

Network Working Group	A. Bryan	
Internet-Draft	T. Kosse	
Intended status: Standards Track	D. Stenberg	
Expires: December 13, 2010	June 11, 2010	

[TOC](#)

FTP Extensions for Cryptographic Hashes

draft-bryan-ftp-hash-04

Abstract

The File Transfer Protocol does not offer any method to verify the integrity of a transferred file, nor can two files be compared against each other without actually transferring them first. Cryptographic hashes are a possible solution to this problem. In the past, several attempts have been made to add commands to obtain checksums and hashes, however none have been formally specified, leading to non-interoperability and confusion. To solve these issues, this document specifies a new FTP command to be used by clients to request cryptographic hashes of files.

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 December 13, 2010.

Copyright Notice

Copyright (c) 2010 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
1.1.	Example
2.	Notational Conventions
3.	The HASH Command (HASH)
3.1.	User-PI usage of HASH
3.2.	Changing the HASH algorithm
3.3.	FEAT Command Response for HASH Command
3.4.	HASH Command Errors
4.	IANA Considerations
5.	Implementation Requirements
6.	Security Considerations
7.	References
7.1.	Normative References
7.2.	Informative References
Appendix A.	Acknowledgements and Contributors
Appendix B.	List of Non-standard Cryptographic Hash or Checksum Commands and Implementations
Appendix C.	Document History
§	Authors' Addresses

1. Introduction

[TOC](#)

The File Transfer Protocol [\[RFC0959\] \(Postel, J. and J. Reynolds, "File Transfer Protocol," October 1985.\)](#) does not offer any method to verify the integrity of a transferred file, nor can two files be compared against each other without actually transferring them first.

Cryptographic hashes are a possible solution to this problem. In the past, several attempts have been made to add commands to obtain checksums and hashes, however none have been formally specified, leading to non-interoperability and confusion. To solve these issues, this document specifies a new FTP command to be used by clients to request cryptographic hashes of files. HTTP has a similar feature named Instance Digests [\[RFC3230\] \(Mogul, J. and A. Van Hoff, "Instance Digests in HTTP," January 2002.\)](#) which allows a client to request the cryptographic hash of a file.

[[Discussion of this draft should take place on ftptext@ietf.org (or apps-discuss@ietf.org if necessary).]]

1.1. Example

[TOC](#)

Example of HASH client request:

```
C> HASH filename.ext
```

HASH server response with Positive Completion code and the requested hash using the currently selected algorithm:

```
S> 213 80bc95fd391772fa61c91ed68567f0980bb45fd9
```

2. Notational Conventions

[TOC](#)

This specification describes conformance of FTP Extensions for cryptographic hashes.

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.

In examples, the "C>" lines are commands from user-PI to server-PI, and the "S>" lines are server-PI replies.

This document also uses notation defined in STD 9, [\[RFC0959\] \(Postel, J. and J. Reynolds, "File Transfer Protocol," October 1985.\)](#). In particular, the terms "reply", "user", "file", "pathname", "FTP commands", "user-PI", "server-FTP process", "server-PI", "mode", "type", and "ASCII", are all used here as defined there.

Syntax required is defined using the Augmented BNF defined in [\[RFC5234\] \(Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," January 2008.\)](#).

3. The HASH Command (HASH)

[TOC](#)

A new command "HASH" is added to the FTP command set to request the cryptographic hash of a file from a server-FTP process.

The syntax for the HASH command is:

```
hash-command = "HASH" SP <pathname>
```

As with all FTP commands, the "HASH" command word is case independent, and MAY be specified in any character case desired.

The HASH command keyword MUST be followed by a single space (ASCII 32) followed by the pathname.

The pathname argument should reference the same file as other file based commands such as STOR or RETR which the same argument would reference.

The text returned in response to the HASH command MUST be:

```
hash-response = "213" SP 1*HEXDIGIT CRLF
```

All hash values MUST be encoded in lowercase hexadecimal format.

The HASH command is meant to be used for files transmitted in Image type mode (TYPE I) and Stream transfer mode (MODE S). The returned hash MUST be calculated over the raw octet data of the file irrespective of the selected data type, transfer mode or any other state affecting the transfer. In other words, if a client were to download a full file using TYPE I and MODE S and were to calculate the hash on the received octet data, it would be identical to the hash returned by HASH.

3.1. User-PI usage of HASH

[TOC](#)

The user-PI issues the FEAT command to query the server-PI about which algorithm is currently selected. This also reveals the other algorithms that the server supports. In this example, the SHA-1 algorithm is currently selected.

```
C> FEAT
S> 211-Extensions supported:
S> ...
S> HASH SHA-256;SHA-512;SHA-1*;MD5
S> ...
S> 211 END
```

OPTS HASH is an alternative method for the user-PI to query the server-PI about which algorithm is currently selected.

```
C> OPTS HASH
S> 200 SHA-1
```

In this example, we wish to select SHA-256, a different algorithm.

```
C> OPTS HASH SHA-256
S> 200 SHA-256
```

The user-PI requests the cryptographic hash of a file with HASH command. Server-PI replies with cryptographic hash of file.

```
C> HASH filename.ext
S> 213 f0ad929cd259957e160ea442eb80986b5f01...
```

Client downloads file. Client hashes the downloaded file and compares its hash to the hash obtained from the server. The HASH command could also be used to verify that an uploaded file is an exact copy.

3.2. Changing the HASH algorithm

[TOC](#)

To query the current hash algorithm and to change it, the OPTS command as defined in [\[RFC2389\] \(Hethmon, P. and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol," August 1998.\)](#) is used with HASH as the first argument. If no second argument is passed, OPTS HASH simply returns the currently selected hash algorithm. To change the algorithm, a valid hashtype MUST be given as second argument. If the command is successful, all future calls to HASH until the next successful OPTS HASH command or until the session is reinitialized (REIN) will use the selected hash algorithm.

```
C> OPTS HASH
S> 200 SHA-1
C> OPTS HASH SHA-512
S> 200 SHA-512
C> OPTS HASH CRC-37
S> 501 Unknown algorithm, current selection not changed
```

```
hashopts-cmd = "OPTS HASH" [ SP hashtype ] CRLF
hashopts-response = "200" SP hashtype CRLF
```

3.3. FEAT Command Response for HASH Command

[TOC](#)

A server-FTP process that supports the HASH command MUST include, in the response to the FEAT command [\[RFC2389\] \(Hethmon, P. and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol," August 1998.\)](#), a feature line indicating that the HASH command is supported, along with a list of all supported hash algorithms in a semicolon separated list. The hash algorithm that is currently selected MUST be marked with an asterisk. The order of hash algorithms is insignificant. This command word is case insensitive, but it SHOULD be transmitted in upper case only. That is, the response SHOULD be:

```

C> FEAT
S> 211-Extensions supported:
S> ...
S> HASH SHA-256;SHA-512;SHA-1*;MD5
S> ...
S> 211 END

```

The ellipses indicate place holders where other features MAY be listed, but this is OPTIONAL. The single space indentation of each feature line is REQUIRED by [\[RFC2389\] \(Hethmon, P. and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol," August 1998.\)](#).

The IANA registry named "Hash Function Textual Names" defines values for hash types. Hash names should be presented in uppercase, but comparisons should be case-insensitive, e.g. MD5, md5, Md5 are all the same.

```

hash-feat = SP "HASH" SP hashlist CRLF
hashlist = 1*( hashname ["*"] ";" )
hashname = 1*( hchar )
hchar = ALPHA / DIGIT / "-" / "_" / "/" / "." / ","

```

3.4. HASH Command Errors

[TOC](#)

The server-PI SHOULD reply with a 500 reply if the HASH command is unrecognized or unimplemented.

The server-PI SHOULD reply with a 501 reply to the OPTS HASH command if the user-PI has requested an unknown or unsupported algorithm.

The server-PI SHOULD reply with a 550 reply if the HASH command is used on a file that can not be found.

The server-PI SHOULD reply with a 552 reply if the user is not allowed to use the HASH command.

The server-PI SHOULD reply with a 450 reply if the server is busy, e.g. already hashing other files yet inviting the client to retry in future.

4. IANA Considerations

[TOC](#)

This new command is added to the "FTP Commands and Extensions" registry created by [\[RFC5797\] \(Klensin, J. and A. Hoenes, "FTP Command and Extension Registry," March 2010.\)](#).

Command Name: HASH

Description: Cryptographic Hash of a file

FEAT String: HASH
Command Type: Service execution
Conformance Requirements: Optional
Reference: This specification

5. Implementation Requirements

[TOC](#)

All conforming implementations MUST at least support the SHA-1 algorithm [\[FIPS-180-3\] \(National Institute of Standards and Technology \(NIST\), "Secure Hash Standard \(SHS\)," October 2008.\)](#). Implementations SHOULD NOT make any algorithm the default that is known to be weaker than SHA-1. Support for any additional algorithms is OPTIONAL.

6. Security Considerations

[TOC](#)

Calculating a file's hash is a CPU intensive operation and can easily consume the available disk I/O resources. If the HASH command isn't implemented carefully, a server could be vulnerable to a denial of service attack. On an affected server a malicious user could, for example, continuously send HASH commands over multiple connections and thus consume most of the FTP server's CPU and disk I/O resources, leaving little room for other operations. To mitigate this risk, a server SHOULD cache the calculated hashes so that the hash of a file is only calculated once even if multiple hash requests are sent for that file.

The performance of commonly used hard disk drives is adversely affected by the amount of time the device needs to reposition its read-and-write heads. A server SHOULD therefore avoid hashing multiple files at the same time which are located on the same physical media and SHOULD instead hash them sequentially. The FTP server's right to refuse to calculate the hash is of course important to help against DOS risks. A possible solution is to use the 450 reply code of HASH to indicate that the server is already busy with another HASH operation.

In addition, the HASH command can be used to draw conclusions about the contents of a file. If the hash of a file on some server matches the hash of some known, local file, both files are likely identical. To prevent this scenario it suffices to limit use of the HASH command to users who would already be able to download the file.

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 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.

7. References

[TOC](#)

7.1. Normative References

[TOC](#)

[FIPS-180-3]	National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," FIPS PUB 180-3, October 2008.
[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.
[RFC2389]	Hethmon, P. and R. Elz, " Feature negotiation mechanism for the File Transfer Protocol ," RFC 2389, August 1998.
[RFC5234]	Crocker, D. and P. Overell, " Augmented BNF for Syntax Specifications: ABNF ," STD 68, RFC 5234, January 2008.

7.2. Informative References

[TOC](#)

[RFC3230]	Mogul, J. and A. Van Hoff, " Instance Digests in HTTP ," RFC 3230, January 2002.
[RFC5797]	Klensin, J. and A. Hoenes, " FTP Command and Extension Registry ," RFC 5797, March 2010.
[draft-twine-ftpmd5]	Twine, J. , " The MD5 and MMD5 FTP Command Extensions ," draft-twine-ftpmd5-00 (work in progress), May 2002.

Appendix A. Acknowledgements and Contributors

[TOC](#)

Thanks to John C. Klensin, Alfred Hoenes, James Twine, Robert McMurray, and Mathias Berchtold.

[TOC](#)

Appendix B. List of Non-standard Cryptographic Hash or Checksum Commands and Implementations

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

A number of similar checksum or hash commands exist, but are not formally specified, leading to non-interoperability and confusion. The commands, any specifications, and relevant details:

*CKSM: GridFTP v2 Protocol Description <http://www.ogf.org/documents/GFD.47.pdf> Usage: OPTS CKSM <algorithm> CRLF. Supports ADLER32, MD5, CRC32.

*MD5/MMD5: Expired Internet Draft [\[draft-twine-ftpmd5\] \(Twine, J., "The MD5 and MMD5 FTP Command Extensions," May 2002.\)](#) from 2002. Usage: MD5 <filepath> Algorithm specific command. Response codes: 251 positive completion, 500 Command Not Recognized, 502 Command Not Implemented, 504 Command Not Implemented for the Specified Argument.

*SITE CHECKSUM: Usage: SITE check_login SP CHECKSUM SP pathname CRLF. Supports CRC32 and MD5.

*SITE SHOHASH: Usage: site shohash [filename]. Supports MD5. Response codes: 200 positive completion.

*XCRC: By GlobalSCAPE in 2001. http://help.globalscape.com/help/secureserver2/File_Integrity_Checking.htm Usage: XCRC <filename> SP EP. SP is starting point and EP is ending point in bytes and are optional parameters. Algorithm specific command. Response codes: 250 positive completion, 450 Requested file action not taken. (File is busy), 550 Requested action not taken. (File not found, no read permission, SP or EP not correct).

*XMD5: XMD5 <filename> SP EP. Similar to XCRC. Algorithm specific command.

*XSHA, XSHA1, XSHA256, XSHA512: Usage similar to XCRC, although SP/EP usage unknown. Algorithm specific commands.

An incomplete list of FTP clients and servers that have implemented these commands:

*Akamai NetStorage (supports SITE CHKSH/SITE SHOHASH) p17-18 http://pigdogslow.dyndns.org/NetStorage_UserGuide.pdf

*Apache Ftp Server (supports MD5/MMD5 from draft-twine-ftpmd5) http://cwiki.apache.org/FTP_SERVER/documentation.html

*Backup4all Pro (supports XCRC)

*Backup to FTP (supports XCRC)

*BlackMoon FTP Server (supports XCRC) <http://www.blackmoonftpserver.com/portal/readmore/features.html>

*C.P.A. Secure (supports XCRC) <http://www.cpasecure.com/CPASecureVsSecureFTP.html>

*Cerberus FTP server (supports XCRC, XMD5, XSHA1, XSHA256, XSHA512) <http://www.softpedia.com/progChangelog/Cerberus-FTP-Server-Changelog-1904.html>

*Core FTP Pro (supports XCRC)

*Cross FTP Server (supports MD5/MMD5)

*FileCOPA FTP Server (supports XCRC, XMD5, XSHA1) <http://www.filecopa-ftpserver.com/features.html>

*File Watchdogs FTP Server (supports XCRC, XMD5, XSHA1, XSHA256, XSHA512) <http://www.filewatchdogs.com/ftpsitehosting/help/15559.htm>

*FireFTP (supports XMD5, XSHA1) <http://fireftp.mozdev.org/features.html>

*FTP Daemon (supports SITE CHECKMETHOD/SITE CHECKSUM) <http://www.pro-bono-publico.de/projects/ftpd.html>

*FTP Voyager (supports XCRC) <http://www.ftpvoyager.com/XCRC.asp>

*Gene6 FTP Server <http://www.g6ftpserver.com/en/information#features>

*GlobalSCAPE's Secure FTP Server / EFT Server / CuteFTP clients (supports XCRC)

*Globus FTP client / Globus Toolkit(supports CKSM) http://www.globus.org/toolkit/releasenotes/3.2.0/gridftp_notes.html

*GoldenGate FTP (Ftp Full Java Server) (supports XCRC, XMD5, XSHA1)

*IceWarp FTP Server http://www.icewarp.com/products/ftp_server/

*ICS FTP client (supports XCRC, XMD5) <http://www.magsys.co.uk/delphi/magics.asp>

*ioFTPD (supports XCRC)

*JAFS (supports XCRC and MD5) <http://www.sbbi.net/site/jafs/features.html>

*Kellerman FTP (supports XCRC) <http://sharptoolbox.com/tools/kellerman-ftp>

*Limagito FTP server (supports XCRC, XMD5, XSHA1) <http://www.limagito.com/file-mover-features.html>

*MOVEit DMZ (supports XSHA1)

*Nofeel FTP server (supports XCRC, XMD5, XSHA1) <http://www.nftpserver.com/history.php>

*Null FTP (supports XCRC, XMD5, XSHA) <http://www.sharewareconnection.com/null-ftp-client-pro.htm>

*Orenosv FTP Client (supports XCRC, XMD5) http://www.orenosv.com/orenosv/ftpcli_en.html

*ProFTPD module mod_digest (supports XCRC, XMD5, XSHA1, SHA256) http://www.smartftp.com/oss/proftpd/mod_digest.html

*PSFTPD Secure FTP Server (supports XCRC, XMD5, XSHA) http://www.psftp.de/psftpd_fo.php

*Quick 'n Easy FTP Server (supports XCRC) http://www.pablosoftwaresolutions.com/html/quick__n_easy_ftp_server_pro.html

*RaidenFTPD32 FTP server (supports XCRC, XMD5)

*Robo-FTP Server (supports XCRC, XMD5, XSHA1) http://kb.robo-ftp.com/change_log/show/61

*SyncBackPro and SyncBackSE (supports XCRC) <http://www.2brightsparks.com/syncback/sbpro-changes.html>

*Secure FTP Factory (supports XCRC)

*Serv-U FTP Server (supports XCRC) http://www.serv-u.com/help/serv_u_help/additional_ftp_commands_supported_by_serv_u.htm

*SmartFTP client (supports XCRC, XMD5, XSHA, CKSM) <http://www.smartftp.com/features/>

*Starksoft Ftp Component for .NET / Mono (supports XCRC, XMD5, XSHA1) http://www.starksoft.com/prod_ftp.html

*Titan FTP Server (supports XCRC)

*Turbo FTP (supports XCRC)

*WISE-FTP (supports XCRC) <http://www.wise-ftp.com/news/>

*WS_FTP client / server (supports XSHA1, server also XMD5, XSHA1, XSHA256, XSHA512) http://ipswitchft.custhelp.com/app/answers/detail/a_id/671/kw/xmd5/r_id/166/sno/1

*wuftpd (supports SITE CHECKMETHOD/SITE CHECKSUM)

*wzdFTPd (supports XCRC, XMD5) <http://www.wzdftpd.net/wiki/index.php/Commands>

*Zalman FTP Client (supports XCRC) <http://www.zalmansoftware.com/download.html>

*zFTPServer

Appendix C. Document History

[TOC](#)

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

Known issues concerning this draft:

*Should HASH support partial file hashes, similar to the Content-MD5 HTTP Header.

*Underspecification of the representation of the file that shall undergo the hash calculation.

*Should the server response include the algorithm? i.e. "S> 213 SHA-256 xxxxxxxxxxxxxxxx"

-04 : June 11, 2010.

*User-PI usage and command errors sections updated.

-03 : May 21, 2010.

*List of non-standard checksum and hash commands and their implementations.

-02 : April 16, 2010.

*Error codes section.

-01 : April 7, 2010.

*Changing HASH algorithm with OPTS.

*Reference RFC 5797 and add IANA Considerations section.

*Informative Reference to expired Internet Draft (draft-twine-ftpmd5) which attempted to address this issue (it only supported one hash, MD5).

-00 : October 19, 2009.

*Initial draft.

Authors' Addresses

[TOC](#)

	Anthony Bryan
	Pompano Beach, FL
	USA
Email:	anthonybryan@gmail.com
URI:	http://www.metalinker.org
	Tim Kosse
Email:	tim.kosse@filezilla-project.org
URI:	http://filezilla-project.org/
	Daniel Stenberg
Email:	daniel@haxx.se
URI:	http://www.haxx.se/