FTP Extensions for Variable Protocol Specification*

Mark Allman
NASA Glenn Research Center/BBN Technologies
`mallman@grc.nasa.gov`

Shawn Ostermann
Ohio University School of Electrical Engineering and Computer Science
`ostermann@cs.ohiou.edu`

NASA Glenn Research Center CR-209414

Abstract

The specification for the File Transfer Protocol (FTP) assumes that the underlying network protocols use a 32-bit network address and a 16-bit transport address (specifically IP version 4 and TCP). With the deployment of version 6 of the Internet Protocol, network addresses will no longer be 32-bits. This paper specifies extensions to FTP that will allow the protocol to work over a variety of network and transport protocols.

1 Introduction

The keywords, such as MUST and SHOULD, found in this document are used as defined in RFC 2119 [Bra97].

The File Transfer Protocol (FTP) defined in RFC 959 [PR85] only provides the ability to open data connections on networks using the TCP/IPv4 protocol suite [Com95]; it assumes network addresses will be 32 bits in length and the transport address (TCP port number) will be 16 bits long. Changes to FTP to support different network and transport protocols will be needed as new protocols are deployed (most notably, IPv6 [DH96]). RFC 1639 [Pis94] specifies extensions to FTP to enable its use over various network protocols. While RFC 1639 allows for variable length transport addresses, it provides no mechanism to choose which transport protocol will be used. The transport protocol is assumed to be based on the network protocol chosen. This document provides a specification that makes no assumptions regarding the underlying network and transport protocols. In this specification, the FTP commands `PORT` and `PASV`, defined in RFC 959, are replaced with `XPRT` and `XPSV`, respectively.

This report is organized as follows. Section 2 discusses the syntax of the `XPRT` command. Section 3 provides the syntax for the `XPSV` command. Section 4 provides some implementation suggestions. Section 5 discusses the implication of using these extensions in the presence of Network Address Translators (NATs). Section 6 gives a brief discussion of the security impacts these changes may have on the Internet. Finally, Section 7 gives conclusions.

*This is an extended version of RFC 2428 [AOM98] that contains material that was not standardized by the IETF. This report is *not* consistent with [AOM98]. This report should be taken as a historic document published in the hope that the additional extensions specified in this report will be helpful to the standards process at some point in the future.

2 The `XPRT` Command

The `XPRT` command allows for the specification of an extended address for the data connection. The extended address MUST consist of the network and transport protocols as well as the network and transport addresses. The format of the `XPRT` command is:

```
XPRT<space><d><net-prt><d><trans-prt><d><net-addr><d><trans-addr><d>
```

The `XPRT` command keyword MUST be followed by a single space (ASCII 32). Following the space, a delimiter character (`<d>`) must be specified. The delimiter character MUST be one of the ASCII characters in the range 33-126 inclusive. In choosing a delimiter, one must be careful not to choose a character that appears in the network or transport layer addresses, such as the "." (see below for address formats). The character "`|`" (ASCII 124) is suggested as the delimiter unless it coincides with a character needed to encode the network or transport address.

The `<net-prt>` and `<trans-prt>` arguments MUST be upper-case strings indicating the protocol to be used (and, implicitly, the address length). This document defines keywords for the network and transport protocols given in Tables 1 and 2. Keywords for additional protocols will be specified as needed. Note that this document does not specify that the protocol keywords be taken from official IANA documents, as specified in RFC 2428 [AOM98]; this allows the `XPRT/XPSV` commands to be used with network and/or transport protocols which are not Internet standards.

Keyword	**Protocol**
`IP4`	Internet Protocol, Version 4 [Pos81a]
`IP6`	Internet Protocol, Version 6 [DH96]

Table 1: Network Protocol Tags

Keyword	**Protocol**
`TCP`	Transmission Control Protocol [Pos81b]
`RDP`	Reliable Data Protocol [PH90]

Table 2: Transport Protocol Tags

The `<net-addr>` and `<trans-addr>` are protocol specific string representations of the respective addresses. For each of the keywords given in Tables 1 and 2, addresses MUST be in the format given in Tables 3 and 4. This address format (which is the same format as used for the `EPRT/EPSV` commands in RFC 2428, but different from `PORT/PASV` in RFC 959 [PR85]) has implications for NATs [SH99], as discussed in Section 5.

Keyword	**Address Format**	**Example**
`IP4`	dotted decimal	132.235.1.2
`IP6`	IPv6 string representations defined in [HD96]	1080::8:800:200C:417A

Table 3: Network Protocol Format

Keyword	**Address Format**	**Example**
`TCP`	string representation of decimal port number	6446
`RDP`	string representation of decimal port number	3684

Table 4: Transport Protocol Format

The `<net-prt>`, `<trans-prt>` and `<net-addr>` fields are optional. If left blank, their default values are as given in Table 5.

The following are sample `XPRT` commands:

```
XPRT |IP4|TCP|132.235.1.2|6275|
XPRT ||RDP||5282|
```

The first command specifies that the server should use IPv4/TCP to open a data connection to the host "132.235.1.2" on port 6275. The second command specifies that the server should use the network protocol and network address used by the control connection to open an RDP data connection on port 5282.

Upon receipt of a valid `XPRT` command, the server MUST return a code of 200 (Command OK). The standard negative error codes 500 and 501 [PR85] are sufficient to handle most errors (e.g., syntax errors) involving the `XPRT` command. However, two additional error codes are needed. The response code 522 indicates that the server does not support the requested network protocol. Likewise, the response code 523 indicates that the server does not support the requested transport protocol. The interpretation of these error codes is given in Table 6.

The text portion of the error response MUST indicate which protocol(s) the server does support. If the network protocol is unsupported, the format of the response string MUST be:

```
<text stating that the network protocol is unsupported> (prot1,prot2,...,protn)
```

In this document, any text enclosed within "<>" is informational text that can be written in any language but MUST NOT include parenthesis. In the above case, the text SHOULD indicate that the network protocol in the `XPRT` command is not supported by the server. Two example response strings follow:

```
Supported network protocols (IP6)
Supported network protocols (IP4,IP6)
```

Similarly, if the transport protocol is not supported, the format of the response string MUST be:

```
<text stating that the transport protocol is unsupported> (prot1,prot2,...,protn)
```

where the protocols are identified by the keywords listed above for the `XPRT` command. In the case when both the network and transport protocols specified in the `XPRT` command are unsupported, the error reported MUST be 522 (unsupported network protocol).

Field	**Default Value If Omitted**
`<net-prt>`	Network protocol of the control connection
`<trans-prt>`	Transport protocol of the control connection
`<net-addr>`	Network address of the control connection

Table 5: Default Values

3 The `XPSV` Command

The `XPSV` command, like the original `PASV` command, requests that a server listen on a data port and wait for a connection. The `XPSV` command takes an optional argument. The response to this command includes all information needed to setup a connection using the `XPRT` command. The response code for entering passive mode using an extended address MUST be 229. The interpretation of this code, according to [PR85] is given in Table 7.

Error Code	Explanation
5yz	Negative Completion
x2z	Connections
xy2	Extended Port Failure - unknown network protocol
xy3	Extended Port Failure - unknown transport protocol

Table 6: XPRT Error Codes

Error Code	Explanation
2yz	Positive Completion
x2z	Connections
xy9	Extended Passive Mode Entered

Table 7: XPSV Error Codes

The text returned in response to a valid XPSV command MUST be:

```
<text indicating server is entering extended passive mode> \
    (<d><net-prt><d><trans-prt><d><net-addr><d><trans-addr><d>)
```

The portion of the string enclosed in parentheses MUST be the exact string needed by the XPRT command to open the data connection, as specified above. As with the XPRT command, the first three fields in the XPSV response are optional. Similar to the XPRT commands, when left blank, these fields default to the values used for the control connection. An example response string follows:

```
Entering Extended Passive Mode (|IP4|TCP|132.235.1.2|6446|)
```

The standard negative error codes 500 and 501 are sufficient to handle all errors involving the XPSV command (e.g., syntax errors).

When the XPSV command is issued with no argument, the server will choose the network and transport protocols for the data connection. However, since it is possible for the server to return an unsupported protocol in the XPSV response, the client needs to be able to request a specific network/transport protocol pair. If the server returns a protocol pair that the client does not support, the client will not be able to open a data connection to the server. In this situation, the client MUST issue an ABOR (abort) command to allow the server to close down the listening connection. The client can then send an XPSV command requesting the use of a specific network and transport protocol pair, as follows:

```
XPSV<space><d><net-prt><d><trans-prt><d>
```

Either the net-prt or the trans-prt field may be omitted, as with the XPRT command, in which case they take on the same values as the control connection (see Table 5). If the requested protocol pair is supported by the server, it SHOULD use the protocol. If not, the server MUST return error message 522 or 523, as outlined in Section 2. The client may issue either form of the XPSV command at any time. In other words, the version without arguments need not be issued before the version with arguments. Finally, the XPSV command can be used with the argument "ALL" to inform Network Address Translators that the XPRT command (as well as other data commands) will no longer be used. An example of this command follows:

```
XPSV<space>ALL
```

Upon receipt of an XPSV ALL command, the server MUST reject all data connection setup commands other than XPSV (i.e., XPRT, EPRT, PORT, PASV, et al.). This use of the XPSV command is further explained in Section 4.

4 Recommended Usage

To aid in transition from IPv4 to IPv6 it is RECOMMENDED that the network address be omitted from the `XPRT` command and the `XPSV` response whenever possible. This will allow the end hosts to utilize standard IPv6 mechanisms to communicate (such as network address translators), rather than forcing FTP to negotiate the network protocol.

For all FTP transfers where the control and data connection(s) are being established between the same two machines, the `XPSV` command SHOULD be used. Using the `XPSV` command benefits performance of transfers that traverse firewalls or Network Address Translators (NATs). RFC 1579 [Bel94] recommends using the passive command when behind firewalls since firewalls do not generally allow incoming connections (which are required when using the `PORT` (`XPRT`) command). In addition, using `XPSV` as defined in this document does not require NATs to change the network address in the traffic as it is forwarded. The NAT may have to change the address if the `XPRT` command were used. Finally, if the client issues an "`XPSV ALL`" command, NATs may be able to put the connection on a "fast path" through the translator, as the `XPRT` command will never be used and therefore, translation of the data portion of the segments will never be needed. When a client only expects to do two-way FTP transfers, it SHOULD issue this command as soon as possible. If a client later finds that it must do a three-way FTP transfer after issuing an `XPSV ALL` command, a new FTP session MUST be started.

5 NAT Implications

The use of Network Address Translators (NATs) is becoming more common on the Internet. NATs are often used in situations in which local machines do not have globally-unique IP addresses. They can also be used on small internets to allow multiple computers to appear to the outside world to have a single IP address, which is a common scenario for home use when several machines connect to the outside world over a modem connected to an Internet Service Provider that only provides a single IP address. NATs can also be useful if the IP address prefix of a site needs to be changed, but not all of the local hosts have been converted. In all of these situations, NATs perform a translation between IP addresses for incoming and outgoing packets.

A NAT must be prepared to watch for embedded IP addresses in the FTP control stream and modify them according to its policy. In the worst case, the new IP address would require a different number of bytes for encoding (consider translating between IP address 1.2.3.4 and address 132.235.212.117), which would require the NAT to alter the TCP sequence numbers and acknowledgment numbers for all future TCP segments on the control connection. In an even worse scenario, a longer IP address could force a TCP segment to require fragmentation in the NAT, although the relatively large MTUs of modern networks and the small average case of TCP segments on an FTP control connection would make this an unlikely event.

When the recommendations in Section 4 are followed, FTP can be used very efficiently with NATs. The recommended exchange is shown (for an exchange using only TCP over IP) in Figure 1. Notice that network addresses never need to be sent across the control connection, so a NAT never needs to translate the contents of the control data stream. In addition, because the client sent `XPSV ALL`, most NATs would not even need to monitor the control stream. Note, however, that some NATs may also need to modify the transport addresses used in FTP; these types of NATs would still need to be prepared to handle the worst-case scenarios discussed above.

Client	Server
establish control connection	
send `XPSV ALL`	
send `XPSV |||`	
	reply `XPRT ||||3456|`
client connects to port 3456	
	server sends file
send `XPSV |||`	
	reply `XPRT ||||3457|`
client connects to port 3457	
	server sends file

Figure 1: Example NAT-friendly FTP Exchange of Two Files

6 Security Issues

These FTP extensions increase the scope of the FTP "bounce attack" by making it possible for connections to be made using a transport protocol other than TCP. See [AO99] for a more general discussion of FTP security issues and techniques to reduce these security problems.

7 Conclusions

The extensions specified in this paper will enable FTP to operate over a variety of network and transport protocols.

Acknowledgments

We gratefully acknowledge the assistance and advice of Craig Metz, as well as many of the members of the IETF's FTP Extensions working group in the preparation of the both the original version of this document and this extended version.

References

[AO99] Mark Allman and Shawn Ostermann. FTP Security Considerations, May 1999. RFC 2577.

[AOM98] Mark Allman, Shawn Ostermann, and Craig Metz. FTP Extensions for IPv6 and NATs, September 1998. RFC 2428.

[Bel94] S. Bellovin. Firewall-Friendly FTP, February 1994. RFC 1579.

[Bra97] Scott Bradner. Key Words for Use in RFCs to Indicate Requirement Levels, March 1997. RFC 2119.

[Com95] Douglas E. Comer. *Internetworking with TCP/IP, Volume I, Principles, Protocols, and Architecture.* Prentice Hall, 3rd edition, 1995.

[DH96] Steve Deering and Robert Hinden. Internet Protocol, Version 6 (IPv6) Specification, January 1996. RFC 1883.

[HD96] Robert Hinden and Steve Deering. IP Version 6 Addressing Architecture, January 1996. RFC 1884.

[PH90] Craig Partridge and Robert Hinden. Version 2 of the Reliable Data Protocol (RDP), April 1990. RFC 1151.

[Pis94] D. Piscitello. FTP Operation Over Big Address Records (FOOBAR), June 1994. RFC 1639.

[Pos81a] Jon Postel. Internet Protocol, September 1981. RFC 791.

[Pos81b] Jon Postel. Transmission Control Protocol, September 1981. RFC 793.

[PR85] Jon Postel and Joyce Reynolds. File Tranfer Protocol (FTP), October 1985. RFC 959.

[SH99] Pyda Srisuresh and Matt Holdrege. IP Network Address Translator (NAT) Terminology and Considerations, August 1999. RFC 2663.