NWG/RFC# 743 KLH 30-Dec-77 08:39 42759
Network Working Group
Request for Comments: 743
NIC: 42758
K. Harrenstien
SRI-KL
30 December 1977

FTP extension: XRSQ/XRCP

This RFC describes an extension to FTP which allows the user of an ITS FTP server (i.e. on MIT-(AI/ML/MC/DMS)) to mail the text of a message to several recipients simultaneously; such message transmission is far more efficient than the current practice of sending the text again and again for each additional recipient at a site.

Within this extension, there are two basic ways of sending a single text to several recipients. In one, all recipients are specified first, and then the text is sent; in the other, the order is reversed and the text is sent first, followed by the recipients. Both schemes are necessary becaue neither by itself is optimal for all systems, as will be explained later. To select a particular scheme, the XRSQ command is used; to specify recipients after a scheme is chosen, XRCP commands are given; and to furnish text, the usual MAIL or MLFL commands apply.

Scheme Selection: XRSQ

XRSQ is the means by which a user program can test for implementation of XRSQ/XRCP, select a particular scheme, reset its state thereof, and even do some rudimentary negotiation. Its format is like that of the TYPE command, as follows:

XRSQ [<SP> <scheme>] <CRLF>

<scheme> = a single character. The following are defined:
R Recipients first. If not implemented, T must be.
T Text first. If this is not implemented, R must be.
? Request for preference. Must always be implemented.

No argument means a "selection" of none of the schemes (the default).

Replies:
200 OK, we'll use specified scheme.
215 <scheme> This is the scheme I prefer.
501 I understand XRSQ but can't use that scheme.
5xx Command unrecognized or unimplemented.
See Appendix A for more about the choice of reply codes.

Three aspects of XRSQ need to be pointed out here. The first is that

an XRSQ with no argument must always return a 200 reply and restore the default state of having no scheme selected. Any other reply implies that XRSQ and hence XRCP are not understood or cannot be performed correctly.

The second is that the use of "?" as a <scheme> asks the FTP server to return a 215 reply in which the server specifies a "preferred" scheme. The format of this reply is simple:

215 <SP> <scheme> [<SP> <arbitrary text>] <CRLF>

Any other reply (e.g. 4xx or 5xx) implies that XRSQ and XRCP are not implemented, because "?" must always be implemented if XRSQ is.

The third important thing about XRSQ is that it always has the side effect of resetting all schemes to their initial state. This reset must be done no matter what the reply will be - 200, 215, or 501. The actions necessary for a reset will be explained when discussing how each scheme actually works.

Message Text Specification: MAIL/MLFL

Regardless of which scheme (if any) has been selected, a MAIL or MLFL with a non-null argument will behave exactly as before; this extension has no effect on them. However, such normal MAIL/MLFL commands do have the same side effect as XRSQ; they "reset" the current scheme to its initial state.

It is only when the argument is null (e.g. MAIL<CRLF> or MLFL<CRLF>) that the particular scheme being used is important, because rather than producing an error (as most servers currently do), the server will accept message text for this "null" specification; what it does with it depends on which scheme is in effect, and will be described in "Scheme Mechanics".

Recipient specification: XRCP

In order to specify recipient names and receive some acknowledgement (or refusal) for each name, the following new command is also defined:

XRCP <SP> <Recipient name> <CRLF>

Reply for no scheme:
507 No scheme specified yet; use XRSQ.
Replies for scheme T are identical to those for MAIL/MLFL.

Replies for scheme R (recipients first):
200 OK, name stored.
440 Recipient table full, this name not stored.
450 Recipient name rejected. (Permanent!)
520 Recipient name rejected.
4xx Temporary error, try this name again later.
5xx Permanent error, report to sender.
See Appendix A for more about the choice of reply codes.

Note that use of this command is an error if no scheme has been selected yet; an XRSQ <scheme> must have been given if XRCP is to be used.

Scheme mechanics: XRSQ R (Recipients first)

In the recipients-first scheme, XRCP is used to specify names which the FTP server stores in a list or table. Normally the reply for each XRCP will be either a 200 for acceptance, or a 4xx/5xx code for rejection; 450 and all 5xx codes are permanent rejections (e.g. user not known) which should be reported to the human sender, whereas 4xx codes in general connote some temporary error that may be rectified later. None of the 4xx/5xx replies impinge on previous or succeeding XRCP commands, except for 440 which indicates that no further XRCP's will succeed unless a message is sent to the already stored recipients or a reset is done.

Sending message text to stored recipients is done by giving a MAIL or MLFL command with no argument; that is, just MAIL<CRLF> or MLFL<CRLF>. Transmission of the message text is exactly the same as for normal MAIL/MLFL; however, a positive acknowledgement at the end of transmission means that the message has been sent to ALL recipients that were remembered with XRCP, and a failure code means that it should be considered to have failed for ALL of these specified recipients. This applies regardless of the actual error code; and whether the reply signifies success or failure, all stored recipient names are flushed and forgotten - in other words, things are reset to their initial state. This purging of the recipient name list must also be done as the "reset" side effect of any use of XRSQ.

A 440 reply to an XRCP can thus be handled by using a MAIL/MLFL to specify the message for currently stored recipients, and then sending more XRCP's and another MAIL/MLFL, as many times as necessary; for example, if a server only had room for 10 names this would result in a 50-recipient message being sent 5 times, to 10 different recipients each time.

If a user attempts to specify message text (MAIL/MLFL with no

argument) before any successful XRCP's have been given, this should be treated exactly as a "normal" MAIL/MLFL with a null recipient would be; most servers will return an error of some type, such as "450 Null recipient".

See Appendix B for an example using XRSQ R.

Scheme mechanics: XRSQ T (Text first)

In the text-first scheme, MAIL/MLFL with no argument is used to specify message text, which the server stores away. Succeeding XRCP's are then treated as if they were MAIL/MLFL commands, except that none of the text transfer manipulations are done; the stored message text is sent to the specified recipient, and a reply code is returned identical to that which an actual MAIL/MLFL would invoke. (Note ANY 2xx code indicates success.)

The stored message text is not forgotten until the next MAIL/MLFL or XRSQ, which will either replace it with new text or flush it entirely. Any use of XRSQ will reset this scheme by flushing stored text, as will any use of MAIL/MLFL with a non-null argument.

If an XRCP is seen before any message text has been stored, the user in effect is trying to send a null message; some servers might allow this, others would return an error code.

See Appendix C for an example using XRSQ T.

Why two schemes anyway?

Because neither by itself is optimal for all systems. XRSQ R allows more of a "bulk" mailing, because everything is saved up and then mailed simultaneously; this is very useful for systems such as ITS where the FTP server does not itself write mail directly, but hands it on to a central mailer demon of great power; the more information (e.g. recipients) associated with a single "hand-off", the more efficiently mail can be delivered.

By contrast, XRSQ T is geared to FTP servers which want to deliver mail directly, in one-by-one incremental fashion. This way they can return an individual success/failure reply code for each recipient given which may depend on variable file system factors such as exceeding disk allocation, mailbox access conflicts, and so forth; if they tried to emulate XRSQ R's bulk mailing, they would have to ensure that a success reply to the MAIL/MLFL indeed meant that it had been delivered to ALL recipients specified - not just some.

Stray notes:

* Because this is after all an extension of FTP protocol, one must be prepared to deal with sites which don't recognize either XRSQ or XRCP. "XRSQ" and "XRSQ ?" are explicitly designed as tests to see whether either scheme is implemented; XRCP is not, and a failure return of the "unimplemented" variety could be confused with "No scheme selected yet", or even with "Recipient unknown". Be safe, be sure, use XRSQ!

* There is no way to indicate in a positive response to "XRSQ ?" that the preferred "scheme" for a server is that of the default state; i.e. none of the multi-recipient schemes. The rationale is that in this case, it would be pointless to implement XRSQ/XRCP at all, and the response would therefore be negative.

* One reason that the use of MAIL/MLFL is restricted to null arguments with this multi-recipient extension is the ambiguity that would result if a non-null argument were allowed; for example, if XRSQ R was in effect and some XRCP's had been given, and a MAIL FOO<CRLF> was done, there would be no way to distinguish a failure reply for mailbox "FOO" from a global failure for all recipients specified. A similar situation exists for XRSQ T; it would not be clear whether the text was stored and the mailbox failed, or vice versa, or both.

* "Resets" are done by all XRSQ's and "normal" MAIL/MLFL's to avoid confusion and overly complicated implementation. The XRSQ command implies a change or uncertainty of status, and the latter commands would otherwise have to use some independent mechanisms to avoid clobbering the data bases (e.g. message text storage area) used by the T/R schemes. However, once a scheme is selected, it remains "in effect" just as a "TYPE A" or "BYTE 8" remains selected. The recommended way for doing a reset, without changing the current selection, is with "XRSQ ?". Remember that "XRSQ" alone reverts to the no-scheme state.

* It is permissible to intersperse other FTP commands among the XRSQ/XRCP/MAIL sequences.

On FTP reply codes

The choice of appropriate reply codes for new or experimental commands is difficult because there have been three possible "official" sets of codes which one may draw on, and it is not clear which of them might be in use at any particular site; these are (1) Old FTP, (2) New FTP, (3) Revised New FTP. In my choice of code assignments, I have for the most part ignored these and used RFC 691, "One More Try on the FTP", by Brian Harvey. My motivation for this is the simple observation that I know of no site which implements "new FTP", and RFC 691 incorporates much of the "new FTP" reply code logic into the framework of "old FTP". The only sharp conflict is treated by allowing 450 to have its "old" meaning, equivalent to 520 - permanent failure. Note that when testing to see whether a site understands a FTP command, a reply of 5xx (specifically, 500) will generally indicate, for all sets of codes, that the command is unrecognized.

By the way, I recommend RFC 691 as required reading for FTP implementors; maybe if enough people get together this mess can be straightened out.

Example of XRSQ R (Recipients first)

This is an example of how XRSQ R is used; first the user must establish that the server in fact implements XRSQ:

U: XRSQ
S: 200 OK, no scheme selected.

An XRSQ with a null argument always returns a 200 if implemented, selecting the "scheme" of null, i.e. none of them. If XRSQ were not implemented, a code of 4xx or 5xx would be returned.

U: XRSQ R
S: 200 OK, using that scheme

All’s well; now the recipients can be specified.

U: XRCP Foo
S: 200 OK

U: XRCP Raboof
S: 520 Who’s that? No such user here.

U: XRCP bar
S: 200 OK

Well, two out of three ain’t bad. Note that the demise of "Raboof" has no effect on the storage of "Foo" or "bar". Now to furnish the message text, by giving a MAIL or MLFL with no argument:

U: MAIL
S: 350 Type mail, ended by <CRLF>.<CRLF>
U: Blah blah blah blah....etc etc etc
U: .
S: 256 Mail sent.

The text has now been sent to both "Foo" and "bar".

Example of XRSQ T (Text first)

Using the same message as the previous example:

```
U: XRSQ ?
S: 215 T Text first, please.
```

XRSQ is indeed implemented, and the server says that it prefers "T", but that needn't stop the user from trying something else:

```
U: XRSQ R
S: 501 Sorry, I really can't do that.
```

Oh well. It's possible that it could have understood "R" also, but in general it's best to use the "preferred" scheme, since the server knows which is most efficient for its particular site. Anyway:

```
U: XRSQ T
S: 200 OK, using that scheme.
```

Scheme "T" is now selected, and the text must be sent:

```
U: MAIL
S: 350 Type mail, ended by <CRLF>.<CRLF>
U: Blah blah blah blah....etc etc etc
U: .
S: 256 Mail stored.
```

Now recipients can be specified:

```
U: XRCP Foo
S: 256 Stored mail sent.

U: XRCP Raboof
S: 520 Who's that? No such user here.

U: XRCP bar
S: 256 Stored mail sent.
```

Again, the text has now been sent to both "Foo" and "bar", and still remains stored. A new message can be sent with another MAIL/XRCP... sequence, but the fastidious or paranoid could chose to do:

```
U: XRSQ ?
S: 215 T Text first, please.
```

Which resets things without altering the scheme in effect.