.

Other methods can be used for error correction. For example, some other metainfo files also include partial file hashes that can be

used to check for errors.

6. IANA Considerations

None.

7. Security Considerations

7.1. URIs and IRIs

Metalink clients handle URIs and IRIs. See Section 7 of [RFC3986] and Section 8 of [RFC3987] for security considerations related to their handling and use.

7.2. Spoofing

There is potential for spoofing attacks where the attacker publishes Metalinks with false information. In that case, this could deceive unaware downloaders into downloading a malicious or worthless file. As with all downloads, users should only download from trusted sources. Also, malicious publishers could attempt a distributed denial of service attack by inserting unrelated URIs into Metalinks. [RFC4732] contains information on amplification attacks and denial of service attacks.

7.3. Cache Poisoning

Proxy caches MUST prevent cache poisoning.

7.4. Cryptographic Hashes

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 that are not suitable for cryptographically strong verification. Malicious parties 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], 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".

7.5. Signing

Metalinks SHOULD include digital signatures, as described in [RFC5854] Section 4.2.13.

Digital signatures provide authentication, message integrity, and enable non-repudiation with proof of origin.

8. References

8.1. Normative References

[BITTORRENT] Cohen, B., "The BitTorrent Protocol Specification", BITTORRENT 11031, February 2008, <http://www.bittorrent.org/beps/bep_0003.html>.

[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.

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 0959, October 1985.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, August 2001.

[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.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation

List (CRL) Profile", RFC 5280, May 2008.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC5854] Bryan, A., Tsujikawa, T., McNab, N., and P. Poeml, "The Metalink Download Description Format", RFC 5854, June 2010.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

8.2. Informative References

[RFC2183] Troost, R., Dorner, S., and K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997.

[RFC4732] Handley, M., Rescorla, E., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, December 2006.

[RFC6249] Bryan, A., McNab, N., Tsujikawa, T., Poeml, P., and H. Nordstrom, "Metalink/HTTP: Mirrors and Hashes", RFC 6249, June 2011.

[RFC6266] Reschke, J., "Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)", RFC 6266, June 2011.

Appendix A. Acknowledgements and Contributors

Some text borrowed from our previous RFCs: [RFC5854] and [RFC6249].

Thanks to the Metalink community.

This document is dedicated to Zimmy Bryan and Juanita Anthony.

Appendix B. Document History

[[to be removed by the RFC editor before publication as an RFC.]]

Known issues concerning this draft:

- o Intro, term, downloads.

- o Repeat requirements from RFC5854 or add pointer to them?

-00 : July 4, 2012.

o Initial draft.

Authors' Addresses

Anthony Bryan
Pompano Beach, FL
USA

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

Tatsuhiro Tsujikawa
Shiga
Japan

EMail: tatsuhiro.t@gmail.com
URI: http://aria2.sourceforge.net

Neil McNab

EMail: neil@nabber.org
URI: http://www.nabber.org

Dr. med. Peter Poeml
MirrorBrain
Venloer Str. 317
Koeln 50823
DE

Phone: +49 221 6778 333 8
EMail: peter@poeml.de
URI: http://mirrorbrain.org/~poeml/