Network Working Group M. Peterson

Internet Draft Rhino Software, Inc.
Intended status: Proposed Standard November 30, 2010
Expires: May 2011

Streamlined FTP Command Extensions
draft-peterson-streamlined-ftp-command-extensions-10.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies FTP commands to download thumbnails of remote images, remove entire directory trees, request the amount of storage space available to the user, and to request the size of a remote directory and its contents. The commands are designed to reduce the number of server / client exchanges, provide information that was not

previously available, and to reduce bandwidth requirements for some higher level operations.

Conventions used in this document

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 [RFC2119].

This document uses previously established definitions for FTP related terms as defined in [RFC959]. In particular, the terms "data connection", "FTP command", "file", "pathname", "reply", "server-FTP process", "user", and "user-FTP process" are used as defined. The terms "server" and "client" are also used in place of "server-FTP process" and "user-FTP process", respectively, for the sake of brevity.

The syntax for responses to FTP commands used in this document is explained using Augmented BNF as defined in [RFC5234]. Additional tokens defined in [RFC3659] that extend ABNF are also used, specifically the TCHAR token.

In command examples, "C>" and "S>" indicate lines sent by the client and server, respectively.

Table of Contents

1. Introduction

This document extends the File Transfer Protocol [RFC959]. It adds five new commands: "RMDA", "DSIZ", "AVBL", "THMB", and "CSID". These commands have been designed to streamline certain client and server communications, reduce the amount of bandwidth required to perform various operations, and identify specific information about the client and server.

2. Remove Directory All (RMDA)

The FTP command REMOVE DIRECTORY ALL (RMDA) removes a directory from the server and all of its contents including all files and subdirectories. The RMDA command is considered analogous to recursively deleting all files and directories contained in a given remote directory (including the directory itself) one at a time.

The primary advantage to using this command is the elimination of the additional commands usually required to perform the equivalent action. For directories containing a large number of files and subdirectories, using RMDA eliminates the overhead of querying for subdirectory listings. The end result is a more responsive operation for both the client and server.

Depending on the content being deleted (files and subdirectories) this can be a very lengthy operation for the server. Servers that

implement this command are encouraged to do so in a way that does not block operations for other client connections.

2.1. RMDA Syntax

The syntax of the RMDA command is:

```
rmda           = "RMDA" SP pathname CRLF
```

All FTP commands, including RMDA, are case insensitive. However, the <pathname> argument provided with the command may be case sensitive as dictated by the server's operating system.

The <pathname> specifies a remote directory that should be deleted along with all of its contents.

```
rmda-response  = "250" SP *TCHAR CRLF /
                 error-response
```

Responses should be the same as those for the RMD command. If processing fails at any point during the operation, the specified pathname is not to be removed. Any cached information about the pathname is considered invalid when RMDA returns an error.

2.2. RMDA Error Responses

The RMDA command can return any error response listed for the RMD command in [RFC959]. The most common of these responses occurs in the event of an error while deleting a file or subdirectory, in which case the server should return a 550 reply. A 550 reply should also be sent if the provided path is a file instead of a directory.

In the event that a nested file or subdirectory cannot be deleted for any reason, the parent directories of that file or subdirectory should not be deleted. When the server encounters this situation, it should attempt to delete all other files and subdirectories before stopping or responding to the client. Once the server has attempted to delete all files and subdirectories, the server MUST report failure to the client if even a single file or subdirectory could not be deleted.

2.3. RMDA FEAT Response

When replying to the FEAT command, a server-FTP process that supports the RMDA command MUST include a line starting with the word "RMDA". If descriptive text is optionally desired, a space MUST immediately follow the "RMDA" word.

```
C> FEAT
S> 211- <any descriptive text>
S>  ...
S>  RMDA <any descriptive text>
S>  ...
S> 211 End
```

The ellipses indicate placeholders where other features may be included, and are not required. The one space indentation of the feature lines is mandatory per [RFC2389].

2.4. RMDA Examples

This example assumes that the server contains a directory named "A" and that the client has the necessary access to delete "A" and its contents. The client-server exchange for using RMDA to accomplish this would be:

```
C> RMDA A
S> 250 RMDA command successful.
```

If deletion of the directory, any contained file, or subdirectory fails the client-server exchange would be:

```
C> RMDA A
S> 550 A: Cannot delete directory.
```

If deletion fails due to inadequate permissions for the directory, any contained file, or subdirectory for the connected client, the client-server exchange would be:

```
C> RMDA A
S> 550 A: Permission denied.
```

3. Directory Size (DSIZ)

The FTP command DIRECTORY SIZE (DSIZ) returns the number of octets (8-bit bytes) on the remote file system occupied by a given directory and its contents. The argument to this command MUST represent a directory path, not a file path.

While a method already exists to retrieve similar information by recursively issuing the SIZE command as defined in [RFC3659] and tallying the results, the SIZE command returns the number of transfer octets for a specified file, which means that it considers the current data representation type in its calculated response. The DSIZ command should not consider this in its response, instead

leaving it up to the underlying file system to determine the number of octets occupied by the files.

In addition, issuing a single DSIZ command saves on the number of required commands and the overhead associated with recursively querying for the SIZE of each individual file. This results in a more responsive and conservative communication exchange between the client and server.

Depending on the size of the path being queried and its location relative to the server, this can be a very lengthy operation. Servers that implement this command are encouraged to do so in a way that does not block operations for other client connections.

3.1. DSIZ Syntax

The syntax of the DSIZ command is:

```
dsiz           = "DSIZ" [SP pathname] CRLF
```

All FTP commands, including DSIZ, are case insensitive. However, the <pathname> argument provided with the command may be case sensitive as dictated by the server's operating system. If the <pathname> argument is not present in the received DSIZ command, its value is assumed to be the current directory (i.e., the path that would be returned in response to a Print Working Directory (PWD) command).

The <pathname> specifies a remote directory that should be queried to retrieve the total storage space occupied by its contents on the server's file system. The <pathname> MUST be a valid directory path; it cannot be a path to a file or a system device.

```
dsiz-response  = "213" SP 1*DIGIT CRLF /
                 error-response
```

The 213 reply is formatted in such a way that the returned size value can be machine parsed; the response MUST simply be the command response followed by an integer value with no delimiters.

A successful response includes a numeric value indicating the number of octets on the server's file system occupied by the provided pathname and its contents.

3.2. DSIZ Error Responses

In the event that the pathname is not a directory, the server MUST return a permanent 550 error reply. Where the command cannot be

parsed, a 500 or 501 reply SHOULD be sent. The client MUST NOT assume that the presence of a 550 reply indicates that it cannot access the directory or its contents. The server may generate this error for other reasons, for example the overhead required for the operation is too great.

3.3. DSIZ FEAT Response

When replying to the FEAT command, a server-FTP process that supports the DSIZ command MUST include a line starting with the word "DSIZ". If descriptive text is optionally desired, a space MUST immediately follow the "DSIZ" word.

```
C> FEAT
S> 211- <any descriptive text>
S>  ...
S>  DSIZ <any descriptive text>
S>  ...
S> 211 End
```

The ellipses indicate placeholders where other features may be included, and are not required. The one space indentation of the feature lines is mandatory per [RFC2389].

3.4. DSIZ Examples

Assuming the presence of directory "A" on the server and that the client has the necessary permissions to access and list it, a typical DSIZ client-server exchange would be:

```
C> DSIZ A
S> 213 123456
```

Where 123456 is the number of octets on the server's file system occupied by the directory "A" and its contents.

Assuming directory "A" cannot be listed due to client permissions, the client-server exchange would be:

```
C> DSIZ A
S> 550 A: Permission denied.
```

Assuming the client provides the argument "file.fid" where "file.fid" is an existing file, the client-server exchange would be:

```
C> DSIZ file.fid
S> 550 file.fid: Is a file.
```

4. Available Octets (AVBL)

The FTP command AVAILABLE OCTETS (AVBL) can be used to retrieve the number of octets available to receive uploads in a specified remote directory for the logged in user. Many server-FTP processes have implemented proprietary methods for restricting the amount of space available, either to the user as a whole or within a specified remote location. The AVBL command MUST take restrictions into account by returning the smaller of the physical space available or any space restriction. The AVBL command offers an interface for retrieving this value in a way that can be machine parsed.

4.1. AVBL Syntax

The syntax of the AVBL command is:

```
avbl          = "AVBL" [SP pathname] CRLF
```

All FTP commands, including AVBL, are case insensitive. However, the optional <pathname> argument provided with the command may be case sensitive as dictated by the server's operating system. If the <pathname> argument is not present in the received AVBL command, its value is assumed to be the current directory (i.e., the path that would be returned in response to a Print Working Directory (PWD) command).

The <pathname> argument specifies the remote directory that the client wants to know how many octets are available to the user for uploads. The <pathname> MUST be a valid directory path; it cannot be a path to a file or a system device.

```
avbl-response  = "213" SP 1*DIGIT CRLF /
                 error-response
```

The 213 reply is formatted in such a way that the returned size can be machine parsed.

A successful response includes a numeric value that may be the number of octets available as dictated by the physical limitations of the underlying storage mechanism. It may also be a limit imposed upon <pathname> or even the user that issued the command. The exact method for calculating this numeric value is up to the specific server-FTP process implementation.

4.2. AVBL Error Responses

In the event that the pathname is not a directory, the server MUST return a permanent 550 error reply. Where the command cannot be parsed, a 500 or 501 reply should be sent. The client MUST NOT assume that the presence of a 550 reply indicates that it cannot access the directory or its contents. The server may generate this error for other reasons, for example the overhead required for the operation is too great.

4.3. AVBL FEAT Response

When replying to the FEAT command, a server-FTP process that supports the AVBL command MUST include a line starting with the word "AVBL". If descriptive text is optionally desired, a space MUST immediately follow the "AVBL" word.

```
C> FEAT
S> 211- <any descriptive text>
S>  ...
S>  AVBL <any descriptive text>
S>  ...
S> 211 End
```

The ellipses indicate placeholders where other features may be included, and are not required. The one space indentation of the feature lines is mandatory per [RFC2389].

4.4. AVBL Examples

A typical client-server exchange using the AVBL command may look like this:

```
C> AVBL
S> 213 123456
```

Where 123456 is the number of octets the user has available for file uploads. In this example, no pathname is provided with the AVBL command. Therefore, the server has used the current working directory when processing its response.

Assuming the presence of directory "A" on the server and that the client has the necessary permissions to access and list it:

```
C> AVBL A
S> 213 123456
```

Assuming directory "A" cannot be listed due to client permissions, the client-server exchange would be:

```
C> AVBL A
S> 550 A: Permission denied.
```

Assuming the client provides the argument "file.fid" where "file.fid" is an existing file, the client-server exchange would be:

```
C> AVBL file.fid
S> 550 file.fid: Is a file.
```

5. Retrieve Thumbnail of Remote Image File (THMB)

Many user-FTP processes have implemented a view of the remote directory contents that allows for a size-reduced representation of remote files recognized as image types to be displayed. This size-reduced representation of the image is commonly referred to as a thumbnail. This view of the remote file system usually requires a large amount of resources, in the form of time and bandwidth, in order to be properly represented.

The THMB FTP command allows the server-FTP process to locally generate a thumbnail of a requested image type and transfer the resulting image instead of the entire contents of the original image file. This facilitates a faster and more efficient method of fulfilling a user-FTP process's implementation of a "thumbnail view".

Depending on the size of the image file being processed and the CPU resources available to the server-FTP process, this can be a lengthy operation. Servers that implement this command are encouraged to do so in a way that does not block operations for other client connections.

5.1. THMB Syntax

The THMB FTP command initiates a file transfer of the generated thumbnail image in much the same way that the Retrieve (RETR) FTP command initiates a download of a remote file. As such, thumbnail transfer occurs over a previously established data connection as indicated in [RFC959]. When the THMB command is received, an initial response is sent over the command connection, the thumbnail is transferred over the data connection, the data connection is closed, and the final response is sent over the command connection.

The syntax of the THMB command is:

```
thmb     = "THMB" SP fmt SP 1*DIGIT SP 1*DIGIT SP pathname CRLF
```

All FTP commands, including THMB, are case insensitive. However, the <pathname> argument provided with the command may be case sensitive as dictated by the server's operating system.

The <fmt> argument indicates an image format recognized by the server-FTP process that should be used to generate the thumbnail image. Its value corresponds to the commonly accepted file extension used for the requested image format. Image formats supported by the server are listed in the THMB FEAT Response [6.3]. The requested thumbnail format does not necessarily conform to the existing format of the image. For example, a client-FTP process may request a PNG thumbnail of a JPEG image.

The two numbers following <fmt> are the requested maximum width and maximum height in pixels, respectively, that the client is willing to accept for the generated thumbnail image. The server SHOULD preserve the original aspect ratio of the image to avoid distortion while ensuring these maximum values are not exceeded by the generated thumbnail image.

The <pathname> specifies a remote file, of a supported image type, that should be sent to the client in thumbnail form according to the other arguments.

The syntax of a positive response is:

```
thmb-response  = "150" SP *TCHAR "(" 1*DIGIT 1*TCHAR ")" CRLF /
                 error-response
```

'1*DIGIT' represents the size in octets of the forthcoming file transfer. The value within the parentheses MAY include descriptive text, for example (123456 Bytes) where "Bytes" is descriptive text. By providing this information to the client, the client can determine the required file size and transfer status information.

The potential responses to the THMB command that should be expected over the command channel correspond to the list of acceptable responses to the RETR command as defined in [RFC959].

As the THMB command initiates a data transfer over a previously negotiated data connection, the binary image data for the server generated thumbnail image is transferred over this data connection. Because image data is always binary in nature, the server-FTP process

will always send this data as if the current data representation type is BINARY, regardless of the setting that was established through the last received TYPE command. After the THMB command is fully processed, the server-FTP process reverts back to using the previously established data representation type.

5.2. THMB Error Responses

If an error is encountered while attempting to process or generate the thumbnail image, the server MUST use the 550 reply code. If an error is encountered while allocating resources for temporary storage of the generated thumbnail, the server MUST use the 550 reply code.

5.3. THMB FEAT Response

When replying to the FEAT command, a server-FTP process that supports the THMB command MUST include a line starting with the word "THMB". If descriptive text is optionally desired, a space MUST immediately follow the required <fmt> argument.

```
C> FEAT
S> 211- <any descriptive text>
S>  ...
S>  THMB <fmt> <any descriptive text>
S>  ...
S> 211 End
```

The <fmt> argument represents the supported image format(s); multiple format abbreviations can be given separated by the "|" character. Each format abbreviation must be the media subtype name of an 'image' media type that is IANA-registered under the provisions established in [RFC4288] describing a unique file storage format and giving a specific file name extension. Supporting servers must support at least JPEG or PNG formats. An example THMB response to the FEAT command might be:

```
THMB JPEG|GIF|TIFF|PNG
```

This response identifies JPEG, GIF, TIFF, and PNG formats as image types supported by the server for thumbnail generation.

The ellipses indicate placeholders where other features may be included, and are not required. The one space indentation of the feature lines is mandatory per [RFC2389].

5.4. THMB Examples

Assuming the existence of image file "widget.png" on the server and that the client has all necessary permissions to access and transfer the "widget.png" file, a request for a "widget.png" thumbnail might look like this:

```
C> THMB PNG 80 80 widget.png
S> 150 Starting thumbnail transfer (1234 Bytes) for widget.png
S> 226 Transfer complete.
```

In between the 150 and 226 server responses, the thumbnail image has been transferred over a negotiated data connection.

If the client requests a thumbnail for an unsupported image type, the exchange would be:

```
C> THMB PSP 80 80 widget.psp
S> 501 Syntax error in parameters or arguments.
```

6. Client / Server Identification (CSID)

The FTP command CLIENT / SERVER IDENTIFICATION (CSID) allows the client and server to exchange relevant identifying information about the programs and underlying platforms being used for the FTP communication.

While the informally documented command CLNT already exists for the user-FTP process to identify itself to the server-FTP process, the CSID command also allows the server to identify itself to the client, much like the "Server" header field of the HTTP protocol as defined in [RFC2616] and the "Protocol Version Exchange" [RFC4253] in SSH.

The CSID command MUST be available only while the client-FTP process is logged in.

6.1. CSID Syntax

The syntax of the CSID command is:

```
csid           = "CSID" SP cfact *([SP] cfact) CRLF
cfact          = cfactname "=" value ";"
cfactname      = "Version" / "Name" / "Vendor" / ext_cfactname
ext_cfactname  = token  ; for extendability
```

All FTP commands, including CSID, are case insensitive.

The "Name" and "Version" arguments are required and are case insensitive. The client must specify the argument name followed by the equal sign, then the value followed by a semicolon. The semicolon following the argument value is required even after the last argument specified. The space between arguments is optional. The order in which the arguments are specified is not relevant, for example, "Version" may appear before "Name".

Both values may be any string that the server can use to identify the client-FTP process. It is recommended that for commercial FTP clients, the "Name" and "Version" values should reflect those of the software in use by the user.

"Vendor" is an optional argument identifying the vendor of the client software. It may be any string that the server can use to identify the client-FTP process.

```
clsid-response    = control-response / error-response
control-response  = "200" [SP *(facts)] CRLF
facts             = 1*(fact ";" [SP])
fact              = factname "=" value
factname          = "Name" / "Version" / "Vendor" /
                    "CaseSensitive" / "OS" / "OSVer" /
                    os-dependant-fact / local-fact
os-depend-fact    = <IANA assigned OS name> "." token
local-fact        = "X." token
value             = *SCHAR
```

The server response is similar to the client arguments, however more information is provided. The following are the facts.

"Name" is an optional argument identifying the name of the server. For commercial FTP servers, this should be the brand name of the server.

"Version" is an optional argument identifying the version number of the server. This may be any string that can be used by the client to identify the server.

"Vendor" is an optional argument identifying the vendor of the server software. It may be any string that the client can use to identify the server.

"OS" is an optional argument identifying the operating system name, i.e., "Windows XP", "BSD UNIX", etc.

"OSVer" is an optional argument identifying the operating system version number.

"CaseSensitive" is a required argument identifying whether or not the underlying file system is letter case sensitive. 0 identifies that the server IS NOT letter case sensitive; 1 identifies that the server IS letter case sensitive.

6.2. CSID Error Responses

If the client-FTP process is not logged in, the server-FTP process MUST return a 530 response code.

If the client-FTP process is logged in, the server-FTP processes that implements the CSID command should return no errors in response to the command. If the client sends unrecognized arguments, they MUST be ignored. If the "Name", "Version", or "Vendor" arguments are not present in the command, they should be assumed to be blank.

6.3. CSID FEAT Response

When replying to the FEAT command, a server-FTP process that supports the CSID command MUST include a line starting with the word "CSID". If descriptive text is optionally desired, a space MUST immediately follow the "CSID" word.

```
C> FEAT
S> 211- <any descriptive text>
S>  ...
S>  CSID <any descriptive text>
S>  ...
S> 211 End
```

The ellipses indicate placeholders where other features may be included, and are not required. The one space indentation of the feature lines is mandatory per [RFC2389].

6.4. CSID Examples

At any time during a logged-in server-FTP process, the client-server exchange would be:

```
C> CSID Name=Client; Version=1.0.0.1;
S> 200 Name=Server; Version=1.0; CaseSensitive=0;
```

In the example above, the server is responding only with its name, version number, and the one required field, CaseSensitive.

A server whose administrator wants to limit the amount of information, yet still support CSID could have a client-server exchange where only the required response values are returned:

```
C> CSID Name=Client; Version=1.0.0.1;
S> 200 CaseSensitive=0;
```

A client-server exchange where all CSID values are returned would be:

```
C> CSID Name=Client; Version=1.0.0.1;
S> 200 Name=Server; Version=1.0; Vendor=Acme Software,
   Inc.; OS=UNIX; OSVer=1.0; CaseSensitive=0;
```

A client-server exchange where the client-FTP process is not logged in would be:

```
C> CSID Name=Client; Version=1.0.0.1;
S> 530 Not logged in.
```

7. Security Considerations

This document does not explicitly address security issues as they pertain to the FTP protocol in general. The commands introduced in this document do not affect the security of the FTP protocol nor do they address any of the security considerations described in [RFC2577].

Implementation of the RMDA, DSIZ, AVBL, and THMB commands can require significant server resources to process. This knowledge could be used by attackers in a denial of service attack. However, this issue has been addressed before with others commands, such as SIZE, and is traditionally mitigated by the server-FTP process's implementation of the command.

The FTP server is encouraged to take specific care when implementing the RMDA command to avoid bypassing permissions. While it may be tempting to take advantage of a method made available by the platform to delete an entire tree of the file system at once, doing so could allow for the possibility of bypassing permissions that apply to specific files. Consequently, the contents of the path being deleted by RMDA must be individually evaluated.

Some FTP server administrators may determine that the server response to the CSID command contains information relevant in a probe to evaluate the security of the system. It's worth noting that the information contained in this response is no different than that which is revealed in the HTTP response header fields for an HTTP

server. However, in consideration of this potential concern, the CSID response should be configurable by the server administrator to exclude optional information as desired.

8. IANA Considerations

These new commands are added to the "FTP Commands and Extensions" registry created by [RFC5797].

8.1. RMDA

Description: Remove Directory All

FEAT String: RMDA

Command Type: Service extension

Conformance Requirements: Optional

Requirements: This specification

8.2. DSIZ

Description: Directory Size

FEAT String: DSIZ

Command Type: Service extension

Conformance Requirements: Optional

Requirements: This specification

8.3. AVBL

Description: Available Octets

FEAT String: AVBL

Command Type: Service extension

Conformance Requirements: Optional

Requirements: This specification

8.4. THMB

Description: Retrieve Thumbnail of Remote Image File

FEAT String: THMB <fmt> <any descriptive text>

Command Type: Service extension

Conformance Requirements: Optional

Requirements: This specification

9. Conclusions

Inclusion of these commands both for servers and client can significantly reduce bandwidth requirements for many operations as well as improving the client-side experience by reducing the amount of time needed to perform these operations via other methods.

10. Acknowledgments

I would like to thank Alfred Hoenes, John C. Klensin, and Anthony Bryan for reviewing this document and providing thorough and thoughtful suggestions on how to make it better.

I would like to thank Mathias Berchtold for his input and ideas regarding the CSID (Client / Server Identification) command.

I would like to thank Joe Touch for his courteous personal assistance with the 2-Word-v2.0.template.dot template used to prepare this document.

I would like to thank Douglas J. Papenthien for his input regarding grammar, formatting, and technical ideas regarding each of the commands.

This document was prepared using 2-Word-v2.0.template.dot.

11. References

11.1. Normative References

[RFC959] Postel, J. and J. Reynolds, "File Transfer Protocol (FTP)", STD 9, RFC 959, 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.

[RFC2577] Allman, M. and S. Ostermann, "FTP Security Considerations", RFC 2577, May 1999.

[RFC3659] Hethmon, P., "Extensions to FTP", RFC 3659, March 2007.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

11.2. Informative References

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

[RFC4253] Ylonen, T., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC5797] Klensin, J. and Hoenes, A., "FTP Command and Extension Registry", RFC 5797, March 2010.

Author's Addresses

Mark P. Peterson
Rhino Software, Inc.
P.O. Box 53
Helenville, WI 53137 U.S.A.

Phone: +1 (262) 560-9627
FAX: +1 (262) 560-9628
Email: mark.peterson@rhinosoft.com