Network Working Group
Request for Comments: 2389
See Also: 959
Category: Standards Track

P. Hethmon
Hethmon Brothers
R. Elz
University of Melbourne

August 1998

Feature negotiation mechanism for the File Transfer Protocol

Status of this Memo

Abstract

The File Transfer Protocol is, from time to time, extended with new commands, or facilities. Implementations of the FTP protocol cannot be assumed to all immediately implement all newly defined mechanisms. This document provides a mechanism by which clients of the FTP protocol can discover which new features are supported by a particular FTP server.

Table of Contents

Abstract 1
1 Introduction 2
2 Document Conventions 2
2.1 Basic Tokens 3
2.2 Server Replies 3
3 Knowledge of Extra Capabilities - the FEAT Command 3
3.1 Feature (FEAT) Command Syntax 4
3.2 FEAT Command Responses 4
3.3 Rationale for FEAT 6
4 The OPTS Command 6
5 Security Considerations 7
6 References 8
Acknowledgements 8
Editors' Addresses 8
Full Copyright Statement 9

1. Introduction

This document amends the File Transfer Protocol (FTP) [1]. Two new commands are added: "FEAT" and "OPTS".

These commands allow a client to discover which optional commands a server supports, and how they are supported, and to select among various options that any FTP command may support.

2. Document Conventions

This document makes use of the document conventions defined in BCP14 [2]. That provides the interpretation of some capitalized words like MUST, SHOULD, etc.

Terms defined in [1] will be used here as defined there. These include ASCII, reply, server-FTP process, user-FTP process, server-PI, user-PI, and user.

Syntax required is defined using the Augmented BNF defined in [3]. Some general ABNF definitions are required throughout the document, those will be defined here. At first reading, it may be wise to simply recall that these definitions exist here, and skip to the next section.

2.1. Basic Tokens

This document imports the definitions given in Appendix A of [3]. There definitions will be found for basic ABNF elements like ALPHA, DIGIT, VCHAR, SP, etc. To that, the following terms are added for use in this document.

```
TCHAR          = VCHAR / SP / HTAB    ; visible plus white space
```

The TCHAR type, and VCHAR from [3], give basic character types from varying sub-sets of the ASCII character set for use in various commands and responses.

```
error-response = error-code SP *TCHAR CRLF
error-code     = ("4" / "5") 2DIGIT
```

Note that in ABNF, strings literals are case insensitive. That convention is preserved in this document. However note that ALPHA, in particular, is case sensitive, as are VCHAR and TCHAR.

2.2. Server Replies

Section 4.2 of [1] defines the format and meaning of replies by the server-PI to FTP commands from the user-PI. Those reply conventions are used here without change. Implementors should note that the ABNF syntax (which was not used in [1]) in this document, and other FTP related documents, sometimes shows replies using the one line format. Unless otherwise explicitly stated, that is not intended to imply that multi-line responses are not permitted. Implementors should assume that, unless stated to the contrary, any reply to any FTP command (including QUIT) may be of the multiline format described in [1].

Throughout this document, replies will be identified by the three digit code that is their first element. Thus the term "500 Reply" means a reply from the server-PI using the three digit code "500".

3. Knowledge of Extra Capabilities - the FEAT Command

It is not to be expected that all servers will necessarily support all of the new commands defined in all future amendments to the FTP protocol. In order to permit clients to determine which new commands are supported by a particular server, without trying each possible command, one new command is added to the FTP command repertoire. This command requests the server to list all extension commands, or extended mechanisms, that it supports. That is, all defined and specified commands and features not defined in [1], or this document, must be included in the FEAT command output in the form specified in

the document that defines the extension.

User-FTP PIs must expect to see, in FEAT command responses, unknown features listed. This is not an error, and simply indicates that the server-FTP implementor has seen, and implemented, the specification of a new feature that is unknown to the user-FTP.

3.1. Feature (FEAT) Command Syntax

```
feat            = "Feat" CRLF
```

The FEAT command consists solely of the word "FEAT". It has no parameters or arguments.

3.2. FEAT Command Responses

Where a server-FTP process does not support the FEAT command, it will respond to the FEAT command with a 500 or 502 reply. This is simply the normal "unrecognized command" reply that any unknown command would elicit. Errors in the command syntax, such as giving parameters, will result in a 501 reply.

Server-FTP processes that recognize the FEAT command, but implement no extended features, and therefore have nothing to report, SHOULD respond with the "no-features" 211 reply. However, as this case is practically indistinguishable from a server-FTP that does not recognize the FEAT command, a 500 or 502 reply MAY also be used. The "no-features" reply MUST NOT use the multi-line response format, exactly one response line is required and permitted.

Replies to the FEAT command MUST comply with the following syntax. Text on the first line of the reply is free form, and not interpreted, and has no practical use, as this text is not expected to be revealed to end users. The syntax of other reply lines is precisely defined, and if present, MUST be exactly as specified.

```
feat-response   = error-response / no-features / feature-listing
no-features     = "211" SP *TCHAR CRLF
feature-listing = "211-" *TCHAR CRLF
                  1*( SP feature CRLF )
                  "211 End" CRLF
feature         = feature-label [ SP feature-parms ]
feature-label   = 1*VCHAR
feature-parms   = 1*TCHAR
```

Note that each feature line in the feature-listing begins with a single space. That space is not optional, nor does it indicate general white space. This space guarantees that the feature line can

never be misinterpreted as the end of the feature-listing, but is required even where there is no possibility of ambiguity.

Each extension supported must be listed on a separate line to facilitate the possible inclusion of parameters supported by each extension command. The feature-label to be used in the response to the FEAT command will be specified as each new feature is added to the FTP command set. Often it will be the name of a new command added, however this is not required. In fact it is not required that a new feature actually add a new command. Any parameters included are to be specified with the definition of the command concerned. That specification shall also specify how any parameters present are to be interpreted.

The feature-label and feature-parms are nominally case sensitive, however the definitions of specific labels and parameters specify the precise interpretation, and it is to be expected that those definitions will usually specify the label and parameters in a case independent manner. Where this is done, implementations are recommended to use upper case letters when transmitting the feature response.

The FEAT command itself is not included in the list of features supported, support for the FEAT command is indicated by return of a reply other than a 500 or 502 reply.

A typical example reply to the FEAT command might be a multiline reply of the form:

```
C> feat
S> 211-Extensions supported:
S>  MLST size*;create;modify*;perm;media-type
S>  SIZE
S>  COMPRESSION
S>  MDTM
S> 211 END
```

The particular extensions shown here are simply examples of what may be defined in other places, no particular meaning should be attributed to them. Recall also, that the feature names returned are not command names, as such, but simply indications that the server possesses some attribute or other.

The order in which the features are returned is of no importance, server-FTP processes are not required to implement any particular order, or even to consistently return the same order when the command is repeated.

FTP implementations which support FEAT MUST include in the response to the FEAT command all properly documented FTP extensions beyond those commands and mechanisms described in RFC959 [1], including any which existed before the existence of FEAT. That is, when a client receives a FEAT response from an FTP server, it can assume that the only extensions the server supports are those that are listed in the FEAT response.

User-FTP processes should, however, be aware that there have been several FTP extensions developed, and in widespread use, prior to the adoption of this document and the FEAT command. The effect of this is that an error response to the FEAT command does not necessarily imply that those extensions are not supported by the server-FTP process. User-PIs should test for such extensions individually if an error response has been received to the FEAT command.

3.3. Rationale for FEAT

While not absolutely necessary, a standard mechanism for the server-PI to inform the user-PI of any features and extensions supported will help reduce unnecessary traffic between the user-PI and server-PI as more extensions may be introduced in the future. If no mechanism existed for this, a user-FTP process would have to try each extension in turn resulting in a series of exchanges between the user-PI and server-PI. Apart from being possibly wasteful, this procedure may not always be possible, as issuing of a command just to determine if it is supported or not may have some effect that is not desired.

4. The OPTS Command

The OPTS (options) command allows a user-PI to specify the desired behavior of a server-FTP process when another FTP command (the target command) is later issued. The exact behavior, and syntax, will vary with the target command indicated, and will be specified with the definition of that command. Where no OPTS behavior is defined for a particular command there are no options available for that command.

Request Syntax:

```
     opts             = opts-cmd SP command-name
                              [ SP command-options ] CRLF
     opts-cmd         = "opts"
     command-name     = <any FTP command which allows option setting>
     command-options  = <format specified by individual FTP command>
```

Response Syntax:

```
     opts-response    = opts-good / opts-bad
     opts-good        = "200" SP response-message CRLF
     opts-bad         = "451" SP response-message CRLF /
                        "501" SP response-message CRLF
     response-message = *TCHAR
```

An "opts-good" response (200 reply) MUST be sent when the command-name specified in the OPTS command is recognized, and the command-options, if any, are recognized, and appropriate. An "opts-bad" response is sent in other cases. A 501 reply is appropriate for any permanent error. That is, for any case where simply repeating the command at some later time, without other changes of state, will also be an error. A 451 reply should be sent where some temporary condition at the server, not related to the state of communications between user and server, prevents the command being accepted when issued, but where if repeated at some later time, a changed environment for the server-FTP process may permit the command to succeed. If the OPTS command itself is not recognized, a 500 or 502 reply will, of course, result.

The OPTS command MUST be implemented whenever the FEAT command is implemented. Because of that, there is no indication in the list of features returned by FEAT to indicate that the OPTS command itself is supported. Neither the FEAT command, nor the OPTS command, have any optional functionality, thus there are no "OPTS FEAT" or "OPTS OPTS" commands.

5. Security Considerations

No significant new security issues, not already present in the FTP protocol, are believed to have been created by this extension. However, this extension does provide a mechanism by which users can determine the capabilities of an FTP server, and from which additional information may be able to be deduced. While the same basic information could be obtained by probing the server for the various commands, if the FEAT command were not provided, that method may reveal an attacker by logging the attempts to access various extension commands. This possibility is not considered a serious enough threat to be worthy of any remedial action.

The security of any additional features that might be reported by the FEAT command, and manipulated by the OPTS command, should be addressed where those features are defined.

6. References

[1] Postel, J. and J. Reynolds, "File Transfer Protocol (FTP)", STD 9, RFC 959, October 1985.

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

[3] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

Acknowledgements

This protocol extension was developed in the FTPEXT Working Group of the IETF, and the members of that group are all acknowledged as its creators.

Editors' Addresses

Paul Hethmon
Hethmon Brothers
2305 Chukar Road
Knoxville, TN 37923 USA

Phone: +1 423 690 8990
Email: phethmon@hethmon.com

Robert Elz
University of Melbourne
Department of Computer Science
Parkville, Vic 3052
Australia

Email: kre@munnari.OZ.AU

Full Copyright Statement