Internet Engineering Task Force (IETF)
Request for Comments: 6249
Category: Standards Track
ISSN: 2070-1721

A. Bryan
N. McNab
T. Tsujikawa

P. Poeml
MirrorBrain
H. Nordstrom

June 2011

Metalink/HTTP: Mirrors and Hashes

Abstract

This document specifies Metalink/HTTP: Mirrors and Cryptographic Hashes in HTTP header fields, a different way to get information that is usually contained in the Metalink XML-based download description format. Metalink/HTTP describes multiple download locations (mirrors), Peer-to-Peer, cryptographic hashes, digital signatures, and other information using existing standards for HTTP header fields. Metalink clients can use this information to make file transfers more robust and reliable. Normative requirements for Metalink/HTTP clients and servers are described here.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6249.

Table of Contents

1. Introduction

Metalink/HTTP is an alternative and complementary representation of Metalink information, which is usually presented as an XML-based document format [RFC5854]. Metalink/HTTP attempts to provide as much functionality as the Metalink/XML format by using existing standards, such as Web Linking [RFC5988], Instance Digests in HTTP [RFC3230], and Entity Tags (also known as ETags) [RFC2616]. Metalink/HTTP is used to list information about a file to be downloaded. This can include lists of multiple URIs (mirrors), 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 At times, however, individual servers can be slow, outdated, or unreachable, but this cannot be determined until the download has been initiated.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 fall back 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. Example Metalink Server Response

This example shows a brief Metalink server response with ETag, mirrors, Peer-to-Peer information, Metalink/XML, OpenPGP signature, and a cryptographic hash of the whole file:

```
Etag: "thvDyvhfIqlvFe+A9MYgxAfm1q5="
Link: <http://www2.example.com/example.ext>; rel=duplicate
Link: <ftp://ftp.example.com/example.ext>; rel=duplicate
Link: <http://example.com/example.ext.torrent>; rel=describedby;
type="application/x-bittorrent"
Link: <http://example.com/example.ext.meta4>; rel=describedby;
type="application/metalink4+xml"
Link: <http://example.com/example.ext.asc>; rel=describedby;
type="application/pgp-signature"
Digest: SHA-256=MWVkMWQxYTRiMzk5MDQ0MzI3NGU5NDEyZTk5OWY1ZGFmNzgyZTJlO
DYzYjRjYzFhOTlmNTQwYzI2M2QwM2U2MQ==
```

1.2. 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.

1.3. Terminology

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

- Metalink server: HTTP server that provides a Metalink in HTTP response header fields.
- Metalink : A collection of HTTP response header fields from a Metalink server, which is the reply to a GET or HEAD request from a client and which includes Link header fields listing mirrors and Instance Digests listing a cryptographic hash.
- Link header field: HTTP response header field, defined in [RFC5988], that can list mirrors and, potentially, other download methods for obtaining a file, along with digital signatures.
- Instance Digest: HTTP response header field, defined in [RFC3230], that contains the cryptographic hash of a file, which is used by the Metalink client to verify the integrity of the file once the download has completed.
- Entity Tag or ETag: HTTP response header field, defined in [RFC2616], that, if synchronized between the Metalink server and mirror servers, allows Metalink clients to provide advanced features.
- Mirror server: Typically, FTP or HTTP servers that "mirror" the Metalink server, i.e., provide identical copies of (at least some) files that are also on the mirrored server.
- Metalink clients: Applications that process Metalinks and use them to provide an improved download experience. They support HTTP and could also support other download protocols like FTP or various Peer-to-Peer methods.
- Metalink/XML: An XML file that can contain similar information to an HTTP response header Metalink, such as mirrors and cryptographic hashes.

2. Requirements

In this context, "Metalink" refers to Metalink/HTTP, which consists of mirrors and cryptographic hashes in HTTP header fields as described in this document. "Metalink/XML" refers to the XML format described in [RFC5854].

Metalink resources include Link header fields [RFC5988] to present a list of mirrors in the response to a client request for the resource. Metalink servers MUST include the cryptographic hash of a resource via Instance Digests in HTTP [RFC3230]. Algorithms used in the Instance Digest field are registered in the IANA registry named "Hypertext Transfer Protocol (HTTP) Digest Algorithm Values" at <http://www.iana.org/>. This document restricts the use of these algorithms. SHA-256 and SHA-512 were added to the registry by [RFC5843]. Metalinks contain whole file hashes as described in Section 6, and MUST include SHA-256, as specified in [FIPS-180-3]. It MAY also include other hashes.

Metalink servers are HTTP servers with one or more Metalink resources. Metalink servers MUST support the Link header fields for listing mirrors and MUST support Instance Digests in HTTP [RFC3230]. Metalink servers MUST return the same Link header fields and Instance Digests on HEAD requests. Metalink servers and their associated preferred mirror servers MUST all share the same ETag policy. Metalink servers and their associated normal mirror servers SHOULD all share the same ETag policy. (See Section 3.3 for the definition of "preferred" and "normal" mirror servers.) It is up to the administrator of the Metalink server to communicate the details of the shared ETag policy to the administrators of the mirror servers so that the mirror servers can be configured with the same ETag policy. To have the same ETag policy means that ETags are synchronized across servers for resources that are mirrored; i.e., byte-for-byte identical files will have the same ETag on mirrors that they have on the Metalink server. For example, it would be better to derive an ETag from a cryptographic hash of the file contents than on server-unique filesystem metadata. Metalink servers SHOULD offer Metalink/XML documents that contain cryptographic hashes of parts of the file (and other information) if error recovery is desirable.

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. HTTP mirror servers SHOULD share the same ETag policy as the originating Metalink server. HTTP mirror servers SHOULD support Instance Digests in HTTP [RFC3230] using the same algorithm as the Metalink server. Optimally, HTTP mirror servers will share the same ETag policy and support Instance Digests in HTTP. Mirror servers that share the same ETag policy and/or support Instance Digests in HTTP using the same algorithm as a Metalink server are known as preferred mirror servers.

Metalink clients use the mirrors provided by a Metalink server in Link header fields [RFC5988] but these clients are restricted to using the mirrors provided by the initial Metalink server they

contacted. If Metalink clients find Link header fields [RFC5988] from mirrors that in turn list mirrors, or from a Metalink server listing itself as a mirror, they MUST discard such Link header fields [RFC5988] to prevent a possible infinite loop. Metalink clients MUST support HTTP 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 MUST support Instance Digests in HTTP [RFC3230] by requesting and verifying cryptographic hashes. 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.

3. Mirrors / Multiple Download Locations

Mirrors are specified with the Link header fields [RFC5988] and a relation type of "duplicate" as defined in Section 8.

The following list contains OPTIONAL attributes, which are defined elsewhere in this document:

- "depth" : mirror depth (see Section 3.4).
- "geo" : mirror geographical location (see Section 3.2).
- "pref" : a preferred mirror server (see Section 3.3).
- "pri" : mirror priority (see Section 3.1).

This example shows a brief Metalink server response with two mirrors only:

```
Link: <http://www2.example.com/example.ext>; rel=duplicate;
pri=1; pref
Link: <ftp://ftp.example.com/example.ext>; rel=duplicate;
pri=2; geo=gb; depth=1
```

As some organizations can have many mirrors, it is up to the organization to configure the amount of Link header fields the Metalink server will provide. Such a decision could be a random selection or a hard-coded limit based on network proximity, file size, server load, or other factors.

3.1. Mirror Priority

Entries for mirror servers MAY have a "pri" value to designate the priority of a mirror. Valid ranges for the "pri" attribute are from 1 to 999999. Mirror servers with a lower value of the "pri" attribute have a higher priority, while mirrors with an undefined "pri" attribute are considered to have a value of 999999, which is the lowest priority. For example, a mirror with "pri=10" has higher priority than a mirror with "pri=20". Metalink clients SHOULD use mirrors with lower "pri" values first, but depending on other conditions, they MAY decide to use other mirrors.

This is purely an expression of the server's preferences; it is up to the client what it does with this information, particularly with reference to how many servers to use at any one time.

3.2. Mirror Geographical Location

Entries for a mirror server MAY have a "geo" value, which is an [ISO3166-1] alpha-2 two-letter country code for the geographical location of the physical server the URI is used to access. A client MAY use this information to select a mirror, or set of mirrors, that is geographically near (if the client has access to such information), with the aim of reducing network load at inter-country bottlenecks.

3.3. Coordinated Mirror Policies

There are two types of mirror servers: preferred and normal. Entries for preferred HTTP mirror servers have a "pref" value and entries for normal mirrors don't. Preferred mirror servers are HTTP mirror servers that MUST share the same ETag policy as the originating Metalink server, or if the ETag is not used MUST provide an Instance Digest using the same algorithm as the Metalink server. Preferred mirrors make it possible for Metalink clients to detect early on, before data is transferred, if the file requested matches the desired file. This early file mismatch detection is described in Section 7.1.1. Normal mirrors do not necessarily share the same ETag policy or support Instance Digests using the same algorithm as the Metalink server. FTP mirrors are considered "normal", as they do not emit ETags or support Instance Digests.

3.4. Mirror Depth

Some mirrors can mirror single files, whole directories, or multiple directories.

Entries for mirror servers can have a "depth" value, where "depth=0" is the default. A value of 0 means that only that file is mirrored and that other URI path segments are not. A value of 1 means that the file and all other files and URI path segments contained in the rightmost URI path segment are mirrored. For values of N, N-1 URI path segments closer to the Host are mirrored. A value of 2 means one URI path segment closer to the Host is mirrored, and all files and URI path segments contained are mirrored. For each higher value, another URI path segment closer to the Host is mirrored.

This example shows a mirror with a depth value of 4:

```
Link: <http://www2.example.com/dir1/dir2/dir3/dir4/dir5/example.ext>;
rel=duplicate; pri=1; pref; depth=4
```

In the above example, four URI path segments closer to the Host are mirrored, from /dir2/ and all files and directories included.

4. Peer-to-Peer / Metainfo

Entries for metainfo files, which describe ways to download a file over Peer-to-Peer networks or otherwise, are specified with the Link header fields [RFC5988] and a relation type of "describedby" and a type parameter that indicates the MIME type of the metadata available at the URI. Since metainfo files can sometimes describe multiple files, or the filename MAY not be the same on the Metalink server and in the metainfo file but MAY still have the same content, an OPTIONAL "name" attribute can be used.

The following list contains an OPTIONAL attribute, which is defined in this document:

- "name" : a file described within the metainfo file.

This example shows a brief Metalink server response with .torrent and .meta4:

```
Link: <http://example.com/example.ext.torrent>; rel=describedby;
type="application/x-bittorrent"; name="differentname.ext"
Link: <http://example.com/example.ext.meta4>; rel=describedby;
type="application/metalink4+xml"
```

Metalink clients MAY support the use of metainfo files for downloading files.

4.1. Metalink/XML Files

Metalink/XML files for a given resource MAY be provided in a Link header field as shown in the example in Section 4. Metalink/XML files are specified in [RFC5854], and they are particularly useful for providing metadata such as cryptographic hashes of parts of a file, allowing a client to recover from errors (see Section 7.1.2). Metalink servers SHOULD provide Metalink/XML files with partial file hashes in Link header fields, and Metalink clients SHOULD use them for error recovery.

5. Signatures

5.1. OpenPGP Signatures

OpenPGP signatures [RFC3156] of requested files are specified with the Link header fields [RFC5988] and a relation type of "describedby" and a type parameter of "application/pgp-signature".

This example shows a brief Metalink server response with OpenPGP signature only:

Link: <http://example.com/example.ext.asc>; rel=describedby; type="application/pgp-signature"

Metalink clients SHOULD support the use of OpenPGP signatures.

5.2. S/MIME Signatures

Secure/Multipurpose Internet Mail Extensions (S/MIME) signatures [RFC5751] of requested files are specified with the Link header fields [RFC5988] and a relation type of "describedby" and a type parameter of "application/pkcs7-mime".

This example shows a brief Metalink server response with S/MIME signature only:

Link: <http://example.com/example.ext.p7m>; rel=describedby; type="application/pkcs7-mime"

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

6. Cryptographic Hashes of Whole Documents

If Instance Digests are not provided by the Metalink servers, the Link header fields pertaining to this specification MUST be ignored.

This example shows a brief Metalink server response with ETag, mirror, and cryptographic hash:

```
Etag: "thvDyvhfIqlvFe+A9MYgxAfm1q5="
Link: <http://www2.example.com/example.ext>; rel=duplicate
Digest: SHA-256=MWVkMWQxYTRiMzk5MDQ0MzI3NGU5NDEyZTk5OWY1ZGFmNzgyZTJlO
DYzYjRjYzFhOTlmNTQwYzI2M2QwM2U2MQ==
```

7. Client / Server Multi-Source Download Interaction

Metalink clients begin a download with a standard HTTP [RFC2616] GET request to the Metalink server. Metalink clients MAY use a range limit if desired.

```
GET /distribution/example.ext HTTP/1.1
Host: www.example.com
```

The Metalink server responds with the data and these header fields:

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 14867603
Content-Type: application/x-cd-image
Etag: "thvDyvhfIqlvFe+A9MYgxAfm1q5="
Link: <http://www2.example.com/example.ext>; rel=duplicate; pref
Link: <ftp://ftp.example.com/example.ext>; rel=duplicate
Link: <http://example.com/example.ext.torrent>; rel=describedby;
type="application/x-bittorrent"
Link: <http://example.com/example.ext.meta4>; rel=describedby;
type="application/metalink4+xml"
Link: <http://example.com/example.ext.asc>; rel=describedby;
type="application/pgp-signature"
Digest: SHA-256=MWVkMWQxYTRiMzk5MDQ0MzI3NGU5NDEyZTk5OWY1ZGFmNzgyZTJlO
DYzYjRjYzFhOTlmNTQwYzI2M2QwM2U2MQ==
```

Alternatively, Metalink clients can begin with a HEAD request to the Metalink server to discover mirrors via Link header fields and then skip to making the following decisions on every available mirror server found via the Link header fields.

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

From the Metalink server response, the client learns some or all of the following metadata about the requested object, in addition to starting to receive the object:

- Mirror locations, with optional attributes describing the mirror's priority, whether it shares the ETag policy of the originating Metalink server, geographical location, and mirror depth.
- Instance Digest, which is the whole file cryptographic hash.
- ETag.
- Object size from the Content-Length header field.
- Metalink/XML, which can include partial file cryptographic hashes to repair a file.
- Peer-to-Peer information.
- Digital signature.

Next, the Metalink client requests a range of the object from a preferred mirror server, so it can use If-Match conditions:

```
GET /example.ext HTTP/1.1
Host: www2.example.com
Range: bytes=7433802-
If-Match: "thvDyvhfIqlvFe+A9MYgxAfm1q5="
Referer: http://www.example.com/distribution/example.ext
```

Metalink clients SHOULD use preferred mirrors, if possible, as they allow early file mismatch detection as described in Section 7.1.1. Preferred mirrors have coordinated ETags, as described in Section 3.3, and Metalink clients SHOULD use If-Match conditions based on the ETag to quickly detect out-of-date mirrors by using the ETag from the Metalink server response. Metalink clients SHOULD use partial file cryptographic hashes as described in Section 7.1.2, if available, to detect if the mirror server returned the correct data.

Optimally, the mirror server also will include an Instance Digest in the mirror response to the client GET request, which the client can also use to detect a mismatch early. Metalink clients MUST reject individual downloads from mirrors that support Instance Digests if the Instance Digest from the mirror does not match the Instance Digest as reported by the Metalink server and the same algorithm is

used. If normal mirrors are used, then a mismatch cannot be detected until the completed object is verified. Errors in transmission and substitutions of incorrect data on mirrors, whether deliberate or accidental, can be detected with error correction as described in Section 7.1.2.

Here, the preferred mirror server has the correct file (the If-Match conditions match) and responds with a 206 Partial Content HTTP status code and appropriate "Content-Length", "Content-Range", ETag, and Instance Digest 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
Etag: "thvDyvhfIqlvFe+A9MYgxAfm1q5="
Digest: SHA-256=MWVkMWQxYTRiMzk5MDQ0MzI3NGU5NDEyZTk5OWY1ZGFmNzgyZTJlO
DYzYjRjYzFhOTlmNTQwYzI2M2QwM2U2MQ==
```

Metalink clients MAY start a number of parallel range requests (one per selected mirror server other than the first) using mirrors provided by the Link header fields with "duplicate" relation type. 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. Metalink clients SHOULD use the location of the original GET request in the "Referer" header field for these range requests.

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 downloading a particular file, Metalink clients MUST NOT make more than one concurrent request to each mirror server from which it downloads.

Metalink clients SHOULD close all but the fastest connection if any range 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.

If the first request was a GET, no Range header field was sent, and the client determines later that it will issue a range request, then the client SHOULD close the first connection when it catches up with the other parallel range requests of the same object. This means the first connection was sacrificed. Metalink clients can use a HEAD request first, if possible, so that the client can find out if there are any Link header fields, and then range-based requests are undertaken to the mirror servers without sacrificing a first connection.

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 by the Metalink server with Instance Digests does not match the cryptographic hash of the downloaded file, see Section 7.1.2 for a possible way to repair errors.

If the download cannot 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].

7.1. Error Prevention, Detection, and Correction

Error prevention, or early file mismatch detection, is possible before file transfers with the use of file sizes, ETags, and Instance Digests provided by Metalink servers. Error detection requires Instance Digests to detect errors in transfer after the transfers have completed. Error correction, or download repair, is possible with partial file cryptographic hashes.

Note that cryptographic hashes obtained from Instance Digests are in base64 encoding, while those from Metalink/XML are in hexadecimal.

7.1.1. Error Prevention (Early File Mismatch Detection)

In HTTP terms, the merging of ranges from multiple responses SHOULD be verified with a strong validator, which in this context is either an Instance Digest or a shared ETag from that Metalink server that matches with the Instance Digest or ETag provided by a preferred mirror server. In most cases, it is sufficient that the Metalink server provides mirrors and Instance Digest information, but operation will be more robust and efficient if the mirror servers do implement a shared ETag policy or Instance Digests as well. There is no need to specify how the ETag is generated, just that it needs to be shared between the Metalink server and the mirror servers. The benefit of having mirror servers return an Instance Digest is that the client then can detect mismatches early even if ETags are not used. Mirrors that support both a shared ETag and Instance Digests do provide value, but just one is sufficient for early detection of mismatches. If the mirror server provides neither shared ETag nor Instance Digest, then early detection of mismatches is not possible unless file length also differs. Finally, errors are still detectable after the download has completed, when the cryptographic hash of the merged response is verified.

ETags cannot be used for verifying the integrity of the received content. If the ETag given by the mirror server matches the ETag given by the Metalink server, then the Metalink client assumes the responses are valid for that object.

This guarantees that a mismatch will be detected by using only the shared ETag from a Metalink server and mirror server. Metalink clients will detect an error if ETags do not match, which will prevent accidental merges of ranges from different versions of files with the same name.

A shared ETag or Instance Digest cannot strictly protect against malicious attacks or server or network errors replacing content. An attacker can make a mirror server seemingly respond with the expected

Instance Digest or ETags even if the file contents have been modified. The same goes for various system failures, which would also cause bad data (i.e., corrupted files) to be returned. The Metalink client has to rely on the Instance Digest returned by the Metalink server in the first response for the verification of the downloaded object as a whole. To verify the individual ranges, which might have been requested from different sources, see Section 7.1.2.

7.1.2. 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 4.1, 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 Instance Digest from the Metalink server. If the cryptographic hash of the object does not match the Instance Digest from the Metalink server, then the client SHOULD fetch the Metalink/XML (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.

8. IANA Considerations

Accordingly, IANA has made the following registration to the "Link Relation Types" registry at <http://www.iana.org/>.

- Relation Name: duplicate
- Description: Refers to a resource whose available representations are byte-for-byte identical with the corresponding representations of the context IRI.
- Reference: This specification.
- Notes: This relation is for static resources. That is, an HTTP GET request on any duplicate will return the same representation. It does not make sense for dynamic or POSTable resources and should not be used for them.

9. Security Considerations

9.1. URIs and IRIs

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

9.2. 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. Metalink clients are advised to prevent loops, possibly from a mirror server to a Metalink server and back again, in Section 2. 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.

9.3. Cryptographic Hashes

Currently, some of the digest values defined in Instance Digests in HTTP [RFC3230] 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.

9.4. Signing

Metalinks SHOULD include digital signatures, as described in Section 5.

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

10. References

10.1. 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.

[RFC3230] Mogul, J. and A. Van Hoff, "Instance Digests in HTTP", RFC 3230, January 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.

[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.

10.2. Informative References

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

[RFC5843] Bryan, A., "Additional Hash Algorithms for HTTP Instance Digests", RFC 5843, April 2010.

Appendix A. Acknowledgements and Contributors

Thanks to the Metalink community, Alexey Melnikov, Julian Reschke, Mark Nottingham, Daniel Stenberg, Matt Domsch, Micah Cowan, David Morris, Yves Lafon, Juergen Schoenwaelder, Ben Campbell, Lars Eggert, Sean Turner, Robert Sparks, and the HTTPBIS Working Group.

Thanks to Alan Ford and Mark Handley for spurring us on to publish this document.

This document is dedicated to Zimmy Bryan, Juanita Anthony, and Janie Burnett.

Authors' Addresses

Anthony Bryan
Pompano Beach, FL
USA

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

Neil McNab

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

Tatsuhiro Tsujikawa
Shiga
Japan

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

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/

Henrik Nordstrom

EMail: henrik@henriknordstrom.net
URI: http://www.henriknordstrom.net/