Network Working Group   C. Feather
Request for Comments: 3977   THUS plc
Obsoletes: 977   October 2006
Updates: 2980
Category: Standards Track

                  Network News Transfer Protocol (NNTP)

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   The Network News Transfer Protocol (NNTP) has been in use in the
   Internet for a decade, and remains one of the most popular protocols
   (by volume) in use today. This document is a replacement for
   RFC 977, and officially updates the protocol specification. It
   clarifies some vagueness in RFC 977, includes some new base
   functionality, and provides a specific mechanism to add standardized
   extensions to NNTP.

Table of Contents

1. Introduction

   This document specifies the Network News Transfer Protocol (NNTP),
   which is used for the distribution, inquiry, retrieval, and posting
   of Netnews articles using a reliable stream-based mechanism. For
   news-reading clients, NNTP enables retrieval of news articles that

are stored in a central database, giving subscribers the ability to
select only those articles they wish to read.

The Netnews model provides for indexing, cross-referencing, and
expiration of aged messages. NNTP is designed for efficient
transmission of Netnews articles over a reliable full duplex
communication channel.

Although the protocol specification in this document is largely
compatible with the version specified in RFC 977 [RFC977], a number
of changes are summarised in Appendix D. In particular:

o   the default character set is changed from US-ASCII [ANSI1986] to
    UTF-8 [RFC3629] (note that US-ASCII is a subset of UTF-8);

o   a number of commands that were optional in RFC 977 or that have
    been taken from RFC 2980 [RFC2980] are now mandatory; and

o   a CAPABILITIES command has been added to allow clients to
    determine what functionality is available from a server.

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 RFC 2119 [RFC2119].

An implementation is not compliant if it fails to satisfy one or more
of the MUST requirements for this protocol. An implementation that
satisfies all the MUST and all the SHOULD requirements for its
protocols is said to be "unconditionally compliant"; one that
satisfies all the MUST requirements but not all the SHOULD
requirements for NNTP is said to be "conditionally compliant".

For the remainder of this document, the terms "client" and "client
host" refer to a host making use of the NNTP service, while the terms
"server" and "server host" refer to a host that offers the NNTP
service.

1.1. Author's Note

This document is written in XML using an NNTP-specific DTD. Custom
software is used to convert this to RFC 2629 [RFC2629] format, and
then the public "xml2rfc" package to further reduce this to text,
nroff source, and HTML.

No perl was used in producing this document.

2. Notation

   The following notational conventions are used in this document.

      UPPERCASE   indicates literal text to be included in the
                       command.

      lowercase   indicates a token described elsewhere.

      [brackets]   indicate that the enclosed material is optional.

      elliptical   indicates that the argument may be repeated any
      ... marks   number of times (it must occur at least once).

      vertical|bar   indicates a choice of two mutually exclusive
                       arguments (exactly one must be provided).

   The name "message-id" for a command or response argument indicates
   that it is the message-id of an article as described in Section 3.6,
   including the angle brackets.

   The name "wildmat" for an argument indicates that it is a wildmat as
   defined in Section 4. If the argument does not meet the requirements
   of that section (for example, if it does not fit the grammar of
   Section 4.1), the NNTP server MAY place some interpretation on it
   (not specified by this document) or otherwise MUST treat it as a
   syntax error.

   Responses for each command will be described in tables listing the
   required format of a response followed by the meaning that should be
   ascribed to that response.

   The terms "NUL", "TAB", "LF", "CR, and "space" refer to the octets
   %x00, %x09, %x0A, %x0D, and %x20, respectively (that is, the octets
   with those codes in US-ASCII [ANSI1986] and thus in UTF-8 [RFC3629]).
   The term "CRLF" or "CRLF pair" means the sequence CR immediately
   followed by LF (that is, %x0D.0A). A "printable US-ASCII character"
   is an octet in the range %x21-7E. Quoted characters refer to the
   octets with those codes in US-ASCII (so "." and "<" refer to %x2E and
   %x3C) and will always be printable US-ASCII characters; similarly,
   "digit" refers to the octets %x30-39.

   A "keyword" MUST consist only of US-ASCII letters, digits, and the
   characters dot (".") and dash ("-") and MUST begin with a letter.
   Keywords MUST be at least three characters in length.

Examples in this document are not normative but serve to illustrate
usages, arguments, and responses. In the examples, a "[C]" will be
used to represent the client host and an "[S]" will be used to
represent the server host. Most of the examples do not rely on a
particular server state. In some cases, however, they do assume that
the currently selected newsgroup (see the GROUP command,
Section 6.1.1) is invalid; when so, this is indicated at the start of
the example. Examples may use commands or other keywords not defined
in this specification (such as an XENCRYPT command). These will be
used to illustrate some point and do not imply that any such command
is defined elsewhere or needs to exist in any particular
implementation.

Terms that might be read as specifying details of a client or server
implementation, such as "database", are used simply to ease
description. Provided that implementations conform to the protocol
and format specifications in this document, no specific technique is
mandated.

3. Basic Concepts

3.1. Commands and Responses

NNTP operates over any reliable bi-directional 8-bit-wide data stream
channel. When the connection is established, the NNTP server host
MUST send a greeting. The client host and server host then exchange
commands and responses (respectively) until the connection is closed
or aborted. If the connection used is TCP, then the server host
starts the NNTP service by listening on a TCP port. When a client
host wishes to make use of the service, it MUST establish a TCP
connection with the server host by connecting to that host on the
same port on which the server is listening.

The character set for all NNTP commands is UTF-8 [RFC3629]. Commands
in NNTP MUST consist of a keyword, which MAY be followed by one or
more arguments. A CRLF pair MUST terminate all commands. Multiple
commands MUST NOT be on the same line. Unless otherwise noted
elsewhere in this document, arguments SHOULD consist of printable US-
ASCII characters. Keywords and arguments MUST each be separated by
one or more space or TAB characters. Command lines MUST NOT exceed
512 octets, which includes the terminating CRLF pair. The arguments
MUST NOT exceed 497 octets. A server MAY relax these limits for
commands defined in an extension.

Where this specification permits UTF-8 characters outside the range
of U+0000 to U+007F, implementations MUST NOT use the Byte Order Mark
(U+FEFF, encoding %xEF.BB.BF) and MUST use the Word Joiner (U+2060,
encoding %xE2.91.A0) for the meaning Zero Width No-Break Space in

command lines and the initial lines of responses. Implementations
SHOULD apply these same principles throughout.

The term "character" means a single Unicode code point.
Implementations are not required to carry out Unicode normalisation.
Thus, U+0084 (A-dieresis) is one character, while U+0041 U+0308 (A
composed with dieresis) is two; the two need not be treated as
equivalent.

Commands may have variants; if so, they use a second keyword
immediately after the first to indicate which variant is required.
The only such commands in this specification are LIST and MODE. Note
that such variants are sometimes referred to as if they were commands
in their own right: "the LIST ACTIVE" command should be read as
shorthand for "the ACTIVE variant of the LIST command".

Keywords are case insensitive; the case of keywords for commands MUST
be ignored by the server. Command and response arguments are case or
language specific only when stated, either in this document or in
other relevant specifications.

In some cases, a command involves more data than just a single line.
The further data may be sent either immediately after the command
line (there are no instances of this in this specification, but there
are in extensions such as [NNTP-STREAM]) or following a request from
the server (indicated by a 3xx response).

Each response MUST start with a three-digit response code that is
sufficient to distinguish all responses. Certain valid responses are
defined to be multi-line; for all others, the response is contained
in a single line. The initial line of the response MUST NOT exceed
512 octets, which includes the response code and the terminating CRLF
pair; an extension MAY specify a greater maximum for commands that it
defines, but not for any other command. Single-line responses
consist of an initial line only. Multi-line responses consist of an
initial line followed by a multi-line data block.

An NNTP server MAY have an inactivity autologout timer. Such a timer
SHOULD be of at least three minutes' duration, with the exception
that there MAY be a shorter limit on how long the server is willing
to wait for the first command from the client. The receipt of any
command from the client during the timer interval SHOULD suffice to
reset the autologout timer. Similarly, the receipt of any
significant amount of data from a client that is sending a multi-line
data block (such as during a POST or IHAVE command) SHOULD suffice to
reset the autologout timer. When the timer expires, the server
SHOULD close the connection without sending any response to the
client.

3.1.1. Multi-line Data Blocks

   A multi-line data block is used in certain commands and responses.
   It MUST adhere to the following rules:

   1. The block consists of a sequence of zero or more "lines", each
       being a stream of octets ending with a CRLF pair. Apart from
       those line endings, the stream MUST NOT include the octets NUL,
       LF, or CR.

   2. In a multi-line response, the block immediately follows the CRLF
       at the end of the initial line of the response. When used in any
       other context, the specific command will define when the block is
       sent.

   3. If any line of the data block begins with the "termination octet"
       ("." or %x2E), that line MUST be "dot-stuffed" by prepending an
       additional termination octet to that line of the block.

   4. The lines of the block MUST be followed by a terminating line
       consisting of a single termination octet followed by a CRLF pair
       in the normal way. Thus, unless it is empty, a multi-line block
       is always terminated with the five octets CRLF "." CRLF
       (%x0D.0A.2E.0D.0A).

   5. When a multi-line block is interpreted, the "dot-stuffing" MUST
       be undone; i.e., the recipient MUST ensure that, in any line
       beginning with the termination octet followed by octets other
       than a CRLF pair, that initial termination octet is disregarded.

   6. Likewise, the terminating line ("." CRLF or %x2E.0D.0A) MUST NOT
       be considered part of the multi-line block; i.e., the recipient
       MUST ensure that any line beginning with the termination octet
       followed immediately by a CRLF pair is disregarded. (The first
       CRLF pair of the terminating CRLF "." CRLF of a non-empty block
       is, of course, part of the last line of the block.)

   Note that texts using an encoding (such as UTF-16 or UTF-32) that may
   contain the octets NUL, LF, or CR other than a CRLF pair cannot be
   reliably conveyed in the above format (that is, they violate the MUST
   requirement above). However, except when stated otherwise, this
   specification does not require the content to be UTF-8, and therefore
   (subject to that same requirement) it MAY include octets above and
   below 128 mixed arbitrarily.

   This document does not place any limit on the length of a line in a
   multi-line block. However, the standards that define the format of
   articles may do so.

3.2. Response Codes

   Each response MUST begin with a three-digit status indicator. These
   are status reports from the server and indicate the response to the
   last command received from the client.

   The first digit of the response broadly indicates the success,
   failure, or progress of the previous command:

      1xx - Informative message
      2xx - Command completed OK
      3xx - Command OK so far; send the rest of it
      4xx - Command was syntactically correct but failed for some reason
      5xx - Command unknown, unsupported, unavailable, or syntax error

   The next digit in the code indicates the function response category:

      x0x - Connection, setup, and miscellaneous messages
      x1x - Newsgroup selection
      x2x - Article selection
      x3x - Distribution functions
      x4x - Posting
      x8x - Reserved for authentication and privacy extensions
      x9x - Reserved for private use (non-standard extensions)

   Certain responses contain arguments such as numbers and names in
   addition to the status indicator. In those cases, to simplify
   interpretation by the client, the number and type of such arguments
   is fixed for each response code, as is whether the code is
   single-line or multi-line. Any extension MUST follow this principle
   as well. Note that, for historical reasons, the 211 response code is
   an exception to this in that the response may be single-line or
   multi-line depending on the command (GROUP or LISTGROUP) that
   generated it. In all other cases, the client MUST only use the
   status indicator itself to determine the nature of the response. The
   exact response codes that can be returned by any given command are
   detailed in the description of that command.

   Arguments MUST be separated from the numeric status indicator and
   from each other by a single space. All numeric arguments MUST be in
   base 10 (decimal) format and MAY have leading zeros. String
   arguments MUST contain at least one character and MUST NOT contain
   TAB, LF, CR, or space. The server MAY add any text after the
   response code or last argument, as appropriate, and the client MUST
   NOT make decisions based on this text. Such text MUST be separated
   from the numeric status indicator or the last argument by at least
   one space.

The server MUST respond to any command with the appropriate generic
response (given in Section 3.2.1) if it represents the situation.
Otherwise, each recognized command MUST return one of the response
codes specifically listed in its description or in an extension. A
server MAY provide extensions to this specification, including new
commands, new variants or features of existing commands, and other
ways of changing the internal state of the server. However, the
server MUST NOT produce any other responses to a client that does not
invoke any of the additional features. (Therefore, a client that
restricts itself to this specification will only receive the
responses that are listed.)

If a client receives an unexpected response, it SHOULD use the first
digit of the response to determine the result. For example, an
unexpected 2xx should be taken as success, and an unexpected 4xx or
5xx as failure.

Response codes not specified in this document MAY be used for any
installation-specific additional commands also not specified. These
SHOULD be chosen to fit the pattern of x9x specified above.

Neither this document nor any registered extension (see
Section 3.3.3) will specify any response codes of the x9x pattern.
(Implementers of extensions are accordingly cautioned not to use such
responses for extensions that may subsequently be submitted for
registration.)

3.2.1. Generic Response Codes

The server MUST respond to any command with the appropriate one of
the following generic responses if it represents the situation.

If the command is not recognized, or if it is an optional command
that is not implemented by the server, the response code 500 MUST be
returned.

If there is a syntax error in the arguments of a recognized command,
including the case where more arguments are provided than the command
specifies or the command line is longer than the server accepts, the
response code 501 MUST be returned. The line MUST NOT be truncated
or split and then interpreted. Note that where a command has
variants depending on a second keyword (e.g., LIST ACTIVE and LIST
NEWSGROUPS), 501 MUST be used when the base command is implemented
but the requested variant is not, and 500 MUST be used only when the
base command itself is not implemented.

If an argument is required to be a base64-encoded string [RFC4648]
(there are no such arguments in this specification, but there may be

in extensions) and is not validly encoded, the response code 504 MUST
be returned.

If the server experiences an internal fault or problem that means it
is unable to carry out the command (for example, a necessary file is
missing or a necessary service could not be contacted), the response
code 403 MUST be returned. If the server recognizes the command but
does not provide an optional feature (for example, because it does
not store the required information), or if it only handles a subset
of legitimate cases (see the HDR command, Section 8.5, for an
example), the response code 503 MUST be returned.

If the client is not authorized to use the specified facility when
the server is in its current state, then the appropriate one of the
following response codes MUST be used.

502: It is necessary to terminate the connection and to start a new
   one with the appropriate authority before the command can be used.
   Historically, some mode-switching servers (see Section 3.4.1) used
   this response to indicate that this command will become available
   after the MODE READER command (Section 5.3) is used, but this
   usage does not conform to this specification and MUST NOT be used.
   Note that the server MUST NOT close the connection immediately
   after a 502 response except at the initial connection
   (Section 5.1) and with the MODE READER command.

480: The client must authenticate itself to the server (that is, it
   must provide information as to the identity of the client) before
   the facility can be used on this connection. This will involve
   the use of an authentication extension such as [NNTP-AUTH].

483: The client must negotiate appropriate privacy protection on the
   connection. This will involve the use of a privacy extension such
   as [NNTP-TLS].

401: The client must change the state of the connection in some other
   manner. The first argument of the response MUST be the capability
   label (see Section 5.2) of the facility that provides the
   necessary mechanism (usually an extension, which may be a private
   extension). The server MUST NOT use this response code except as
   specified by the definition of the capability in question.

If the server has to terminate the connection for some reason, it
MUST give a 400 response code to the next command and then
immediately close the connection. Following a 400 response, clients
SHOULD NOT simply reconnect immediately and retry the same actions.
Rather, a client SHOULD either use an exponentially increasing delay
between retries (e.g., double the waiting time after each 400

   response) or present any associated text to the user for them to
   decide whether and when to retry.

   The client MUST be prepared to receive any of these responses for any
   command (except, of course, that the server MUST NOT generate a 500
   response code for mandatory commands).

3.2.1.1. Examples

   Example of an unknown command:

      [C] MAIL
      [S] 500 Unknown command

   Example of an unsupported command:

      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] READER
      [S] NEWNEWS
      [S] LIST ACTIVE NEWSGROUPS
      [S] .
      [C] OVER
      [S] 500 Unknown command

   Example of an unsupported variant:

      [C] MODE POSTER
      [S] 501 Unknown MODE option

   Example of a syntax error:

      [C] ARTICLE a.message.id@no.angle.brackets
      [S] 501 Syntax error

   Example of an overlong command line:

      [C] HEAD 53 54 55
      [S] 501 Too many arguments

   Example of a bad wildmat:

      [C] LIST ACTIVE u[ks].*
      [S] 501 Syntax error

Example of a base64-encoding error (the second argument is meant to
be base64 encoded):

    [C] XENCRYPT RSA abcd=efg
    [S] 504 Base64 encoding error

Example of an attempt to access a facility not available to this
connection:

    [C] MODE READER
    [S] 200 Reader mode, posting permitted
    [C] IHAVE <i.am.an.article.you.will.want@example.com>
    [S] 500 Permission denied

Example of an attempt to access a facility requiring authentication:

    [C] GROUP secret.group
    [S] 480 Permission denied

Example of a successful attempt following such authentication:

    [C] XSECRET fred flintstone
    [S] 290 Password for fred accepted
    [C] GROUP secret.group
    [S] 211 5 1 20 secret.group selected

Example of an attempt to access a facility requiring privacy:

    [C] GROUP secret.group
    [S] 483 Secure connection required
    [C] XENCRYPT
    [Client and server negotiate encryption on the link]
    [S] 283 Encrypted link established
    [C] GROUP secret.group
    [S] 211 5 1 20 secret.group selected

Example of a need to change mode before a facility is used:

    [C] GROUP binary.group
    [S] 401 XHOST Not on this virtual host
    [C] XHOST binary.news.example.org
    [S] 290 binary.news.example.org virtual host selected
    [C] GROUP binary.group
    [S] 211 5 1 77 binary.group selected

   Example of a temporary failure:

      [C] GROUP archive.local
      [S] 403 Archive server temporarily offline

   Example of the server needing to close down immediately:

      [C] ARTICLE 123
      [S] 400 Power supply failed, running on UPS
      [Server closes connection.]

3.3. Capabilities and Extensions

   Not all NNTP servers provide exactly the same facilities, both
   because this specification allows variation and because servers may
   provide extensions. A set of facilities that are related are called
   a "capability". This specification provides a way to determine what
   capabilities are available, includes a list of standard capabilities,
   and includes a mechanism (the extension mechanism) for defining new
   capabilities.

3.3.1. Capability Descriptions

   A client can determine the available capabilities of the server by
   using the CAPABILITIES command (Section 5.2). This returns a
   capability list, which is a list of capability lines. Each line
   describes one available capability.

   Each capability line consists of one or more tokens, which MUST be
   separated by one or more space or TAB characters. A token is a
   string of 1 or more printable UTF-8 characters (that is, either
   printable US-ASCII characters or any UTF-8 sequence outside the US-
   ASCII range, but not space or TAB). Unless stated otherwise, tokens
   are case insensitive. Each capability line consists of the
   following:

   o The capability label, which is a keyword indicating the
      capability. A capability label may be defined by this
      specification or a successor, or by an extension.

   o The label is then followed by zero or more tokens, which are
      arguments of the capability. The form and meaning of these tokens
      is specific to each capability.

   The server MUST ensure that the capability list accurately reflects
   the capabilities (including extensions) currently available. If a
   capability is only available with the server in a certain state (for
   example, only after authentication), the list MUST only include the

capability label when the server is in that state. Similarly, if
only some of the commands in an extension will be available, or if
the behaviour of the extension will change in some other manner,
according to the state of the server, this MUST be indicated by
different arguments in the capability line.

Note that a capability line can only begin with a letter. Lines
beginning with other characters are reserved for future versions of
this specification. In order to interoperate with such versions,
clients MUST be prepared to receive lines beginning with other
characters and MUST ignore any they do not understand.

3.3.2. Standard Capabilities

The following capabilities are defined by this specification.

VERSION
   This capability MUST be advertised by all servers and MUST be the
   first capability in the capability list; it indicates the
   version(s) of NNTP that the server supports. There must be at
   least one argument; each argument is a decimal number and MUST NOT
   have a leading zero. Version numbers are assigned only in RFCs
   that update or replace this specification; servers MUST NOT create
   their own version numbers.

   The version number of this specification is 2.

READER
   This capability indicates that the server implements the various
   commands useful for reading clients.

IHAVE
   This capability indicates that the server implements the IHAVE
   command.

POST
   This capability indicates that the server implements the POST
   command.

NEWNEWS
   This capability indicates that the server implements the NEWNEWS
   command.

HDR
   This capability indicates that the server implements the header
   access commands (HDR and LIST HEADERS).

   OVER
      This capability indicates that the server implements the overview
      access commands (OVER and LIST OVERVIEW.FMT). If and only if the
      server supports the message-id form of the OVER command, there
      must be a single argument MSGID.

   LIST
      This capability indicates that the server implements at least one
      variant of the LIST command. There MUST be one argument for each
      variant of the LIST command supported by the server, giving the
      keyword for that variant.

   IMPLEMENTATION
      This capability MAY be provided by a server. If so, the arguments
      SHOULD be used to provide information such as the server software
      name and version number. The client MUST NOT use this line to
      determine capabilities of the server. (While servers often
      provide this information in the initial greeting, clients need to
      guess whether this is the case; this capability makes it clear
      what the information is.)

   MODE-READER
      This capability indicates that the server is mode-switching
      (Section 3.4.2) and that the MODE READER command needs to be used
      to enable the READER capability.

3.3.3. Extensions

   Although NNTP is widely and robustly deployed, some parts of the
   Internet community might wish to extend the NNTP service. It must be
   emphasized that any extension to NNTP should not be considered
   lightly. NNTP's strength comes primarily from its simplicity.
   Experience with many protocols has shown that:

      Protocols with few options tend towards ubiquity, whilst protocols
      with many options tend towards obscurity.

   This means that each and every extension, regardless of its benefits,
   must be carefully scrutinized with respect to its implementation,
   deployment, and interoperability costs. In many cases, the cost of
   extending the NNTP service will likely outweigh the benefit.

   An extension is a package of associated facilities, often but not
   always including one or more new commands. Each extension MUST
   define at least one new capability label (this will often, but need
   not, be the name of one of these new commands). While any additional
   capability information can normally be specified using arguments to

that label, an extension MAY define more than one capability label.
However, this SHOULD be limited to exceptional circumstances.

An extension is either a private extension, or its capabilities are
included in the IANA registry of capabilities (see Section 3.3.4) and
it is defined in an RFC (in which case it is a "registered
extension"). Such RFCs either must be on the standards track or must
define an IESG-approved experimental protocol.

The definition of an extension must include the following:

o a descriptive name for the extension.

o the capability label or labels defined by the extension (the
   capability label of a registered extension MUST NOT begin with
   "X").

o The syntax, values, and meanings of any arguments for each
   capability label defined by the extension.

o Any new NNTP commands associated with the extension (the names of
   commands associated with registered extensions MUST NOT begin with
   "X").

o The syntax and possible values of arguments associated with the
   new NNTP commands.

o The response codes and possible values of arguments for the
   responses of the new NNTP commands.

o Any new arguments the extension associates with any other
   pre-existing NNTP commands.

o Any increase in the maximum length of commands and initial
   response lines over the value specified in this document.

o A specific statement about the effect on pipelining that this
   extension may have (if any).

o A specific statement about the circumstances when use of this
   extension can alter the contents of the capabilities list (other
   than the new capability labels it defines).

o A specific statement about the circumstances under which the
   extension can cause any pre-existing command to produce a 401,
   480, or 483 response.

   o A description of how the use of MODE READER on a mode-switching
      server interacts with the extension.

   o A description of how support for the extension affects the
      behaviour of a server and NNTP client in any other manner not
      outlined above.

   o Formal syntax as described in Section 9.9.

   A private extension MAY or MAY NOT be included in the capabilities
   list. If it is, the capability label MUST begin with "X". A server
   MAY provide additional keywords (for new commands and also for new
   variants of existing commands) as part of a private extension. To
   avoid the risk of a clash with a future registered extension, these
   keywords SHOULD begin with "X".

   If the server advertises a capability defined by a registered
   extension, it MUST implement the extension so as to fully conform
   with the specification (for example, it MUST implement all the
   commands that the extension describes as mandatory). If it does not
   implement the extension as specified, it MUST NOT list the extension
   in the capabilities list under its registered name. In that case, it
   MAY, but SHOULD NOT, provide a private extension (not listed, or
   listed with a different name) that implements part of the extension
   or implements the commands of the extension with a different meaning.

   A server MUST NOT send different response codes to basic NNTP
   commands documented here or to commands documented in registered
   extensions in response to the availability or use of a private
   extension.

3.3.4. Initial IANA Register

   IANA will maintain a registry of NNTP capability labels. All
   capability labels in the registry MUST be keywords and MUST NOT begin
   with X.

The initial content of the registry consists of these entries:

Label	Meaning	Definition
AUTHINFO	Authentication	[NNTP-AUTH]
HDR	Batched header retrieval	Section 3.3.2, Section 8.5, and Section 8.6
IHAVE	IHAVE command available	Section 3.3.2 and Section 6.3.2
IMPLEMENTATION	Server implementation-specific information	Section 3.3.2
LIST	LIST command variants	Section 3.3.2 and Section 7.6.1
MODE-READER	Mode-switching server and MODE READER command available	Section 3.4.2
NEWNEWS	NEWNEWS command available	Section 3.3.2 and Section 7.4
OVER	Overview support	Section 3.3.2, Section 8.3, and Section 8.4
POST	POST command available	Section 3.3.2 and Section 6.3.1
READER	Reader commands available	Section 3.3.2
SASL	Supported SASL mechanisms	[NNTP-AUTH]
STARTTLS	Transport layer security	[NNTP-TLS]
STREAMING	Streaming feeds	[NNTP-STREAM]
VERSION	Supported NNTP versions	Section 3.3.2

3.4. Mandatory and Optional Commands

   For a number of reasons, not all the commands in this specification
   are mandatory. However, it is equally undesirable for every command
   to be optional, since this means that a client will have no idea what
   facilities are available. Therefore, as a compromise, some of the
   commands in this specification are mandatory (they must be supported
   by all servers) while the remainder are not. The latter are then
   subdivided into bundles, each indicated by a single capability label.

   o If the label is included in the capability list returned by the
      server, the server MUST support all commands in that bundle.

   o If the label is not included, the server MAY support none or some
      of the commands but SHOULD NOT support all of them. In general,
      there will be no way for a client to determine which commands are
      supported without trying them.

   The bundles have been chosen to provide useful functionality, and
   therefore server authors are discouraged from implementing only part
   of a bundle.

   The description of each command will either indicate that it is
   mandatory, or will give, using the term "indicating capability", the
   capability label indicating whether the bundle including this command
   is available.

   Where a server does not implement a command, it MUST always generate
   a 500 generic response code (or a 501 generic response code in the
   case of a variant of a command depending on a second keyword where
   the base command is recognised). Otherwise, the command MUST be
   fully implemented as specified; a server MUST NOT only partially
   implement any of the commands in this specification. (Client authors
   should note that some servers not conforming to this specification
   will return a 502 generic response code to some commands that are not
   implemented.)

   Note: some commands have cases that require other commands to be used
   first. If the former command is implemented but the latter is not,
   the former MUST still generate the relevant specific response code.
   For example, if ARTICLE (Section 6.2.1) is implemented but GROUP
   (Section 6.1.1) is not, the correct response to "ARTICLE 1234"
   remains 412.

3.4.1. Reading and Transit Servers

   NNTP is traditionally used in two different ways. The first use is
   "reading", where the client fetches articles from a large store
   maintained by the server for immediate or later presentation to a
   user and sends articles created by that user back to the server (an
   action called "posting") to be stored and distributed to other stores
   and users. The second use is for the bulk transfer of articles from
   one store to another. Since the hosts making this transfer tend to
   be peers in a network that transmit articles among one another, and
   not end-user systems, this process is called "peering" or "transit".
   (Even so, one host is still the client and the other is the server).

   In practice, these two uses are so different that some server
   implementations are optimised for reading or for transit and, as a
   result, do not offer the other facility or only offer limited
   features. Other implementations are more general and offer both.
   This specification allows for this by bundling the relevant commands
   accordingly: the IHAVE command is designed for transit, while the
   commands indicated by the READER capability are designed for reading
   clients.

   Except as an effect of the MODE READER command (Section 5.3) on a
   mode-switching server, once a server advertises either or both of the
   IHAVE or READER capabilities, it MUST continue to advertise them for
   the entire session.

   A server MAY provide different modes of behaviour (transit, reader,
   or a combination) to different client connections and MAY use
   external information, such as the IP address of the client, to
   determine which mode to provide to any given connection.

   The official TCP port for the NNTP service is 119. However, if a
   host wishes to offer separate servers for transit and reading
   clients, port 433 SHOULD be used for the transit server and 119 for
   the reading server.

3.4.2. Mode Switching

   An implementation MAY, but SHOULD NOT, provide both transit and
   reader facilities on the same server but require the client to select
   which it wishes to use. Such an arrangement is called a
   "mode-switching" server.

A mode-switching server has two modes:

o Transit mode, which applies after the initial connection.

   * It MUST advertise the MODE-READER capability.

   * It MUST NOT advertise the READER capability.

   However, the server MAY cease to advertise the MODE-READER
   capability after the client uses any command except CAPABILITIES.

o Reading mode, after a successful MODE READER command (see Section
   5.3).

   * It MUST NOT advertise the MODE-READER capability.

   * It MUST advertise the READER capability.

   * It MAY NOT advertise the IHAVE capability, even if it was
      advertising it in transit mode.

A client SHOULD only issue a MODE READER command to a server if it is
advertising the MODE-READER capability. If the server does not
support CAPABILITIES (and therefore does not conform to this
specification), the client MAY use the following heuristic:

o If the client wishes to use any "reader" commands, it SHOULD use
   the MODE READER command immediately after the initial connection.

o Otherwise, it SHOULD NOT use the MODE READER command.

In each case, it should be prepared for some commands to be
unavailable that would have been available if it had made the other
choice.

3.5. Pipelining

NNTP is designed to operate over a reliable bi-directional
connection, such as TCP. Therefore, if a command does not depend on
the response to the previous one, it should not matter if it is sent
before that response is received. Doing this is called "pipelining".
However, certain server implementations throw away all text received
from the client following certain commands before sending their
response. If this happens, pipelining will be affected because one
or more commands will have been ignored or misinterpreted, and the
client will be matching the wrong responses to each command. Since
there are significant benefits to pipelining, but also circumstances
where it is reasonable or common for servers to behave in the above

manner, this document puts certain requirements on both clients and
servers.

Except where stated otherwise, a client MAY use pipelining. That is,
it may send a command before receiving the response for the previous
command. The server MUST allow pipelining and MUST NOT throw away
any text received after a command. Irrespective of whether
pipelining is used, the server MUST process commands in the order
they are sent.

If the specific description of a command says it "MUST NOT be
pipelined", that command MUST end any pipeline of commands. That is,
the client MUST NOT send any following command until it receives the
CRLF at the end of the response from the command. The server MAY
ignore any data received after the command and before the CRLF at the
end of the response is sent to the client.

The initial connection must not be part of a pipeline; that is, the
client MUST NOT send any command until it receives the CRLF at the
end of the greeting.

If the client uses blocking system calls to send commands, it MUST
ensure that the amount of text sent in pipelining does not cause a
deadlock between transmission and reception. The amount of text
involved will depend on window sizes in the transmission layer;
typically, it is 4k octets for TCP. (Since the server only sends
data in response to commands from the client, the converse problem
does not occur.)

3.5.1. Examples

Example of correct use of pipelining:

    [C] GROUP misc.test
    [C] STAT
    [C] NEXT
    [S] 211 1234 3000234 3002322 misc.test
    [S] 223 3000234 <45223423@example.com> retrieved
    [S] 223 3000237 <668929@example.org> retrieved

Example of incorrect use of pipelining (the MODE READER command may
not be pipelined):

    [C] MODE READER
    [C] DATE
    [C] NEXT
    [S] 200 Server ready, posting allowed
    [S] 223 3000237 <668929@example.org> retrieved

The DATE command has been thrown away by the server, so there is no
111 response to match it.

3.6. Articles

   NNTP is intended to transfer articles between clients and servers.
   For the purposes of this specification, articles are required to
   conform to the rules in this section, and clients and servers MUST
   correctly process any article received from the other that does so.
   Note that this requirement applies only to the contents of
   communications over NNTP; it does not prevent the client or server
   from subsequently rejecting an article for reasons of local policy.
   Also see Appendix A for further restrictions on the format of
   articles in some uses of NNTP.

   An article consists of two parts: the headers and the body. They are
   separated by a single empty line, or in other words by two
   consecutive CRLF pairs (if there is more than one empty line, the
   second and subsequent ones are part of the body). In order to meet
   the general requirements of NNTP, an article MUST NOT include the
   octet NUL, MUST NOT contain the octets LF and CR other than as part
   of a CRLF pair, and MUST end with a CRLF pair. This specification
   puts no further restrictions on the body; in particular, it MAY be
   empty.

   The headers of an article consist of one or more header lines. Each
   header line consists of a header name, a colon, a space, the header
   content, and a CRLF, in that order. The name consists of one or more
   printable US-ASCII characters other than colon and, for the purposes
   of this specification, is not case sensitive. There MAY be more than
   one header line with the same name. The content MUST NOT contain
   CRLF; it MAY be empty. A header may be "folded"; that is, a CRLF
   pair may be placed before any TAB or space in the line. There MUST
   still be some other octet between any two CRLF pairs in a header
   line. (Note that folding means that the header line occupies more
   than one line when displayed or transmitted; nevertheless, it is
   still referred to as "a" header line.) The presence or absence of
   folding does not affect the meaning of the header line; that is, the
   CRLF pairs introduced by folding are not considered part of the
   header content. Header lines SHOULD NOT be folded before the space
   after the colon that follows the header name and SHOULD include at
   least one octet other than %x09 or %x20 between CRLF pairs. However,
   if an article that fails to satisfy this requirement has been
   received from elsewhere, clients and servers MAY transfer it to each
   other without re-folding it.

The content of a header SHOULD be in UTF-8. However, if an
implementation receives an article from elsewhere that uses octets in
the range 128 to 255 in some other manner, it MAY pass it to a client
or server without modification. Therefore, implementations MUST be
prepared to receive such headers, and data derived from them (e.g.,
in the responses from the OVER command, Section 8.3), and MUST NOT
assume that they are always UTF-8. Any external processing of those
headers, including identifying the encoding used, is outside the
scope of this document.

Each article MUST have a unique message-id; two articles offered by
an NNTP server MUST NOT have the same message-id. For the purposes
of this specification, message-ids are opaque strings that MUST meet
the following requirements:

o A message-id MUST begin with "<", end with ">", and MUST NOT
   contain the latter except at the end.

o A message-id MUST be between 3 and 250 octets in length.

o A message-id MUST NOT contain octets other than printable US-ASCII
   characters.

Two message-ids are the same if and only if they consist of the same
sequence of octets.

This specification does not describe how the message-id of an article
is determined. If the server does not have any way to determine a
message-id from the article itself, it MUST synthesize one (this
specification does not require that the article be changed as a
result). See also Appendix A.2.

4. The WILDMAT Format

The WILDMAT format described here is based on the version first
developed by Rich Salz [SALZ1992], which was in turn derived from the
format used in the UNIX "find" command to articulate file names. It
was developed to provide a uniform mechanism for matching patterns in
the same manner that the UNIX shell matches filenames.

4.1. Wildmat Syntax

   A wildmat is described by the following ABNF [RFC4234] syntax, which
   is an extract of that in Section 9.8.

      wildmat = wildmat-pattern *("," ["!"] wildmat-pattern)
      wildmat-pattern = 1*wildmat-item
      wildmat-item = wildmat-exact / wildmat-wild
      wildmat-exact = %x22-29 / %x2B / %x2D-3E / %x40-5A / %x5E-7E /
          UTF8-non-ascii ; exclude ! * , ? [\ ]
      wildmat-wild = "*" / "?"

   Note: the characters ",", "\", "[", and "]" are not allowed in
   wildmats, while * and ? are always wildcards. This should not be a
   problem, since these characters cannot occur in newsgroup names,
   which is the only current use of wildmats. Backslash is commonly
   used to suppress the special meaning of characters, whereas brackets
   are used to introduce sets. However, these usages are not universal,
   and interpretation of these characters in the context of UTF-8
   strings is potentially complex and differs from existing practice, so
   they were omitted from this specification. A future extension to
   this specification may provide semantics for these characters.

4.2. Wildmat Semantics

   A wildmat is tested against a string and either matches or does not
   match. To do this, each constituent <wildmat-pattern> is matched
   against the string, and the rightmost pattern that matches is
   identified. If that <wildmat-pattern> is not preceded with "!", the
   whole wildmat matches. If it is preceded by "!", or if no <wildmat-
   pattern> matches, the whole wildmat does not match.

   For example, consider the wildmat "a*,!*b,*c*":

   o The string "aaa" matches because the rightmost match is with "a*".

   o The string "abb" does not match because the rightmost match is
      with "*b".

   o The string "ccb" matches because the rightmost match is with
      "*c*".

   o The string "xxx" does not match because no <wildmat-pattern>
      matches.

   A <wildmat-pattern> matches a string if the string can be broken into
   components, each of which matches the corresponding <wildmat-item> in
   the pattern. The matches must be in the same order, and the whole

string must be used in the match. The pattern is "anchored"; that
is, the first and last characters in the string must match the first
and last item, respectively (unless that item is an asterisk matching
zero characters).

A <wildmat-exact> matches the same character (which may be more than
one octet in UTF-8).

"?" matches exactly one character (which may be more than one octet).

"*" matches zero or more characters. It can match an empty string,
but it cannot match a subsequence of a UTF-8 sequence that is not
aligned to the character boundaries.

4.3. Extensions

An NNTP server or extension MAY extend the syntax or semantics of
wildmats provided that all wildmats that meet the requirements of
Section 4.1 have the meaning ascribed to them by Section 4.2. Future
editions of this document may also extend wildmats.

4.4. Examples

In these examples, $ and @ are used to represent the two octets %xC2
and %xA3, respectively; $@ is thus the UTF-8 encoding for the pound
sterling symbol, shown as # in the descriptions.

```
  Wildmat     Description of strings that match
    abc       The one string "abc"
    abc,def   The two strings "abc" and "def"
    $@        The one character string "#"
    a*        Any string that begins with "a"
    a*b       Any string that begins with "a" and ends with "b"
    a*,*b     Any string that begins with "a" or ends with "b"
    a*,!*b    Any string that begins with "a" and does not end with
              "b"
  a*,!*b,c*   Any string that begins with "a" and does not end with
              "b", and any string that begins with "c" no matter
              what it ends with
  a*,c*,!*b   Any string that begins with "a" or "c" and does not
              end with "b"
    ?a*       Any string with "a" as its second character
    ??a*      Any string with "a" as its third character
    *a?       Any string with "a" as its penultimate character
    *a??      Any string with "a" as its antepenultimate character
```

5. Session Administration Commands

5.1. Initial Connection

5.1.1. Usage

   This command MUST NOT be pipelined.

   Responses [1]
     200   Service available, posting allowed
     201   Service available, posting prohibited
     400   Service temporarily unavailable [2]
     502   Service permanently unavailable [2]

   [1] These are the only valid response codes for the initial greeting;
       the server MUST not return any other generic response code.

   [2] Following a 400 or 502 response, the server MUST immediately
       close the connection.

5.1.2. Description

   There is no command presented by the client upon initial connection
   to the server. The server MUST present an appropriate response code
   as a greeting to the client. This response informs the client
   whether service is available and whether the client is permitted to
   post.

   If the server will accept further commands from the client including
   POST, the server MUST present a 200 greeting code. If the server
   will accept further commands from the client, but the client is not
   authorized to post articles using the POST command, the server MUST
   present a 201 greeting code.

   Otherwise, the server MUST present a 400 or 502 greeting code and
   then immediately close the connection. 400 SHOULD be used if the
   issue is only temporary (for example, because of load) and the client
   can expect to be able to connect successfully at some point in the
   future without making any changes. 502 MUST be used if the client is
   not permitted under any circumstances to interact with the server,
   and MAY be used if the server has insufficient information to
   determine whether the issue is temporary or permanent.

   Note: the distinction between the 200 and 201 response codes has
   turned out in practice to be insufficient; for example, some servers
   do not allow posting until the client has authenticated, while other
   clients assume that a 201 response means that posting will never be
   possible even after authentication. Therefore, clients SHOULD use

the CAPABILITIES command (Section 5.2) rather than rely on this
response.

5.1.3. Examples

Example of a normal connection from an authorized client that then
terminates the session (see Section 5.4):

    [Initial connection set-up completed.]
    [S] 200 NNTP Service Ready, posting permitted
    [C] QUIT
    [S] 205 NNTP Service exits normally
    [Server closes connection.]

Example of a normal connection from an authorized client that is not
permitted to post, which also immediately terminates the session:

    [Initial connection set-up completed.]
    [S] 201 NNTP Service Ready, posting prohibited
    [C] QUIT
    [S] 205 NNTP Service exits normally
    [Server closes connection.]

Example of a normal connection from an unauthorized client:

    [Initial connection set-up completed.]
    [S] 502 NNTP Service permanently unavailable
    [Server closes connection.]

Example of a connection from a client if the server is unable to
provide service:

    [Initial connection set-up completed.]
    [S] 400 NNTP Service temporarily unavailable
    [Server closes connection.]

5.2. CAPABILITIES

5.2.1. Usage

This command is mandatory.

Syntax
  CAPABILITIES [keyword]

Responses
  101   Capability list follows (multi-line)

        Parameters
          keyword   additional feature, see description

5.2.2. Description

      The CAPABILITIES command allows a client to determine the
      capabilities of the server at any given time.

      This command MAY be issued at any time; the server MUST NOT require
      it to be issued in order to make use of any capability. The response
      generated by this command MAY change during a session because of
      other state information (which, in turn, may be changed by the
      effects of other commands or by external events). An NNTP client is
      only able to get the current and correct information concerning
      available capabilities at any point during a session by issuing a
      CAPABILITIES command at that point of that session and processing the
      response.

      The capability list is returned as a multi-line data block following
      the 101 response code. Each capability is described by a separate
      capability line. The server MUST NOT list the same capability twice
      in the response, even with different arguments. Except that the
      VERSION capability MUST be the first line, the order in which the
      capability lines appears is not significant; the server need not even
      consistently return the same order.

      While some capabilities are likely to be always available or never
      available, others (notably extensions) will appear and disappear
      depending on server state changes within the session or on external
      events between sessions. An NNTP client MAY cache the results of
      this command, but MUST NOT rely on the correctness of any cached
      results, whether from earlier in this session or from a previous
      session, MUST cope gracefully with the cached status being out of
      date, and SHOULD (if caching results) provide a way to force the
      cached information to be refreshed. Furthermore, a client MUST NOT
      use cached results in relation to security, privacy, and
      authentication extensions. See Section 12.6 for further discussion
      of this topic.

      The keyword argument is not used by this specification. It is
      provided so that extensions or revisions to this specification can
      include extra features for this command without requiring the
      CAPABILITIES command to be used twice (once to determine if the extra
      features are available, and a second time to make use of them). If
      the server does not recognise the argument (and it is a keyword), it
      MUST respond with the 101 response code as if the argument had been
      omitted. If an argument is provided that the server does recognise,
      it MAY use the 101 response code or MAY use some other response code

(which will be defined in the specification of that feature). If the
argument is not a keyword, the 501 generic response code MUST be
returned. The server MUST NOT generate any other response code to
the CAPABILITIES command.

5.2.3. Examples

Example of a minimal response (a read-only server):

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] LIST ACTIVE NEWSGROUPS
[S] .
```

Example of a response from a server that has a range of facilities
and that also describes itself:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] IHAVE
[S] POST
[S] NEWNEWS
[S] LIST ACTIVE NEWSGROUPS ACTIVE.TIMES OVERVIEW.FMT
[S] IMPLEMENTATION INN 4.2 2004-12-25
[S] OVER MSGID
[S] STREAMING
[S] XSECRET
[S] .
```

Example of a server that supports more than one version of NNTP:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2 3
[S] READER
[S] LIST ACTIVE NEWSGROUPS
[S] .
```

Example of a client attempting to use a feature of the CAPABILITIES
command that the server does not support:

```
[C] CAPABILITIES AUTOUPDATE
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] IHAVE
[S] LIST ACTIVE NEWSGROUPS OVERVIEW.FMT HEADERS
[S] OVER MSGID
[S] HDR
[S] NEWNEWS
[S] .
```

5.3. MODE READER

5.3.1. Usage

   Indicating capability: MODE-READER

   This command MUST NOT be pipelined.

   Syntax
     MODE READER

   Responses
     200   Posting allowed
     201   Posting prohibited
     502   Reading service permanently unavailable [1]

   [1] Following a 502 response the server MUST immediately close the
       connection.

5.3.2. Description

   The MODE READER command instructs a mode-switching server to switch
   modes, as described in Section 3.4.2.

   If the server is mode-switching, it switches from its transit mode to
   its reader mode, indicating this by changing the capability list
   accordingly. It MUST then return a 200 or 201 response with the same
   meaning as for the initial greeting (as described in Section 5.1.1).
   Note that the response need not be the same as that presented during
   the initial greeting. The client MUST NOT issue MODE READER more
   than once in a session or after any security or privacy commands are
   issued. When the MODE READER command is issued, the server MAY reset
   its state to that immediately after the initial connection before
   switching mode.

   If the server is not mode-switching, then the following apply:

   o If it advertises the READER capability, it MUST return a 200 or
      201 response with the same meaning as for the initial greeting; in
      this case, the command MUST NOT affect the server state in any
      way.

   o If it does not advertise the READER capability, it MUST return a
      502 response and then immediately close the connection.

5.3.3. Examples

   Example of use of the MODE READER command on a transit-only server
   (which therefore does not providing reading facilities):

      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] IHAVE
      [S] .
      [C] MODE READER
      [S] 502 Transit service only
      [Server closes connection.]

   Example of use of the MODE READER command on a server that provides
   reading facilities:

      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] READER
      [S] LIST ACTIVE NEWSGROUPS
      [S] .
      [C] MODE READER
      [S] 200 Reader mode, posting permitted
      [C] IHAVE <i.am.an.article.you.have@example.com>
      [S] 500 Permission denied
      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test

   Note that in both of these situations, the client SHOULD NOT use MODE
   READER.

   Example of use of the MODE READER command on a mode-switching server:

      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] IHAVE
      [S] MODE-READER
      [S] .
      [C] MODE READER
      [S] 200 Reader mode, posting permitted
      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] READER
      [S] NEWNEWS
      [S] LIST ACTIVE NEWSGROUPS
      [S] STARTTLS
      [S] .

   In this case, the server offers (but does not require) TLS privacy in
   its reading mode but not in its transit mode.

   Example of use of the MODE READER command where the client is not
   permitted to post:

      [C] MODE READER
      [S] 201 NNTP Service Ready, posting prohibited

5.4. QUIT

5.4.1. Usage

   This command is mandatory.

   Syntax
     QUIT

   Responses
     205   Connection closing

5.4.2. Description

   The client uses the QUIT command to terminate the session. The
   server MUST acknowledge the QUIT command and then close the
   connection to the client. This is the preferred method for a client
   to indicate that it has finished all of its transactions with the
   NNTP server.

If a client simply disconnects (or if the connection times out or
some other fault occurs), the server MUST gracefully cease its
attempts to service the client, disconnecting from its end if
necessary.

The server MUST NOT generate any response code to the QUIT command
other than 205 or, if any arguments are provided, 501.

5.4.3. Examples

```
[C] QUIT
[S] 205 closing connection
[Server closes connection.]
```

6. Article Posting and Retrieval

News-reading clients have available a variety of mechanisms to
retrieve articles via NNTP. The news articles are stored and indexed
using three types of keys. The first type of key is the message-id
of an article and is globally unique. The second type of key is
composed of a newsgroup name and an article number within that
newsgroup. On a particular server, there MUST only be one article
with a given number within any newsgroup, and an article MUST NOT
have two different numbers in the same newsgroup. An article can be
cross-posted to multiple newsgroups, so there may be multiple keys
that point to the same article on the same server; these MAY have
different numbers in each newsgroup. However, this type of key is
not required to be globally unique, so the same key MAY refer to
different articles on different servers. (Note that the terms
"group" and "newsgroup" are equivalent.)

The final type of key is the arrival timestamp, giving the time that
the article arrived at the server. The server MUST ensure that
article numbers are issued in order of arrival timestamp; that is,
articles arriving later MUST have higher numbers than those that
arrive earlier. The server SHOULD allocate the next sequential
unused number to each new article.

Article numbers MUST lie between 1 and 2,147,483,647, inclusive. The
client and server MAY use leading zeroes in specifying article
numbers but MUST NOT use more than 16 digits. In some situations,
the value zero replaces an article number to show some special
situation.

Note that it is likely that the article number limit of 2,147,483,647
will be increased by a future revision or extension to this
specification. While servers MUST NOT send article numbers greater
than this current limit, client and server developers are advised to

use internal structures and datatypes capable of handling larger
values in anticipation of such a change.

6.1. Group and Article Selection

The following commands are used to set the "currently selected
newsgroup" and the "current article number", which are used by
various commands. At the start of an NNTP session, both of these
values are set to the special value "invalid".

6.1.1. GROUP

6.1.1.1. Usage

Indicating capability: READER

Syntax
  GROUP group

Responses
  211 number low high group   Group successfully selected
  411   No such newsgroup

Parameters
  group   Name of newsgroup
  number   Estimated number of articles in the group
  low   Reported low water mark
  high   Reported high water mark

6.1.1.2. Description

The GROUP command selects a newsgroup as the currently selected
newsgroup and returns summary information about it.

The required argument is the name of the newsgroup to be selected
(e.g., "news.software.nntp"). A list of valid newsgroups may be
obtained by using the LIST ACTIVE command (see Section 7.6.3).

The successful selection response will return the article numbers of
the first and last articles in the group at the moment of selection
(these numbers are referred to as the "reported low water mark" and
the "reported high water mark") and an estimate of the number of
articles in the group currently available.

If the group is not empty, the estimate MUST be at least the actual
number of articles available and MUST be no greater than one more
than the difference between the reported low and high water marks.
(Some implementations will actually count the number of articles

currently stored. Others will just subtract the low water mark from
the high water mark and add one to get an estimate.)

If the group is empty, one of the following three situations will
occur. Clients MUST accept all three cases; servers MUST NOT
represent an empty group in any other way.

o The high water mark will be one less than the low water mark, and
   the estimated article count will be zero. Servers SHOULD use this
   method to show an empty group. This is the only time that the
   high water mark can be less than the low water mark.

o All three numbers will be zero.

o The high water mark is greater than or equal to the low water
   mark. The estimated article count might be zero or non-zero; if
   it is non-zero, the same requirements apply as for a non-empty
   group.

The set of articles in a group may change after the GROUP command is
carried out:

o Articles may be removed from the group.

o Articles may be reinstated in the group with the same article
   number, but those articles MUST have numbers no less than the
   reported low water mark (note that this is a reinstatement of the
   previous article, not a new article reusing the number).

o New articles may be added with article numbers greater than the
   reported high water mark. (If an article that was the one with
   the highest number has been removed and the high water mark has
   been adjusted accordingly, the next new article will not have the
   number one greater than the reported high water mark.)

Except when the group is empty and all three numbers are zero,
whenever a subsequent GROUP command for the same newsgroup is issued,
either by the same client or a different client, the reported low
water mark in the response MUST be no less than that in any previous
response for that newsgroup in this session, and it SHOULD be no less
than that in any previous response for that newsgroup ever sent to
any client. Any failure to meet the latter condition SHOULD be
transient only. The client may make use of the low water mark to
remove all remembered information about articles with lower numbers,
as these will never recur. This includes the situation when the high
water mark is one less than the low water mark. No similar
assumption can be made about the high water mark, as this can

   decrease if an article is removed and then increase again if it is
   reinstated or if new articles arrive.

   When a valid group is selected by means of this command, the
   currently selected newsgroup MUST be set to that group, and the
   current article number MUST be set to the first article in the group
   (this applies even if the group is already the currently selected
   newsgroup). If an empty newsgroup is selected, the current article
   number is made invalid. If an invalid group is specified, the
   currently selected newsgroup and current article number MUST NOT be
   changed.

   The GROUP or LISTGROUP command (see Section 6.1.2) MUST be used by a
   client, and a successful response received, before any other command
   is used that depends on the value of the currently selected newsgroup
   or current article number.

   If the group specified is not available on the server, a 411 response
   MUST be returned.

6.1.1.3. Examples

   Example for a group known to the server:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test

   Example for a group unknown to the server:

      [C] GROUP example.is.sob.bradner.or.barber
      [S] 411 example.is.sob.bradner.or.barber is unknown

   Example of an empty group using the preferred response:

      [C] GROUP example.currently.empty.newsgroup
      [S] 211 0 4000 3999 example.currently.empty.newsgroup

   Example of an empty group using an alternative response:

      [C] GROUP example.currently.empty.newsgroup
      [S] 211 0 0 0 example.currently.empty.newsgroup

   Example of an empty group using a different alternative response:

      [C] GROUP example.currently.empty.newsgroup
      [S] 211 0 4000 4321 example.currently.empty.newsgroup

Example reselecting the currently selected newsgroup:

```
[C] GROUP misc.test
[S] 211 1234 234 567 misc.test
[C] STAT 444
[S] 223 444 <123456@example.net> retrieved
[C] GROUP misc.test
[S] 211 1234 234 567 misc.test
[C] STAT
[S] 223 234 <different@example.net> retrieved
```

6.1.2. LISTGROUP

6.1.2.1. Usage

Indicating capability: READER

Syntax
  LISTGROUP [group [range]]

Responses
  211 number low high group   Article numbers follow (multi-line)
  411   No such newsgroup
  412   No newsgroup selected [1]

Parameters
  group   Name of newsgroup
  range   Range of articles to report
  number   Estimated number of articles in the group
  low   Reported low water mark
  high   Reported high water mark

[1] The 412 response can only occur if no group has been specified.

6.1.2.2. Description

The LISTGROUP command selects a newsgroup in the same manner as the
GROUP command (see Section 6.1.1) but also provides a list of article
numbers in the newsgroup. If no group is specified, the currently
selected newsgroup is used.

On success, a list of article numbers is returned as a multi-line
data block following the 211 response code (the arguments on the
initial response line are the same as for the GROUP command). The
list contains one number per line and is in numerical order. It
lists precisely those articles that exist in the group at the moment
of selection (therefore, an empty group produces an empty list). If
the optional range argument is specified, only articles within the

    range are included in the list (therefore, the list MAY be empty even
    if the group is not).

    The range argument may be any of the following:

    o An article number.

    o An article number followed by a dash to indicate all following.

    o An article number followed by a dash followed by another article
       number.

    In the last case, if the second number is less than the first number,
    then the range contains no articles. Omitting the range is
    equivalent to the range 1- being specified.

    If the group specified is not available on the server, a 411 response
    MUST be returned. If no group is specified and the currently
    selected newsgroup is invalid, a 412 response MUST be returned.

    Except that the group argument is optional, that a range argument can
    be specified, and that a multi-line data block follows the 211
    response code, the LISTGROUP command is identical to the GROUP
    command. In particular, when successful, the command sets the
    current article number to the first article in the group, if any,
    even if this is not within the range specified by the second
    argument.

    Note that the range argument is a new feature in this specification
    and servers that do not support CAPABILITIES (and therefore do not
    conform to this specification) are unlikely to support it.

6.1.2.3. Examples

    Example of LISTGROUP being used to select a group:

       [C] LISTGROUP misc.test
       [S] 211 2000 3000234 3002322 misc.test list follows
       [S] 3000234
       [S] 3000237
       [S] 3000238
       [S] 3000239
       [S] 3002322
       [S] .

   Example of LISTGROUP on an empty group:

      [C] LISTGROUP example.empty.newsgroup
      [S] 211 0 0 0 example.empty.newsgroup list follows
      [S] .

   Example of LISTGROUP on a valid, currently selected newsgroup:

      [C] GROUP misc.test
      [S] 211 2000 3000234 3002322 misc.test
      [C] LISTGROUP
      [S] 211 2000 3000234 3002322 misc.test list follows
      [S] 3000234
      [S] 3000237
      [S] 3000238
      [S] 3000239
      [S] 3002322
      [S] .

   Example of LISTGROUP with a range:

      [C] LISTGROUP misc.test 3000238-3000248
      [S] 211 2000 3000234 3002322 misc.test list follows
      [S] 3000238
      [S] 3000239
      [S] .

   Example of LISTGROUP with an empty range:

      [C] LISTGROUP misc.test 12345678-
      [S] 211 2000 3000234 3002322 misc.test list follows
      [S] .

   Example of LISTGROUP with an invalid range:

      [C] LISTGROUP misc.test 9999-111
      [S] 211 2000 3000234 3002322 misc.test list follows
      [S] .

6.1.3. LAST

6.1.3.1. Usage

   Indicating capability: READER

   Syntax
     LAST

   Responses
     223 n message-id   Article found
     412   No newsgroup selected
     420   Current article number is invalid
     422   No previous article in this group

   Parameters
     n   Article number
     message-id   Article message-id

6.1.3.2. Description

   If the currently selected newsgroup is valid, the current article
   number MUST be set to the previous article in that newsgroup (that
   is, the highest existing article number less than the current article
   number). If successful, a response indicating the new current
   article number and the message-id of that article MUST be returned.
   No article text is sent in response to this command.

   There MAY be no previous article in the group, although the current
   article number is not the reported low water mark. There MUST NOT be
   a previous article when the current article number is the reported
   low water mark.

   Because articles can be removed and added, the results of multiple
   LAST and NEXT commands MAY not be consistent over the life of a
   particular NNTP session.

   If the current article number is already the first article of the
   newsgroup, a 422 response MUST be returned. If the current article
   number is invalid, a 420 response MUST be returned. If the currently
   selected newsgroup is invalid, a 412 response MUST be returned. In
   all three cases, the currently selected newsgroup and current article
   number MUST NOT be altered.

6.1.3.3. Examples

   Example of a successful article retrieval using LAST:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] NEXT
      [S] 223 3000237 <668929@example.org> retrieved
      [C] LAST
      [S] 223 3000234 <45223423@example.com> retrieved

   Example of an attempt to retrieve an article without having selected
   a group (via the GROUP command) first:

      [Assumes currently selected newsgroup is invalid.]
      [C] LAST
      [S] 412 no newsgroup selected

   Example of an attempt to retrieve an article using the LAST command
   when the current article number is that of the first article in the
   group:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] LAST
      [S] 422 No previous article to retrieve

   Example of an attempt to retrieve an article using the LAST command
   when the currently selected newsgroup is empty:

      [C] GROUP example.empty.newsgroup
      [S] 211 0 0 0 example.empty.newsgroup
      [C] LAST
      [S] 420 No current article selected

6.1.4. NEXT

6.1.4.1. Usage

   Indicating capability: READER

   Syntax
     NEXT

   Responses
     223 n message-id   Article found
     412   No newsgroup selected
     420   Current article number is invalid
     421   No next article in this group

   Parameters
     n   Article number
     message-id   Article message-id

6.1.4.2. Description

   If the currently selected newsgroup is valid, the current article
   number MUST be set to the next article in that newsgroup (that is,
   the lowest existing article number greater than the current article
   number). If successful, a response indicating the new current
   article number and the message-id of that article MUST be returned.
   No article text is sent in response to this command.

   If the current article number is already the last article of the
   newsgroup, a 421 response MUST be returned. In all other aspects
   (apart, of course, from the lack of 422 response), this command is
   identical to the LAST command (Section 6.1.3).

6.1.4.3. Examples

   Example of a successful article retrieval using NEXT:

     [C] GROUP misc.test
     [S] 211 1234 3000234 3002322 misc.test
     [C] NEXT
     [S] 223 3000237 <668929@example.org> retrieved

Example of an attempt to retrieve an article without having selected
a group (via the GROUP command) first:

```
[Assumes currently selected newsgroup is invalid.]
[C] NEXT
[S] 412 no newsgroup selected
```

Example of an attempt to retrieve an article using the NEXT command
when the current article number is that of the last article in the
group:

```
[C] GROUP misc.test
[S] 211 1234 3000234 3002322 misc.test
[C] STAT 3002322
[S] 223 3002322 <411@example.net> retrieved
[C] NEXT
[S] 421 No next article to retrieve
```

Example of an attempt to retrieve an article using the NEXT command
when the currently selected newsgroup is empty:

```
[C] GROUP example.empty.newsgroup
[S] 211 0 0 0 example.empty.newsgroup
[C] NEXT
[S] 420 No current article selected
```

6.2. Retrieval of Articles and Article Sections

The ARTICLE, BODY, HEAD, and STAT commands are very similar. They
differ only in the parts of the article that are presented to the
client and in the successful response code. The ARTICLE command is
described here in full, while the other three commands are described
in terms of the differences. As specified in Section 3.6, an article
consists of two parts: the article headers and the article body.

When responding to one of these commands, the server MUST present the
entire article or appropriate part and MUST NOT attempt to alter or
translate it in any way.

6.2.1. ARTICLE

6.2.1.1. Usage

   Indicating capability: READER

   Syntax
     ARTICLE message-id
     ARTICLE number
     ARTICLE

   Responses

   First form (message-id specified)
     220 0|n message-id   Article follows (multi-line)
     430   No article with that message-id

   Second form (article number specified)
     220 n message-id   Article follows (multi-line)
     412   No newsgroup selected
     423   No article with that number

   Third form (current article number used)
     220 n message-id   Article follows (multi-line)
     412   No newsgroup selected
     420   Current article number is invalid

   Parameters
     number   Requested article number
     n   Returned article number
     message-id   Article message-id

6.2.1.2. Description

   The ARTICLE command selects an article according to the arguments and
   presents the entire article (that is, the headers, an empty line, and
   the body, in that order) to the client. The command has three forms.

   In the first form, a message-id is specified, and the server presents
   the article with that message-id. In this case, the server MUST NOT
   alter the currently selected newsgroup or current article number.
   This is both to facilitate the presentation of articles that may be
   referenced within another article being read, and because of the
   semantic difficulties of determining the proper sequence and
   membership of an article that may have been cross-posted to more than
   one newsgroup.

In the response, the article number MUST be replaced with zero,
unless there is a currently selected newsgroup and the article is
present in that group, in which case the server MAY use the article's
number in that group. (The server is not required to determine
whether the article is in the currently selected newsgroup or, if so,
what article number it has; the client MUST always be prepared for
zero to be specified.) The server MUST NOT provide an article number
unless use of that number in a second ARTICLE command immediately
following this one would return the same article. Even if the server
chooses to return article numbers in these circumstances, it need not
do so consistently; it MAY return zero to any such command (also see
the STAT examples, Section 6.2.4.3).

In the second form, an article number is specified. If there is an
article with that number in the currently selected newsgroup, the
server MUST set the current article number to that number.

In the third form, the article indicated by the current article
number in the currently selected newsgroup is used.

Note that a previously valid article number MAY become invalid if the
article has been removed. A previously invalid article number MAY
become valid if the article has been reinstated, but this article
number MUST be no less than the reported low water mark for that
group.

The server MUST NOT change the currently selected newsgroup as a
result of this command. The server MUST NOT change the current
article number except when an article number argument was provided
and the article exists; in particular, it MUST NOT change it
following an unsuccessful response.

Since the message-id is unique for each article, it may be used by a
client to skip duplicate displays of articles that have been posted
more than once, or to more than one newsgroup.

The article is returned as a multi-line data block following the 220
response code.

If the argument is a message-id and no such article exists, a 430
response MUST be returned. If the argument is a number or is omitted
and the currently selected newsgroup is invalid, a 412 response MUST
be returned. If the argument is a number and that article does not
exist in the currently selected newsgroup, a 423 response MUST be
returned. If the argument is omitted and the current article number
is invalid, a 420 response MUST be returned.

6.2.1.3. Examples

   Example of a successful retrieval of an article (explicitly not using
   an article number):

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] ARTICLE
      [S] 220 3000234 <45223423@example.com>
      [S] Path: pathost!demo!whitehouse!not-for-mail
      [S] From: "Demo User" <nobody@example.net>
      [S] Newsgroups: misc.test
      [S] Subject: I am just a test article
      [S] Date: 6 Oct 1998 04:38:40 -0500
      [S] Organization: An Example Net, Uncertain, Texas
      [S] Message-ID: <45223423@example.com>
      [S]
      [S] This is just a test article.
      [S] .

   Example of a successful retrieval of an article by message-id:

      [C] ARTICLE <45223423@example.com>
      [S] 220 0 <45223423@example.com>
      [S] Path: pathost!demo!whitehouse!not-for-mail
      [S] From: "Demo User" <nobody@example.net>
      [S] Newsgroups: misc.test
      [S] Subject: I am just a test article
      [S] Date: 6 Oct 1998 04:38:40 -0500
      [S] Organization: An Example Net, Uncertain, Texas
      [S] Message-ID: <45223423@example.com>
      [S]
      [S] This is just a test article.
      [S] .

   Example of an unsuccessful retrieval of an article by message-id:

      [C] ARTICLE <i.am.not.there@example.com>
      [S] 430 No Such Article Found

   Example of an unsuccessful retrieval of an article by number:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 news.groups
      [C] ARTICLE 300256
      [S] 423 No article with that number

   Example of an unsuccessful retrieval of an article by number because
   no newsgroup was selected first:

      [Assumes currently selected newsgroup is invalid.]
      [C] ARTICLE 300256
      [S] 412 No newsgroup selected

   Example of an attempt to retrieve an article when the currently
   selected newsgroup is empty:

      [C] GROUP example.empty.newsgroup
      [S] 211 0 0 0 example.empty.newsgroup
      [C] ARTICLE
      [S] 420 No current article selected

6.2.2. HEAD

6.2.2.1. Usage

   This command is mandatory.

   Syntax
     HEAD message-id
     HEAD number
     HEAD

   Responses

   First form (message-id specified)
     221 0|n message-id   Headers follow (multi-line)
     430   No article with that message-id

   Second form (article number specified)
     221 n message-id   Headers follow (multi-line)
     412   No newsgroup selected
     423   No article with that number

   Third form (current article number used)
     221 n message-id   Headers follow (multi-line)
     412   No newsgroup selected
     420   Current article number is invalid

   Parameters
     number   Requested article number
     n   Returned article number
     message-id   Article message-id

6.2.2.2. Description

   The HEAD command behaves identically to the ARTICLE command except
   that, if the article exists, the response code is 221 instead of 220
   and only the headers are presented (the empty line separating the
   headers and body MUST NOT be included).

6.2.2.3. Examples

   Example of a successful retrieval of the headers of an article
   (explicitly not using an article number):

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] HEAD
      [S] 221 3000234 <45223423@example.com>
      [S] Path: pathost!demo!whitehouse!not-for-mail
      [S] From: "Demo User" <nobody@example.net>
      [S] Newsgroups: misc.test
      [S] Subject: I am just a test article
      [S] Date: 6 Oct 1998 04:38:40 -0500
      [S] Organization: An Example Net, Uncertain, Texas
      [S] Message-ID: <45223423@example.com>
      [S] .

   Example of a successful retrieval of the headers of an article by
   message-id:

      [C] HEAD <45223423@example.com>
      [S] 221 0 <45223423@example.com>
      [S] Path: pathost!demo!whitehouse!not-for-mail
      [S] From: "Demo User" <nobody@example.net>
      [S] Newsgroups: misc.test
      [S] Subject: I am just a test article
      [S] Date: 6 Oct 1998 04:38:40 -0500
      [S] Organization: An Example Net, Uncertain, Texas
      [S] Message-ID: <45223423@example.com>
      [S] .

   Example of an unsuccessful retrieval of the headers of an article by
   message-id:

      [C] HEAD <i.am.not.there@example.com>
      [S] 430 No Such Article Found

Example of an unsuccessful retrieval of the headers of an article by
number:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] HEAD 300256
      [S] 423 No article with that number

Example of an unsuccessful retrieval of the headers of an article by
number because no newsgroup was selected first:

      [Assumes currently selected newsgroup is invalid.]
      [C] HEAD 300256
      [S] 412 No newsgroup selected

Example of an attempt to retrieve the headers of an article when the
currently selected newsgroup is empty:

      [C] GROUP example.empty.newsgroup
      [S] 211 0 0 0 example.empty.newsgroup
      [C] HEAD
      [S] 420 No current article selected

6.2.3. BODY

6.2.3.1. Usage

   Indicating capability: READER

   Syntax
     BODY message-id
     BODY number
     BODY

   Responses

   First form (message-id specified)
     222 0|n message-id   Body follows (multi-line)
     430   No article with that message-id

   Second form (article number specified)
     222 n message-id   Body follows (multi-line)
     412   No newsgroup selected
     423   No article with that number

```
   Third form (current article number used)
     222 n message-id      Body follows (multi-line)
     412                   No newsgroup selected
     420                   Current article number is invalid

   Parameters
     number        Requested article number
     n             Returned article number
     message-id    Article message-id
```

6.2.3.2. Description

   The BODY command behaves identically to the ARTICLE command except
   that, if the article exists, the response code is 222 instead of 220
   and only the body is presented (the empty line separating the headers
   and body MUST NOT be included).

6.2.3.3. Examples

   Example of a successful retrieval of the body of an article
   (explicitly not using an article number):

```
     [C] GROUP misc.test
     [S] 211 1234 3000234 3002322 misc.test
     [C] BODY
     [S] 222 3000234 <45223423@example.com>
     [S] This is just a test article.
     [S] .
```

   Example of a successful retrieval of the body of an article by
   message-id:

```
     [C] BODY <45223423@example.com>
     [S] 222 0 <45223423@example.com>
     [S] This is just a test article.
     [S] .
```

   Example of an unsuccessful retrieval of the body of an article by
   message-id:

```
     [C] BODY <i.am.not.there@example.com>
     [S] 430 No Such Article Found
```

Example of an unsuccessful retrieval of the body of an article by
number:

        [C] GROUP misc.test
        [S] 211 1234 3000234 3002322 misc.test
        [C] BODY 300256
        [S] 423 No article with that number

Example of an unsuccessful retrieval of the body of an article by
number because no newsgroup was selected first:

        [Assumes currently selected newsgroup is invalid.]
        [C] BODY 300256
        [S] 412 No newsgroup selected

Example of an attempt to retrieve the body of an article when the
currently selected newsgroup is empty:

        [C] GROUP example.empty.newsgroup
        [S] 211 0 0 0 example.empty.newsgroup
        [C] BODY
        [S] 420 No current article selected

6.2.4. STAT

6.2.4.1. Usage

   This command is mandatory.

   Syntax
     STAT message-id
     STAT number
     STAT

   Responses

   First form (message-id specified)
     223 0|n message-id   Article exists
     430   No article with that message-id

   Second form (article number specified)
     223 n message-id   Article exists
     412   No newsgroup selected
     423   No article with that number

   Third form (current article number used)
     223 n message-id   Article exists
     412   No newsgroup selected
     420   Current article number is invalid

   Parameters
     number   Requested article number
     n   Returned article number
     message-id   Article message-id

6.2.4.2. Description

   The STAT command behaves identically to the ARTICLE command except
   that, if the article exists, it is NOT presented to the client and
   the response code is 223 instead of 220. Note that the response is
   NOT multi-line.

   This command allows the client to determine whether an article exists
   and, in the second and third forms, what its message-id is, without
   having to process an arbitrary amount of text.

6.2.4.3. Examples

   Example of STAT on an existing article (explicitly not using an
   article number):

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] STAT
      [S] 223 3000234 <45223423@example.com>

   Example of STAT on an existing article by message-id:

      [C] STAT <45223423@example.com>
      [S] 223 0 <45223423@example.com>

   Example of STAT on an article not on the server by message-id:

      [C] STAT <i.am.not.there@example.com>
      [S] 430 No Such Article Found

   Example of STAT on an article not in the server by number:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] STAT 300256
      [S] 423 No article with that number

Example of STAT on an article by number when no newsgroup was
selected first:

    [Assumes currently selected newsgroup is invalid.]
    [C] STAT 300256
    [S] 412 No newsgroup selected

Example of STAT on an article when the currently selected newsgroup
is empty:

    [C] GROUP example.empty.newsgroup
    [S] 211 0 0 0 example.empty.newsgroup
    [C] STAT
    [S] 420 No current article selected

Example of STAT by message-id on a server that sometimes reports the
actual article number:

    [C] GROUP misc.test
    [S] 211 1234 3000234 3002322 misc.test
    [C] STAT
    [S] 223 3000234 <45223423@example.com>
    [C] STAT <45223423@example.com>
    [S] 223 0 <45223423@example.com>
    [C] STAT <45223423@example.com>
    [S] 223 3000234 <45223423@example.com>
    [C] GROUP example.empty.newsgroup
    [S] 211 0 0 0 example.empty.newsgroup
    [C] STAT <45223423@example.com>
    [S] 223 0 <45223423@example.com>
    [C] GROUP alt.crossposts
    [S] 211 9999 111111 222222 alt.crossposts
    [C] STAT <45223423@example.com>
    [S] 223 123456 <45223423@example.com>
    [C] STAT
    [S] 223 111111 <23894720@example.com>

The first STAT command establishes the identity of an article in the
group. The second and third show that the server may, but need not,
give the article number when the message-id is specified. The fourth
STAT command shows that zero must be specified if the article isn't
in the currently selected newsgroup. The fifth shows that the
number, if provided, must be that relating to the currently selected
newsgroup. The last one shows that the current article number is
still not changed by the use of STAT with a message-id even if it
returns an article number.

6.3. Article Posting

   Article posting is done in one of two ways: individual article
   posting from news-reading clients using POST, and article transfer
   from other news servers using IHAVE.

6.3.1. POST

6.3.1.1. Usage

   Indicating capability: POST

   This command MUST NOT be pipelined.

   Syntax
     POST

   Responses

   Initial responses
     340   Send article to be posted
     440   Posting not permitted

   Subsequent responses
     240   Article received OK
     441   Posting failed

6.3.1.2. Description

   If posting is allowed, a 340 response MUST be returned to indicate
   that the article to be posted should be sent. If posting is
   prohibited for some installation-dependent reason, a 440 response
   MUST be returned.

   If posting is permitted, the article MUST be in the format specified
   in Section 3.6 and MUST be sent by the client to the server as a
   multi-line data block (see Section 3.1.1). Thus a single dot (".")
   on a line indicates the end of the text, and lines starting with a
   dot in the original text have that dot doubled during transmission.

   Following the presentation of the termination sequence by the client,
   the server MUST return a response indicating success or failure of
   the article transfer. Note that response codes 340 and 440 are used
   in direct response to the POST command while 240 and 441 are returned
   after the article is sent.

A response of 240 SHOULD indicate that, barring unforeseen server
errors, the posted article will be made available on the server
and/or transferred to other servers, as appropriate, possibly
following further processing. In other words, articles not wanted by
the server SHOULD be rejected with a 441 response, rather than being
accepted and then discarded silently. However, the client SHOULD NOT
assume that the article has been successfully transferred unless it
receives an affirmative response from the server and SHOULD NOT
assume that it is being made available to other clients without
explicitly checking (for example, using the STAT command).

If the session is interrupted before the response is received, it is
possible that an affirmative response was sent but has been lost.
Therefore, in any subsequent session, the client SHOULD either check
whether the article was successfully posted before resending or
ensure that the server will allocate the same message-id to the new
attempt (see Appendix A.2). The latter approach is preferred since
the article might not have been made available for reading yet (for
example, it may have to go through a moderation process).

6.3.1.3. Examples

Example of a successful posting:

    [C] POST
    [S] 340 Input article; end with <CR-LF>.<CR-LF>
    [C] From: "Demo User" <nobody@example.net>
    [C] Newsgroups: misc.test
    [C] Subject: I am just a test article
    [C] Organization: An Example Net
    [C]
    [C] This is just a test article.
    [C] .
    [S] 240 Article received OK

Example of an unsuccessful posting:

    [C] POST
    [S] 340 Input article; end with <CR-LF>.<CR-LF>
    [C] From: "Demo User" <nobody@example.net>
    [C] Newsgroups: misc.test
    [C] Subject: I am just a test article
    [C] Organization: An Example Net
    [C]
    [C] This is just a test article.
    [C] .
    [S] 441 Posting failed

Example of an attempt to post when posting is not allowed:

    [Initial connection set-up completed.]
    [S] 201 NNTP Service Ready, posting prohibited
    [C] POST
    [S] 440 Posting not permitted

6.3.2. IHAVE

6.3.2.1. Usage

   Indicating capability: IHAVE

   This command MUST NOT be pipelined.

   Syntax
     IHAVE message-id

   Responses

   Initial responses
     335   Send article to be transferred
     435   Article not wanted
     436   Transfer not possible; try again later

   Subsequent responses
     235   Article transferred OK
     436   Transfer failed; try again later
     437   Transfer rejected; do not retry

   Parameters
     message-id   Article message-id

6.3.2.2. Description

   The IHAVE command informs the server that the client has an article
   with the specified message-id. If the server desires a copy of that
   article, a 335 response MUST be returned, instructing the client to
   send the entire article. If the server does not want the article
   (if, for example, the server already has a copy of it), a 435
   response MUST be returned, indicating that the article is not wanted.
   Finally, if the article isn't wanted immediately but the client
   should retry later if possible (if, for example, another client is in
   the process of sending the same article to the server), a 436
   response MUST be returned.

   If transmission of the article is requested, the client MUST send the
   entire article, including headers and body, to the server as a
   multi-line data block (see Section 3.1.1). Thus, a single dot (".")
   on a line indicates the end of the text, and lines starting with a
   dot in the original text have that dot doubled during transmission.
   The server MUST return a 235 response, indicating that the article
   was successfully transferred; a 436 response, indicating that the
   transfer failed but should be tried again later; or a 437 response,
   indicating that the article was rejected.

   This function differs from the POST command in that it is intended
   for use in transferring already-posted articles between hosts. It
   SHOULD NOT be used when the client is a personal news-reading
   program, since use of this command indicates that the article has
   already been posted at another site and is simply being forwarded
   from another host. However, despite this, the server MAY elect not
   to post or forward the article if, after further examination of the
   article, it deems it inappropriate to do so. Reasons for such
   subsequent rejection of an article may include problems such as
   inappropriate newsgroups or distributions, disc space limitations,
   article lengths, garbled headers, and the like. These are typically
   restrictions enforced by the server host's news software and not
   necessarily by the NNTP server itself.

   The client SHOULD NOT assume that the article has been successfully
   transferred unless it receives an affirmative response from the
   server. A lack of response (such as a dropped network connection or
   a network timeout) SHOULD be treated the same as a 436 response.

   Because some news server software may not immediately be able to
   determine whether an article is suitable for posting or forwarding,
   an NNTP server MAY acknowledge the successful transfer of the article
   (with a 235 response) but later silently discard it.

6.3.2.3. Examples

   Example of successfully sending an article to another site:

      [C] IHAVE <i.am.an.article.you.will.want@example.com>
      [S] 335 Send it; end with <CR-LF>.<CR-LF>
      [C] Path: pathost!demo!somewhere!not-for-mail
      [C] From: "Demo User" <nobody@example.com>
      [C] Newsgroups: misc.test
      [C] Subject: I am just a test article
      [C] Date: 6 Oct 1998 04:38:40 -0500
      [C] Organization: An Example Com, San Jose, CA
      [C] Message-ID: <i.am.an.article.you.will.want@example.com>
      [C]
      [C] This is just a test article.
      [C] .
      [S] 235 Article transferred OK

   Example of sending an article to another site that rejects it. Note
   that the message-id in the IHAVE command is not the same as the one
   in the article headers; while this is bad practice and SHOULD NOT be
   done, it is not forbidden.

      [C] IHAVE <i.am.an.article.you.will.want@example.com>
      [S] 335 Send it; end with <CR-LF>.<CR-LF>
      [C] Path: pathost!demo!somewhere!not-for-mail
      [C] From: "Demo User" <nobody@example.com>
      [C] Newsgroups: misc.test
      [C] Subject: I am just a test article
      [C] Date: 6 Oct 1998 04:38:40 -0500
      [C] Organization: An Example Com, San Jose, CA
      [C] Message-ID: <i.am.an.article.you.have@example.com>
      [C]
      [C] This is just a test article.
      [C] .
      [S] 437 Article rejected; don't send again

Example of sending an article to another site where the transfer
fails:

```
[C] IHAVE <i.am.an.article.you.will.want@example.com>
[S] 335 Send it; end with <CR-LF>.<CR-LF>
[C] Path: pathost!demo!somewhere!not-for-mail
[C] From: "Demo User" <nobody@example.com>
[C] Newsgroups: misc.test
[C] Subject: I am just a test article
[C] Date: 6 Oct 1998 04:38:40 -0500
[C] Organization: An Example Com, San Jose, CA
[C] Message-ID: <i.am.an.article.you.will.want@example.com>
[C]
[C] This is just a test article.
[C] .
[S] 436 Transfer failed
```

Example of sending an article to a site that already has it:

```
[C] IHAVE <i.am.an.article.you.have@example.com>
[S] 435 Duplicate
```

Example of sending an article to a site that requests that the
article be tried again later:

```
[C] IHAVE <i.am.an.article.you.defer@example.com>
[S] 436 Retry later
```

7. Information Commands

   This section lists other commands that may be used at any time
   between the beginning of a session and its termination. Using these
   commands does not alter any state information, but the response
   generated from their use may provide useful information to clients.

7.1. DATE

7.1.1. Usage

   Indicating capability: READER

   Syntax
     DATE

   Responses
     111 yyyymmddhhmmss   Server date and time

      Parameters
        yyyymmddhhmmss   Current UTC date and time on server

7.1.2. Description

   This command exists to help clients find out the current Coordinated
   Universal Time [TF.686-1] from the server's perspective. This
   command SHOULD NOT be used as a substitute for NTP [RFC1305] but to
   provide information that might be useful when using the NEWNEWS
   command (see Section 7.4).

   The DATE command MUST return a timestamp from the same clock as is
   used for determining article arrival and group creation times (see
   Section 6). This clock SHOULD be monotonic, and adjustments SHOULD
   be made by running it fast or slow compared to "real" time rather
   than by making sudden jumps. A system providing NNTP service SHOULD
   keep the system clock as accurate as possible, either with NTP or by
   some other method.

   The server MUST return a 111 response specifying the date and time on
   the server in the form yyyymmddhhmmss. This date and time is in
   Coordinated Universal Time.

7.1.3. Examples

      [C] DATE
      [S] 111 19990623135624

7.2. HELP

7.2.1. Usage

   This command is mandatory.

   Syntax
     HELP

   Responses
     100   Help text follows (multi-line)

7.2.2. Description

   This command provides a short summary of the commands that are
   understood by this implementation of the server. The help text will
   be presented as a multi-line data block following the 100 response
   code.

   This text is not guaranteed to be in any particular format (but must
   be UTF-8) and MUST NOT be used by clients as a replacement for the
   CAPABILITIES command described in Section 5.2.

7.2.3. Examples

      [C] HELP
      [S] 100 Help text follows
      [S] This is some help text. There is no specific
      [S] formatting requirement for this test, though
      [S] it is customary for it to list the valid commands
      [S] and give a brief definition of what they do.
      [S] .

7.3. NEWGROUPS

7.3.1. Usage

   Indicating capability: READER

   Syntax
     NEWGROUPS date time [GMT]

   Responses
     231   List of new newsgroups follows (multi-line)

   Parameters
     date   Date in yymmdd or yyyymmdd format
     time   Time in hhmmss format

7.3.2. Description

   This command returns a list of newsgroups created on the server since
   the specified date and time. The results are in the same format as
   the LIST ACTIVE command (see Section 7.6.3). However, they MAY
   include groups not available on the server (and so not returned by
   LIST ACTIVE) and MAY omit groups for which the creation date is not
   available.

   The date is specified as 6 or 8 digits in the format [xx]yymmdd,
   where xx is the first two digits of the year (19-99), yy is the last
   two digits of the year (00-99), mm is the month (01-12), and dd is
   the day of the month (01-31). Clients SHOULD specify all four digits
   of the year. If the first two digits of the year are not specified
   (this is supported only for backward compatibility), the year is to
   be taken from the current century if yy is smaller than or equal to
   the current year, and the previous century otherwise.

The time is specified as 6 digits in the format hhmmss, where hh is
the hours in the 24-hour clock (00-23), mm is the minutes (00-59),
and ss is the seconds (00-60, to allow for leap seconds). The token
"GMT" specifies that the date and time are given in Coordinated
Universal Time [TF.686-1]; if it is omitted, then the date and time
are specified in the server's local timezone. Note that there is no
way of using the protocol specified in this document to establish the
server's local timezone.

Note that an empty list is a possible valid response and indicates
that there are no new newsgroups since that date-time.

Clients SHOULD make all queries using Coordinated Universal Time
(i.e., by including the "GMT" argument) when possible.

7.3.3. Examples

Example where there are new groups:

    [C] NEWGROUPS 19990624 000000 GMT
    [S] 231 list of new newsgroups follows
    [S] alt.rfc-writers.recovery 4 1 y
    [S] tx.natives.recovery 89 56 y
    [S] .

Example where there are no new groups:

    [C] NEWGROUPS 19990624 000000 GMT
    [S] 231 list of new newsgroups follows
    [S] .

7.4. NEWNEWS

7.4.1. Usage

Indicating capability: NEWNEWS

Syntax
  NEWNEWS wildmat date time [GMT]

Responses
  230   List of new articles follows (multi-line)

Parameters
  wildmat   Newsgroups of interest
  date   Date in yymmdd or yyyymmdd format
  time   Time in hhmmss format

7.4.2. Description

   This command returns a list of message-ids of articles posted or
   received on the server, in the newsgroups whose names match the
   wildmat, since the specified date and time. One message-id is sent
   on each line; the order of the response has no specific significance
   and may vary from response to response in the same session. A
   message-id MAY appear more than once; if it does, it has the same
   meaning as if it appeared only once.

   Date and time are in the same format as the NEWGROUPS command (see
   Section 7.3).

   Note that an empty list is a possible valid response and indicates
   that there is currently no new news in the relevant groups.

   Clients SHOULD make all queries in Coordinated Universal Time (i.e.,
   by using the "GMT" argument) when possible.

7.4.3. Examples

   Example where there are new articles:

      [C] NEWNEWS news.*,sci.* 19990624 000000 GMT
      [S] 230 list of new articles by message-id follows
      [S] <i.am.a.new.article@example.com>
      [S] <i.am.another.new.article@example.com>
      [S] .

   Example where there are no new articles:

      [C] NEWNEWS alt.* 19990624 000000 GMT
      [S] 230 list of new articles by message-id follows
      [S] .

7.5. Time

   As described in Section 6, each article has an arrival timestamp.
   Each newsgroup also has a creation timestamp. These timestamps are
   used by the NEWNEWS and NEWGROUP commands to construct their
   responses.

   Clients can ensure that they do not have gaps in lists of articles or
   groups by using the DATE command in the following manner:

   First session:
      Issue DATE command and record result.
      Issue NEWNEWS command using a previously chosen timestamp.

   Subsequent sessions:
       Issue DATE command and hold result in temporary storage.
       Issue NEWNEWS command using timestamp saved from previous session.
       Overwrite saved timestamp with that currently in temporary
       storage.

   In order to allow for minor errors, clients MAY want to adjust the
   timestamp back by two or three minutes before using it in NEWNEWS.

7.5.1. Examples

   First session:

       [C] DATE
       [S] 111 20010203112233
       [C] NEWNEWS local.chat 20001231 235959 GMT
       [S] 230 list follows
       [S] <article.1@local.service>
       [S] <article.2@local.service>
       [S] <article.3@local.service>
       [S] .

   Second session (the client has subtracted 3 minutes from the
   timestamp returned previously):

       [C] DATE
       [S] 111 20010204003344
       [C] NEWNEWS local.chat 20010203 111933 GMT
       [S] 230 list follows
       [S] <article.3@local.service>
       [S] <article.4@local.service>
       [S] <article.5@local.service>
       [S] .

   Note how <article.3@local.service> arrived in the 3 minute gap and so
   is listed in both responses.

7.6. The LIST Commands

   The LIST family of commands all return information that is multi-line
   and that can, in general, be expected not to change during the
   session. Often the information is related to newsgroups, in which
   case the response has one line per newsgroup and a wildmat MAY be
   provided to restrict the groups for which information is returned.

   The set of available keywords (including those provided by
   extensions) is given in the capability list with capability label
   LIST.

7.6.1. LIST

7.6.1.1. Usage

   Indicating capability: LIST

   Syntax
     LIST [keyword [wildmat|argument]]

   Responses
     215   Information follows (multi-line)

   Parameters
     keyword   Information requested [1]
     argument   Specific to keyword
     wildmat   Groups of interest

   [1] If no keyword is provided, it defaults to ACTIVE.

7.6.1.2. Description

   The LIST command allows the server to provide blocks of information
   to the client. This information may be global or may be related to
   newsgroups; in the latter case, the information may be returned
   either for all groups or only for those matching a wildmat. Each
   block of information is represented by a different keyword. The
   command returns the specific information identified by the keyword.

   If the information is available, it is returned as a multi-line data
   block following the 215 response code. The format of the information
   depends on the keyword. The information MAY be affected by the
   additional argument, but the format MUST NOT be.

   If the information is based on newsgroups and the optional wildmat
   argument is specified, the response is limited to only the groups (if
   any) whose names match the wildmat and for which the information is
   available.

   Note that an empty list is a possible valid response; for a
   newsgroup-based keyword, it indicates that there are no groups
   meeting the above criteria.

   If the keyword is not recognised, or if an argument is specified and
   the keyword does not expect one, a 501 response code MUST BE
   returned. If the keyword is recognised but the server does not
   maintain the information, a 503 response code MUST BE returned.

   The LIST command MUST NOT change the visible state of the server in
   any way; that is, the behaviour of subsequent commands MUST NOT be
   affected by whether the LIST command was issued. For example, it
   MUST NOT make groups available that otherwise would not have been.

7.6.1.3. Examples

   Example of LIST with the ACTIVE keyword:

      [C] LIST ACTIVE
      [S] 215 list of newsgroups follows
      [S] misc.test 3002322 3000234 y
      [S] comp.risks 442001 441099 m
      [S] alt.rfc-writers.recovery 4 1 y
      [S] tx.natives.recovery 89 56 y
      [S] tx.natives.recovery.d 11 9 n
      [S] .

   Example of LIST with no keyword:

      [C] LIST
      [S] 215 list of newsgroups follows
      [S] misc.test 3002322 3000234 y
      [S] comp.risks 442001 441099 m
      [S] alt.rfc-writers.recovery 4 1 y
      [S] tx.natives.recovery 89 56 y
      [S] tx.natives.recovery.d 11 9 n
      [S] .

   The output is identical to that of the previous example.

   Example of LIST on a newsgroup-based keyword with and without
   wildmat:

      [C] LIST ACTIVE.TIMES
      [S] 215 information follows
      [S] misc.test 930445408 <creatme@isc.org>
      [S] alt.rfc-writers.recovery 930562309 <m@example.com>
      [S] tx.natives.recovery 930678923 <sob@academ.com>
      [S] .
      [C] LIST ACTIVE.TIMES tx.*
      [S] 215 information follows
      [S] tx.natives.recovery 930678923 <sob@academ.com>
      [S] .

   Example of LIST returning an error where the keyword is recognized
   but the software does not maintain this information:

      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] READER
      [S] LIST ACTIVE NEWSGROUPS ACTIVE.TIMES XTRA.DATA
      [S] .
      [C] LIST XTRA.DATA
      [S] 503 Data item not stored

   Example of LIST where the keyword is not recognised:

      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] READER
      [S] LIST ACTIVE NEWSGROUPS ACTIVE.TIMES XTRA.DATA
      [S] .
      [C] LIST DISTRIB.PATS
      [S] 501 Syntax Error

7.6.2. Standard LIST Keywords

   This specification defines the following LIST keywords:

   +--------------+--------------+----------------------------------+
   | Keyword | Definition | Status |
   +--------------+--------------+----------------------------------+
ACTIVE	Section 7.6.3	Mandatory if the READER capability
		is advertised
ACTIVE.TIMES	Section 7.6.4	Optional
DISTRIB.PATS	Section 7.6.5	Optional
HEADERS	Section 8.6	Mandatory if the HDR capability is
		advertised
NEWSGROUPS	Section 7.6.6	Mandatory if the READER capability
		is advertised
OVERVIEW.FMT	Section 8.4	Mandatory if the OVER capability
		is advertised
   +--------------+--------------+----------------------------------+

Where one of these LIST keywords is supported by a server, it MUST
have the meaning given in the relevant sub-section.

7.6.3. LIST ACTIVE

This keyword MUST be supported by servers advertising the READER
capability.

LIST ACTIVE returns a list of valid newsgroups and associated
information. If no wildmat is specified, the server MUST include
every group that the client is permitted to select with the GROUP
command (Section 6.1.1). Each line of this list consists of four
fields separated from each other by one or more spaces:

o The name of the newsgroup.
o The reported high water mark for the group.
o The reported low water mark for the group.
o The current status of the group on this server.

The reported high and low water marks are as described in the GROUP
command (see Section 6.1.1), but note that they are in the opposite
order to the 211 response to that command.

The status field is typically one of the following:

"y" Posting is permitted.

"n" Posting is not permitted.

"m" Postings will be forwarded to the newsgroup moderator.

The server SHOULD use these values when these meanings are required
and MUST NOT use them with any other meaning. Other values for the
status may exist; the definition of these other values and the
circumstances under which they are returned may be specified in an
extension or may be private to the server. A client SHOULD treat an
unrecognized status as giving no information.

The status of a newsgroup only indicates how posts to that newsgroup
are normally processed and is not necessarily customised to the
specific client. For example, if the current client is forbidden
from posting, then this will apply equally to groups with status "y".
Conversely, a client with special privileges (not defined by this
specification) might be able to post to a group with status "n".

For example:

```
[C] LIST ACTIVE
[S] 215 list of newsgroups follows
[S] misc.test 3002322 3000234 y
[S] comp.risks 442001 441099 m
[S] alt.rfc-writers.recovery 4 1 y
[S] tx.natives.recovery 89 56 y
[S] tx.natives.recovery.d 11 9 n
[S] .
```

or, on an implementation that includes leading zeroes:

```
[C] LIST ACTIVE
[S] 215 list of newsgroups follows
[S] misc.test 0003002322 0003000234 y
[S] comp.risks 0000442001 0000441099 m
[S] alt.rfc-writers.recovery 0000000004 0000000001 y
[S] tx.natives.recovery 0000000089 0000000056 y
[S] tx.natives.recovery.d 0000000011 0000000009 n
[S] .
```

The information is newsgroup based, and a wildmat MAY be specified,
in which case the response is limited to only the groups (if any)
whose names match the wildmat. For example:

```
[C] LIST ACTIVE *.recovery
[S] 215 list of newsgroups follows
[S] alt.rfc-writers.recovery 4 1 y
[S] tx.natives.recovery 89 56 y
[S] .
```

7.6.4. LIST ACTIVE.TIMES

This keyword is optional.

The active.times list is maintained by some NNTP servers to contain
information about who created a particular newsgroup and when. Each
line of this list consists of three fields separated from each other
by one or more spaces. The first field is the name of the newsgroup.
The second is the time when this group was created on this news
server, measured in seconds since the start of January 1, 1970. The
third is plain text intended to describe the entity that created the
newsgroup; it is often a mailbox as defined in RFC 2822 [RFC2822].
For example:

```
[C] LIST ACTIVE.TIMES
[S] 215 information follows
[S] misc.test 930445408 <creatme@isc.org>
[S] alt.rfc-writers.recovery 930562309 <m@example.com>
[S] tx.natives.recovery 930678923 <sob@academ.com>
[S] .
```

The list MAY omit newsgroups for which the information is unavailable
and MAY include groups not available on the server; in particular, it
MAY omit all groups created before the date and time of the oldest
entry. The client MUST NOT assume that the list is complete or that
it matches the list returned by the LIST ACTIVE command
(Section 7.6.3). The NEWGROUPS command (Section 7.3) may provide a
better way to access this information, and the results of the two
commands SHOULD be consistent except that, if the latter is invoked
with a date and time earlier than the oldest entry in active.times
list, its result may include extra groups.

The information is newsgroup based, and a wildmat MAY be specified,
in which case the response is limited to only the groups (if any)
whose names match the wildmat.

7.6.5. LIST DISTRIB.PATS

This keyword is optional.

The distrib.pats list is maintained by some NNTP servers to assist
clients to choose a value for the content of the Distribution header
of a news article being posted. Each line of this list consists of
three fields separated from each other by a colon (":"). The first
field is a weight, the second field is a wildmat (which may be a
simple newsgroup name), and the third field is a value for the
Distribution header content. For example:

```
[C] LIST DISTRIB.PATS
[S] 215 information follows
[S] 10:local.*:local
[S] 5:*:world
[S] 20:local.here.*:thissite
[S] .
```

The client MAY use this information to construct an appropriate
Distribution header given the name of a newsgroup. To do so, it
should determine the lines whose second field matches the newsgroup
name, select from among them the line with the highest weight (with 0
being the lowest), and use the value of the third field to construct
the Distribution header.

   The information is not newsgroup based, and an argument MUST NOT be
   specified.

7.6.6. LIST NEWSGROUPS

   This keyword MUST be supported by servers advertising the READER
   capability.

   The newsgroups list is maintained by NNTP servers to contain the name
   of each newsgroup that is available on the server and a short
   description about the purpose of the group. Each line of this list
   consists of two fields separated from each other by one or more space
   or TAB characters (the usual practice is a single TAB). The first
   field is the name of the newsgroup, and the second is a short
   description of the group. For example:

       [C] LIST NEWSGROUPS
       [S] 215 information follows
       [S] misc.test General Usenet testing
       [S] alt.rfc-writers.recovery RFC Writers Recovery
       [S] tx.natives.recovery Texas Natives Recovery
       [S] .

   The list MAY omit newsgroups for which the information is unavailable
   and MAY include groups not available on the server. The client MUST
   NOT assume that the list is complete or that it matches the list
   returned by LIST ACTIVE.

   The description SHOULD be in UTF-8. However, servers often obtain
   the information from external sources. These sources may have used
   different encodings (ones that use octets in the range 128 to 255 in
   some other manner) and, in that case, the server MAY pass it on
   unchanged. Therefore, clients MUST be prepared to receive such
   descriptions.

   The information is newsgroup based, and a wildmat MAY be specified,
   in which case the response is limited to only the groups (if any)
   whose names match the wildmat.

8. Article Field Access Commands

   This section lists commands that may be used to access specific
   article fields; that is, headers of articles and metadata about
   articles. These commands typically fetch data from an "overview
   database", which is a database of headers extracted from incoming
   articles plus metadata determined as the article arrives. Only
   certain fields are included in the database.

This section is based on the Overview/NOV database [ROBE1995]
developed by Geoff Collyer.

8.1. Article Metadata

Article "metadata" is data about articles that does not occur within
the article itself. Each metadata item has a name that MUST begin
with a colon (and that MUST NOT contain a colon elsewhere within it).
As with header names, metadata item names are not case sensitive.

When generating a metadata item, the server MUST compute it for
itself and MUST NOT trust any related value provided in the article.
(In particular, a Lines or Bytes header in the article MUST NOT be
assumed to specify the correct number of lines or bytes in the
article.) If the server has access to several non-identical copies
of an article, the value returned MUST be correct for any copy of
that article retrieved during the same session.

This specification defines two metadata items: ":bytes" and ":lines".
Other metadata items may be defined by extensions. The names of
metadata items defined by registered extensions MUST NOT begin with
":x-". To avoid the risk of a clash with a future registered
extension, the names of metadata items defined by private extensions
SHOULD begin with ":x-".

8.1.1. The :bytes Metadata Item

The :bytes metadata item for an article is a decimal integer. It
SHOULD equal the number of octets in the entire article: headers,
body, and separating empty line (counting a CRLF pair as two octets,
and excluding both the "." CRLF terminating the response and any "."
added for "dot-stuffing" purposes).

Note to client implementers: some existing servers return a value
different from that above. The commonest reasons for this are as
follows:

o Counting a CRLF pair as one octet.

o Including the "." character used for dot-stuffing in the number.

o Including the terminating "." CRLF in the number.

o Using one copy of an article for counting the octets but then
   returning another one that differs in some (permitted) manner.

Implementations should be prepared for such variation and MUST NOT
rely on the value being accurate.

8.1.2. The :lines Metadata Item

   The :lines metadata item for an article is a decimal integer. It
   MUST equal the number of lines in the article body (excluding the
   empty line separating headers and body). Equivalently, it is two
   less than the number of CRLF pairs that the BODY command would return
   for that article (the extra two are those following the response code
   and the termination octet).

8.2. Database Consistency

   The information stored in the overview database may change over time.
   If the database records the content or absence of a given field (that
   is, a header or metadata item) for all articles, it is said to be
   "consistent" for that field. If it records the content of a header
   for some articles but not for others that nevertheless included that
   header, or if it records a metadata item for some articles but not
   for others to which that item applies, it is said to be
   "inconsistent" for that field.

   The LIST OVERVIEW.FMT command SHOULD list all the fields for which
   the database is consistent at that moment. It MAY omit such fields
   (for example, if it is not known whether the database is consistent
   or inconsistent). It MUST NOT include fields for which the database
   is inconsistent or that are not stored in the database. Therefore,
   if a header appears in the LIST OVERVIEW.FMT output but not in the
   OVER output for a given article, that header does not appear in the
   article (similarly for metadata items).

   These rules assume that the fields being stored in the database
   remain constant for long periods of time, and therefore the database
   will be consistent. When the set of fields to be stored is changed,
   it will be inconsistent until either the database is rebuilt or the
   only articles remaining are those received since the change.
   Therefore, the output from LIST OVERVIEW.FMT needs to be altered
   twice. Firstly, before any fields stop being stored they MUST be
   removed from the output; then, when the database is once more known
   to be consistent, the new fields SHOULD be added to the output.

   If the HDR command uses the overview database rather than taking
   information directly from the articles, the same issues of
   consistency and inconsistency apply, and the LIST HEADERS command
   SHOULD take the same approach as the LIST OVERVIEW.FMT command in
   resolving them.

8.3. OVER

8.3.1. Usage

   Indicating capability: OVER

   Syntax
     OVER message-id
     OVER range
     OVER

   Responses

   First form (message-id specified)
     224   Overview information follows (multi-line)
     430   No article with that message-id

   Second form (range specified)
     224   Overview information follows (multi-line)
     412   No newsgroup selected
     423   No articles in that range

   Third form (current article number used)
     224   Overview information follows (multi-line)
     412   No newsgroup selected
     420   Current article number is invalid

   Parameters
     range   Number(s) of articles
     message-id   Message-id of article

8.3.2. Description

   The OVER command returns the contents of all the fields in the
   database for an article specified by message-id, or from a specified
   article or range of articles in the currently selected newsgroup.

   The message-id argument indicates a specific article. The range
   argument may be any of the following:

   o An article number.

   o An article number followed by a dash to indicate all following.

   o An article number followed by a dash followed by another article
      number.

   If neither is specified, the current article number is used.

Support for the first (message-id) form is optional. If it is
supported, the OVER capability line MUST include the argument
"MSGID". Otherwise, the capability line MUST NOT include this
argument, and the OVER command MUST return the generic response code
503 when this form is used.

If the information is available, it is returned as a multi-line data
block following the 224 response code and contains one line per
article, sorted in numerical order of article number. (Note that
unless the argument is a range including a dash, there will be
exactly one line in the data block.) Each line consists of a number
of fields separated by a TAB. A field may be empty (in which case
there will be two adjacent TABs), and a sequence of trailing TABs may
be omitted.

The first 8 fields MUST be the following, in order:

    "0" or article number (see below)
    Subject header content
    From header content
    Date header content
    Message-ID header content
    References header content
    :bytes metadata item
    :lines metadata item

If the article is specified by message-id (the first form of the
command), the article number MUST be replaced with zero, except that
if there is a currently selected newsgroup and the article is present
in that group, the server MAY use the article's number in that group.
(See the ARTICLE command (Section 6.2.1) and STAT examples
(Section 6.2.4.3) for more details.) In the other two forms of the
command, the article number MUST be returned.

Any subsequent fields are the contents of the other headers and
metadata held in the database.

For the five mandatory headers, the content of each field MUST be
based on the content of the header (that is, with the header name and
following colon and space removed). If the article does not contain
that header, or if the content is empty, the field MUST be empty.
For the two mandatory metadata items, the content of the field MUST
be just the value, with no other text.

For all subsequent fields that contain headers, the content MUST be
the entire header line other than the trailing CRLF. For all
subsequent fields that contain metadata, the field consists of the
metadata name, a single space, and then the value.

For all fields, the value is processed by first removing all CRLF
pairs (that is, undoing any folding and removing the terminating
CRLF) and then replacing each TAB with a single space. If there is
no such header in the article, no such metadata item, or no header or
item stored in the database for that article, the corresponding field
MUST be empty.

Note that, after unfolding, the characters NUL, LF, and CR cannot
occur in the header of an article offered by a conformant server.
Nevertheless, servers SHOULD check for these characters and replace
each one by a single space (so that, for example, CR LF LF TAB will
become two spaces, since the CR and first LF will be removed by the
unfolding process). This will encourage robustness in the face of
non-conforming data; it is also possible that future versions of this
specification could permit these characters to appear in articles.

The server SHOULD NOT produce output for articles that no longer
exist.

If the argument is a message-id and no such article exists, a 430
response MUST be returned. If the argument is a range or is omitted
and the currently selected newsgroup is invalid, a 412 response MUST
be returned. If the argument is a range and no articles in that
number range exist in the currently selected newsgroup, including the
case where the second number is less than the first one, a 423
response MUST be returned. If the argument is omitted and the
current article number is invalid, a 420 response MUST be returned.

8.3.3. Examples

In the first four examples, TAB has been replaced by vertical bar and
some lines have been folded for readability.

Example of a successful retrieval of overview information for an
article (explicitly not using an article number):

    [C] GROUP misc.test
    [S] 211 1234 3000234 3002322 misc.test
    [C] OVER
    [S] 224 Overview information follows
    [S] 3000234|I am just a test article|"Demo User"
        <nobody@example.com>|6 Oct 1998 04:38:40 -0500|
        <45223423@example.com>|<45454@example.net>|1234|
        17|Xref: news.example.com misc.test:3000363
    [S] .

Example of a successful retrieval of overview information for an
article by message-id:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] OVER MSGID
[S] LIST ACTIVE NEWSGROUPS OVERVIEW.FMT
[S] .
[C] OVER <45223423@example.com>
[S] 224 Overview information follows
[S] 0|I am just a test article|"Demo User"
    <nobody@example.com>|6 Oct 1998 04:38:40 -0500|
    <45223423@example.com>|<45454@example.net>|1234|
    17|Xref: news.example.com misc.test:3000363
[S] .
```

Note that the article number has been replaced by "0".

Example of the same commands on a system that does not implement
retrieval by message-id:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] OVER
[S] LIST ACTIVE NEWSGROUPS OVERVIEW.FMT
[S] .
[C] OVER <45223423@example.com>
[S] 503 Overview by message-id unsupported
```

Example of a successful retrieval of overview information for a range
of articles:

```
[C] GROUP misc.test
[S] 211 1234 3000234 3002322 misc.test
[C] OVER 3000234-3000240
[S] 224 Overview information follows
[S] 3000234|I am just a test article|"Demo User"
    <nobody@example.com>|6 Oct 1998 04:38:40 -0500|
    <45223423@example.com>|<45454@example.net>|1234|
    17|Xref: news.example.com misc.test:3000363
[S] 3000235|Another test article|nobody@nowhere.to
    (Demo User)|6 Oct 1998 04:38:45 -0500|<45223425@to.to>||
    4818|37||Distribution: fi
[S] 3000238|Re: I am just a test article|somebody@elsewhere.to|
    7 Oct 1998 11:38:40 +1200|<kfwer3v@elsewhere.to>|
    <45223423@to.to>|9234|51
[S] .
```

Note the missing "References" and Xref headers in the second line,
the missing trailing fields in the first and last lines, and that
there are only results for those articles that still exist.

Example of an unsuccessful retrieval of overview information on an
article by number:

```
[C] GROUP misc.test
[S] 211 1234 3000234 3002322 misc.test
[C] OVER 300256
[S] 423 No such article in this group
```

Example of an invalid range:

```
[C] GROUP misc.test
[S] 211 1234 3000234 3002322 misc.test
[C] OVER 3000444-3000222
[S] 423 Empty range
```

Example of an unsuccessful retrieval of overview information by
number because no newsgroup was selected first:

```
[Assumes currently selected newsgroup is invalid.]
[C] OVER
[S] 412 No newsgroup selected
```

Example of an attempt to retrieve information when the currently
selected newsgroup is empty:

    [C] GROUP example.empty.newsgroup
    [S] 211 0 0 0 example.empty.newsgroup
    [C] OVER
    [S] 420 No current article selected

8.4. LIST OVERVIEW.FMT

8.4.1. Usage

   Indicating capability: OVER

   Syntax
     LIST OVERVIEW.FMT

   Responses
     215   Information follows (multi-line)

8.4.2. Description

   See Section 7.6.1 for general requirements of the LIST command.

   The LIST OVERVIEW.FMT command returns a description of the fields in
   the database for which it is consistent (as described above). The
   information is returned as a multi-line data block following the 215
   response code. The information contains one line per field in the
   order in which they are returned by the OVER command; the first 7
   lines MUST (except for the case of letters) be exactly as follows:

       Subject:
       From:
       Date:
       Message-ID:
       References:
       :bytes
       :lines

   For compatibility with existing implementations, the last two lines
   MAY instead be:

       Bytes:
       Lines:

   even though they refer to metadata, not headers.

All subsequent lines MUST consist of either a header name followed by
":full", or the name of a piece of metadata.

There are no leading or trailing spaces in the output.

Note that the 7 fixed lines describe the 2nd to 8th fields of the
OVER output. The "full" suffix (which may use either uppercase,
lowercase, or a mix) is a reminder that the corresponding fields
include the header name.

This command MAY generate different results if it is used more than
once in a session.

If the OVER command is not implemented, the meaning of the output
from this command is not specified, but it must still meet the above
syntactic requirements.

8.4.3. Examples

Example of LIST OVERVIEW.FMT output corresponding to the example OVER
output above, in the preferred format:

    [C] LIST OVERVIEW.FMT
    [S] 215 Order of fields in overview database.
    [S] Subject:
    [S] From:
    [S] Date:
    [S] Message-ID:
    [S] References:
    [S] :bytes
    [S] :lines
    [S] Xref:full
    [S] Distribution:full
    [S] .

Example of LIST OVERVIEW.FMT output corresponding to the example OVER
output above, in the alternative format:

```
[C] LIST OVERVIEW.FMT
[S] 215 Order of fields in overview database.
[S] Subject:
[S] From:
[S] Date:
[S] Message-ID:
[S] References:
[S] Bytes:
[S] Lines:
[S] Xref:FULL
[S] Distribution:FULL
[S] .
```

8.5. HDR

8.5.1. Usage

Indicating capability: HDR

Syntax
  HDR field message-id
  HDR field range
  HDR field

Responses

First form (message-id specified)
  225   Headers follow (multi-line)
  430   No article with that message-id

Second form (range specified)
  225   Headers follow (multi-line)
  412   No newsgroup selected
  423   No articles in that range

Third form (current article number used)
  225   Headers follow (multi-line)
  412   No newsgroup selected
  420   Current article number is invalid

Parameters
  field   Name of field
  range   Number(s) of articles
  message-id   Message-id of article

8.5.2. Description

   The HDR command provides access to specific fields from an article
   specified by message-id, or from a specified article or range of
   articles in the currently selected newsgroup. It MAY take the
   information directly from the articles or from the overview database.
   In the case of headers, an implementation MAY restrict the use of
   this command to a specific list of headers or MAY allow it to be used
   with any header; it may behave differently when it is used with a
   message-id argument and when it is used with a range or no argument.

   The required field argument is the name of a header with the colon
   omitted (e.g., "subject") or the name of a metadata item including
   the leading colon (e.g., ":bytes"), and is case insensitive.

   The message-id argument indicates a specific article. The range
   argument may be any of the following:

   o An article number.

   o An article number followed by a dash to indicate all following.

   o An article number followed by a dash followed by another article
      number.

   If neither is specified, the current article number is used.

   If the information is available, it is returned as a multi-line data
   block following the 225 response code and contains one line for each
   article in the range that exists. (Note that unless the argument is
   a range including a dash, there will be exactly one line in the data
   block.) The line consists of the article number, a space, and then
   the contents of the field. In the case of a header, the header name,
   the colon, and the first space after the colon are all omitted.

   If the article is specified by message-id (the first form of the
   command), the article number MUST be replaced with zero, except that
   if there is a currently selected newsgroup and the article is present
   in that group, the server MAY use the article's number in that group.
   (See the ARTICLE command (Section 6.2.1) and STAT examples
   (Section 6.2.4.3) for more details.) In the other two forms of the
   command, the article number MUST be returned.

   Header contents are modified as follows: all CRLF pairs are removed,
   and then each TAB is replaced with a single space. (Note that this
   is the same transformation as is performed by the OVER command
   (Section 8.3.2), and the same comment concerning NUL, CR, and LF
   applies.)

Note the distinction between headers and metadata appearing to have
the same meaning. Headers are always taken unchanged from the
article; metadata are always calculated. For example, a request for
"Lines" returns the contents of the "Lines" header of the specified
articles, if any, no matter whether they accurately state the number
of lines, while a request for ":lines" returns the line count
metadata, which is always the actual number of lines irrespective of
what any header may state.

If the requested header is not present in the article, or if it is
present but empty, a line for that article is included in the output,
but the header content portion of the line is empty (the space after
the article number MAY be retained or omitted). If the header occurs
in a given article more than once, only the content of the first
occurrence is returned by HDR. If any article number in the provided
range does not exist in the group, no line for that article number is
included in the output.

If the second argument is a message-id and no such article exists, a
430 response MUST be returned. If the second argument is a range or
is omitted and the currently selected newsgroup is invalid, a 412
response MUST be returned. If the second argument is a range and no
articles in that number range exist in the currently selected
newsgroup, including the case where the second number is less than
the first one, a 423 response MUST be returned. If the second
argument is omitted and the current article number is invalid, a 420
response MUST be returned.

A server MAY only allow HDR commands for a limited set of fields; it
may behave differently in this respect for the first (message-id)
form from how it would for the other forms. If so, it MUST respond
with the generic 503 response to attempts to request other fields,
rather than return erroneous results, such as a successful empty
response.

If HDR uses the overview database and it is inconsistent for the
requested field, the server MAY return what results it can, or it MAY
respond with the generic 503 response. In the latter case, the field
MUST NOT appear in the output from LIST HEADERS.

8.5.3. Examples

   Example of a successful retrieval of subject lines from a range of
   articles (3000235 has no Subject header, and 3000236 is missing):

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] HDR Subject 3000234-3000238
      [S] 225 Headers follow
      [S] 3000234 I am just a test article
      [S] 3000235
      [S] 3000237 Re: I am just a test article
      [S] 3000238 Ditto
      [S] .

   Example of a successful retrieval of line counts from a range of
   articles:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] HDR :lines 3000234-3000238
      [S] 225 Headers follow
      [S] 3000234 42
      [S] 3000235 5
      [S] 3000237 11
      [S] 3000238 2378
      [S] .

   Example of a successful retrieval of the subject line from an article
   by message-id:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] HDR subject <i.am.a.test.article@example.com>
      [S] 225 Header information follows
      [S] 0 I am just a test article
      [S] .

   Example of a successful retrieval of the subject line from the
   current article:

      [C] GROUP misc.test
      [S] 211 1234 3000234 3002322 misc.test
      [C] HDR subject
      [S] 225 Header information follows
      [S] 3000234 I am just a test article
      [S] .

Example of an unsuccessful retrieval of a header from an article by
message-id:

    [C] HDR subject <i.am.not.there@example.com>
    [S] 430 No Such Article Found

Example of an unsuccessful retrieval of headers from articles by
number because no newsgroup was selected first:

    [Assumes currently selected newsgroup is invalid.]
    [C] HDR subject 300256-
    [S] 412 No newsgroup selected

Example of an unsuccessful retrieval of headers because the currently
selected newsgroup is empty:

    [C] GROUP example.empty.newsgroup
    [S] 211 0 0 0 example.empty.newsgroup
    [C] HDR subject 1-
    [S] 423 No articles in that range

Example of an unsuccessful retrieval of headers because the server
does not allow HDR commands for that header:

    [C] GROUP misc.test
    [S] 211 1234 3000234 3002322 misc.test
    [C] HDR Content-Type 3000234-3000238
    [S] 503 HDR not permitted on Content-Type

8.6. LIST HEADERS

8.6.1. Usage

   Indicating capability: HDR

   Syntax
     LIST HEADERS [MSGID|RANGE]

   Responses
     215   Field list follows (multi-line)

   Parameters
     MSGID   Requests list for access by message-id
     RANGE   Requests list for access by range

8.6.2. Description

   See Section 7.6.1 for general requirements of the LIST command.

   The LIST HEADERS command returns a list of fields that may be
   retrieved using the HDR command.

   The information is returned as a multi-line data block following the
   215 response code and contains one line for each field name
   (excluding the trailing colon for headers and including the leading
   colon for metadata items). If the implementation allows any header
   to be retrieved, it MUST NOT include any header names in the list but
   MUST include the special entry ":" (a single colon on its own). It
   MUST still explicitly list any metadata items that are available.
   The order of items in the list is not significant; the server need
   not even consistently return the same order. The list MAY be empty
   (though in this circumstance there is little point in providing the
   HDR command).

   An implementation that also supports the OVER command SHOULD at least
   permit all the headers and metadata items listed in the output from
   the LIST OVERVIEW.FMT command.

   If the server treats the first form of the HDR command (message-id
   specified) differently from the other two forms (range specified or
   current article number used) in respect of which headers or metadata
   items are available, then the following apply:

   o If the MSGID argument is specified, the results MUST be those
      available for the first form of the HDR command.

   o If the RANGE argument is specified, the results MUST be those
      available for the second and third forms of the HDR command.

   o If no argument is specified, the results MUST be those available
      in all forms of the HDR command (that is, it MUST only list those
      items listed in both the previous cases).

   If the server does not treat the various forms differently, then it
   MUST ignore any argument and always produce the same results (though
   not necessarily always in the same order).

   If the HDR command is not implemented, the meaning of the output from
   this command is not specified, but it must still meet the above
   syntactic requirements.

8.6.3. Examples

   Example of an implementation providing access to only a few headers:

      [C] LIST HEADERS
      [S] 215 headers supported:
      [S] Subject
      [S] Message-ID
      [S] Xref
      [S] .

   Example of an implementation providing access to the same fields as
   the first example in Section 8.4.3:

      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] READER
      [S] OVER
      [S] HDR
      [S] LIST ACTIVE NEWSGROUPS HEADERS OVERVIEW.FMT
      [S] .
      [C] LIST HEADERS
      [S] 215 headers and metadata items supported:
      [S] Date
      [S] Distribution
      [S] From
      [S] Message-ID
      [S] References
      [S] Subject
      [S] Xref
      [S] :bytes
      [S] :lines
      [S] .

   Example of an implementation providing access to all headers:

      [C] LIST HEADERS
      [S] 215 metadata items supported:
      [S] :
      [S] :lines
      [S] :bytes
      [S] :x-article-number
      [S] .

Example of an implementation distinguishing the first form of the HDR
command from the other two forms:

```
[C] LIST HEADERS RANGE
[S] 215 metadata items supported:
[S] :
[S] :lines
[S] :bytes
[S] .
[C] LIST HEADERS MSGID
[S] 215 headers and metadata items supported:
[S] Date
[S] Distribution
[S] From
[S] Message-ID
[S] References
[S] Subject
[S] :lines
[S] :bytes
[S] :x-article-number
[S] .
[C] LIST HEADERS
[S] 215 headers and metadata items supported:
[S] Date
[S] Distribution
[S] From
[S] Message-ID
[S] References
[S] Subject
[S] :lines
[S] :bytes
[S] .
```

Note that :x-article-number does not appear in the last set of
output.

9. Augmented BNF Syntax for NNTP

9.1. Introduction

Each of the following sections describes the syntax of a major
element of NNTP. This syntax extends and refines the descriptions
elsewhere in this specification and should be given precedence when
resolving apparent conflicts. Note that ABNF [RFC4234] strings are
case insensitive. Non-terminals used in several places are defined
in a separate section at the end.

   Between them, the non-terminals <command-line>, <command-datastream>,
   <command-continuation>, and <response> specify the text that flows
   between client and server. A consistent naming scheme is used in
   this document for the non-terminals relating to each command, and
   SHOULD be used by the specification of registered extensions.

   For each command, the sequence is as follows:

   o The client sends an instance of <command-line>; the syntax for the
      EXAMPLE command is <example-command>.

   o If the client is one that immediately streams data, it sends an
      instance of <command-datastream>; the syntax for the EXAMPLE
      command is <example-datastream>.

   o The server sends an instance of <response>.

      * The initial response line is independent of the command that
         generated it; if the 000 response has arguments, the syntax of
         the initial line is <response-000-content>.

      * If the response is multi-line, the initial line is followed by
         a <multi-line-data-block>. The syntax for the contents of this
         block after "dot-stuffing" has been removed is (for the 000
         response to the EXAMPLE command) <example-000-ml-content> and
         is an instance of <multi-line-response-content>.

   o While the latest response is one that indicates more data is
      required (in general, a 3xx response):

      * the client sends an instance of <command-continuation>; the
         syntax for the EXAMPLE continuation following a 333 response is
         <example-333-continuation>;

      * the server sends another instance of <response>, as above.

   (There are no commands in this specification that immediately stream
   data, but this non-terminal is defined for the convenience of
   extensions.)

9.2. Commands

   This syntax defines the non-terminal <command-line>, which represents
   what is sent from the client to the server (see section 3.1 for
   limits on lengths).

      command-line = command EOL
      command = X-command
      X-command = keyword *(WS token)

      command =/ article-command /
            body-command /
            capabilities-command /
            date-command /
            group-command /
            hdr-command /
            head-command /
            help-command /
            ihave-command /
            last-command /
            list-command /
            listgroup-command /
            mode-reader-command /
            newgroups-command /
            newnews-command /
            next-command /
            over-command /
            post-command /
            quit-command /
            stat-command

      article-command = "ARTICLE" [WS article-ref]
      body-command = "BODY" [WS article-ref]
      capabilities-command = "CAPABILITIES" [WS keyword]
      date-command = "DATE"
      group-command = "GROUP" [WS newsgroup-name]
      hdr-command = "HDR" WS header-meta-name [WS range-ref]
      head-command = "HEAD" [WS article-ref]
      help-command = "HELP"
      ihave-command = "IHAVE" WS message-id
      last-command = "LAST"
      list-command = "LIST" [WS list-arguments]
      listgroup-command = "LISTGROUP" [WS newsgroup-name [WS range]]
      mode-reader-command = "MODE" WS "READER"
      newgroups-command = "NEWGROUPS" WS date-time
      newnews-command = "NEWNEWS" WS wildmat WS date-time
      next-command = "NEXT"
      over-command = "OVER" [WS range-ref]

```
   post-command = "POST"
   quit-command = "QUIT"
   stat-command = "STAT" [WS article-ref]

   article-ref = article-number / message-id
   date = date2y / date4y
   date4y = 4DIGIT 2DIGIT 2DIGIT
   date2y = 2DIGIT 2DIGIT 2DIGIT
   date-time = date WS time [WS "GMT"]
   header-meta-name = header-name / metadata-name
   list-arguments = keyword [WS token]
   metadata-name = ":" 1*A-NOTCOLON
   range = article-number ["-" [article-number]]
   range-ref = range / message-id
   time = 2DIGIT 2DIGIT 2DIGIT
```

9.3. Command Continuation

   This syntax defines the further material sent by the client in the
   case of multi-stage commands and those that stream data.

```
   command-datastream = UNDEFINED
     ; not used, provided as a hook for extensions
   command-continuation = ihave-335-continuation /
         post-340-continuation

   ihave-335-continuation = encoded-article
   post-340-continuation = encoded-article

   encoded-article = multi-line-data-block
     ; after undoing the "dot-stuffing", this MUST match <article>
```

9.4. Responses

9.4.1. Generic Responses

   This syntax defines the non-terminal <response>, which represents the
   generic form of responses; that is, what is sent from the server to
   the client in response to a <command> or a <command-continuation>.

```
   response = simple-response / multi-line-response
   simple-response = initial-response-line
   multi-line-response = initial-response-line multi-line-data-block

   initial-response-line =
         initial-response-content [SP trailing-comment] CRLF
   initial-response-content = X-initial-response-content
   X-initial-response-content = 3DIGIT *(SP response-argument)
```

```
      response-argument = 1*A-CHAR
      trailing-comment = *U-CHAR
```

9.4.2. Initial Response Line Contents

   This syntax defines the specific initial response lines for the
   various commands in this specification (see section 3.1 for limits on
   lengths). Only those response codes with arguments are listed.

```
      initial-response-content =/ response-111-content /
            response-211-content /
            response-220-content /
            response-221-content /
            response-222-content /
            response-223-content /
            response-401-content

      response-111-content = "111" SP date4y time
      response-211-content = "211" 3(SP article-number) SP newsgroup-name
      response-220-content = "220" SP article-number SP message-id
      response-221-content = "221" SP article-number SP message-id
      response-222-content = "222" SP article-number SP message-id
      response-223-content = "223" SP article-number SP message-id
      response-401-content = "401" SP capability-label
```

9.4.3. Multi-line Response Contents

   This syntax defines the content of the various multi-line responses;
   more precisely, it defines the part of the response in the multi-line
   data block after any "dot-stuffing" has been undone. The numeric
   portion of each non-terminal name indicates the response code that is
   followed by this data.

```
      multi-line-response-content = article-220-ml-content /
            body-222-ml-content /
            capabilities-101-ml-content /
            hdr-225-ml-content /
            head-221-ml-content /
            help-100-ml-content /
            list-215-ml-content /
            listgroup-211-ml-content /
            newgroups-231-ml-content /
            newnews-230-ml-content /
            over-224-ml-content

      article-220-ml-content = article
      body-222-ml-content = body
      capabilities-101-ml-content = version-line CRLF
```

```
        *(capability-line CRLF)
   hdr-225-ml-content = *(article-number SP hdr-content CRLF)
   head-221-ml-content = 1*header
   help-100-ml-content = *(*U-CHAR CRLF)
   list-215-ml-content = list-content
   listgroup-211-ml-content = *(article-number CRLF)
   newgroups-231-ml-content = active-groups-list
   newnews-230-ml-content = *(message-id CRLF)
   over-224-ml-content = *(article-number over-content CRLF)

   active-groups-list = *(newsgroup-name SPA article-number
         SPA article-number SPA newsgroup-status CRLF)
   hdr-content = *S-NONTAB
   hdr-n-content = [(header-name ":" / metadata-name) SP hdr-content]
   list-content = body
   newsgroup-status = %x79 / %x6E / %x6D / private-status
   over-content = 1*6(TAB hdr-content) /
         7(TAB hdr-content) *(TAB hdr-n-content)
   private-status = token ; except the values in newsgroup-status
```

9.5. Capability Lines

   This syntax defines the generic form of a capability line in the
   capabilities list (see Section 3.3.1).

```
   capability-line = capability-entry
   capability-entry = X-capability-entry
   X-capability-entry = capability-label *(WS capability-argument)
   capability-label = keyword
   capability-argument = token
```

   This syntax defines the specific capability entries for the
   capabilities in this specification.

```
   capability-entry =/
         hdr-capability /
         ihave-capability /
         implementation-capability /
         list-capability /
         mode-reader-capability /
         newnews-capability /
         over-capability /
         post-capability /
         reader-capability

   hdr-capability = "HDR"
   ihave-capability = "IHAVE"
   implementation-capability = "IMPLEMENTATION" *(WS token)
```

```
      list-capability = "LIST" 1*(WS keyword)
      mode-reader-capability = "MODE-READER"
      newnews-capability = "NEWNEWS"
      over-capability = "OVER" [WS "MSGID"]
      post-capability = "POST"
      reader-capability = "READER"

      version-line = "VERSION" 1*(WS version-number)
      version-number = nzDIGIT *5DIGIT
```

9.6. LIST Variants

   This section defines more specifically the keywords for the LIST
   command and the syntax of the corresponding response contents.

```
      ; active
      list-arguments =/ "ACTIVE" [WS wildmat]
      list-content =/ list-active-content
      list-active-content = active-groups-list


      ; active.times
      list-arguments =/ "ACTIVE.TIMES" [WS wildmat]
      list-content =/ list-active-times-content
      list-active-times-content =
            *(newsgroup-name SPA 1*DIGIT SPA newsgroup-creator CRLF)
      newsgroup-creator = U-TEXT


      ; distrib.pats
      list-arguments =/ "DISTRIB.PATS"
      list-content =/ list-distrib-pats-content
      list-distrib-pats-content =
            *(1*DIGIT ":" wildmat ":" distribution CRLF)
      distribution = token


      ; headers
      list-arguments =/ "HEADERS" [WS ("MSGID" / "RANGE")]
      list-content =/ list-headers-content
      list-headers-content = *(header-meta-name CRLF) /
            *((metadata-name / ":") CRLF)


      ; newsgroups
      list-arguments =/ "NEWSGROUPS" [WS wildmat]
      list-content =/ list-newsgroups-content
      list-newsgroups-content =
```

```
        *(newsgroup-name WS newsgroup-description CRLF)
   newsgroup-description = S-TEXT


   ; overview.fmt
   list-arguments =/ "OVERVIEW.FMT"
   list-content =/ list-overview-fmt-content
   list-overview-fmt-content = "Subject:" CRLF
        "From:" CRLF
        "Date:" CRLF
        "Message-ID:" CRLF
        "References:" CRLF
        ( ":bytes" CRLF ":lines" / "Bytes:" CRLF "Lines:") CRLF
        *((header-name ":full" / metadata-name) CRLF)
```

9.7. Articles

This syntax defines the non-terminal <article>, which represents the
format of an article as described in Section 3.6.

```
   article = 1*header CRLF body
   header = header-name ":" [CRLF] SP header-content CRLF
   header-content = *(S-CHAR / [CRLF] WS)
   body = *(*B-CHAR CRLF)
```

9.8. General Non-terminals

These non-terminals are used at various places in the syntax and are
collected here for convenience. A few of these non-terminals are not
used in this specification but are provided for the consistency and
convenience of extension authors.

```
   multi-line-data-block = content-lines termination
   content-lines = *([content-text] CRLF)
   content-text = (".." / B-NONDOT) *B-CHAR
   termination = "." CRLF

   article-number = 1*16DIGIT
   header-name = 1*A-NOTCOLON
   keyword = ALPHA 2*(ALPHA / DIGIT / "." / "-")
   message-id = "<" 1*248A-NOTGT ">"
   newsgroup-name = 1*wildmat-exact
   token = 1*P-CHAR

   wildmat = wildmat-pattern *("," ["!"] wildmat-pattern)
   wildmat-pattern = 1*wildmat-item
   wildmat-item = wildmat-exact / wildmat-wild
   wildmat-exact = %x22-29 / %x2B / %x2D-3E / %x40-5A / %x5E-7E /
```

```
        UTF8-non-ascii  ; exclude ! * , ? [ \ ]
   wildmat-wild = "*" / "?"

   base64 = *(4base64-char) [base64-terminal]
   base64-char = UPPER / LOWER / DIGIT / "+" / "/"
   base64-terminal = 2base64-char "==" / 3base64-char "="

   ; Assorted special character sets
   ;   A- means based on US-ASCII, excluding controls and SP
   ;   P- means based on UTF-8, excluding controls and SP
   ;   U- means based on UTF-8, excluding NUL CR and LF
   ;   B- means based on bytes, excluding NUL CR and LF
   A-CHAR     = %x21-7E
   A-NOTCOLON = %x21-39 / %x3B-7E  ; exclude ":"
   A-NOTGT    = %x21-3D / %x3F-7E  ; exclude ">"
   P-CHAR     = A-CHAR / UTF8-non-ascii
   U-CHAR     = CTRL / TAB / SP / A-CHAR / UTF8-non-ascii
   U-NONTAB   = CTRL /      SP / A-CHAR / UTF8-non-ascii
   U-TEXT     = P-CHAR *U-CHAR
   B-CHAR     = CTRL / TAB / SP / %x21-FF
   B-NONDOT   = CTRL / TAB / SP / %x21-2D / %x2F-FF  ; exclude "."

   ALPHA = UPPER / LOWER   ; use only when case-insensitive
   CR = %x0D
   CRLF = CR LF
   CTRL = %x01-08 / %x0B-0C / %x0E-1F
   DIGIT = %x30-39
   nzDIGIT = %x31-39
   EOL = *(SP / TAB) CRLF
   LF = %x0A
   LOWER = %x61-7A
   SP = %x20
   SPA = 1*SP
   TAB = %x09
   UPPER = %x41-5A
   UTF8-non-ascii = UTF8-2 / UTF8-3 / UTF8-4
   UTF8-2    = %xC2-DF UTF8-tail
   UTF8-3    = %xE0 %xA0-BF UTF8-tail / %xE1-EC 2UTF8-tail /
               %xED %x80-9F UTF8-tail / %xEE-EF 2UTF8-tail
   UTF8-4    = %xF0 %x90-BF 2UTF8-tail / %xF1-F3 3UTF8-tail /
               %xF4 %x80-8F 2UTF8-tail
   UTF8-tail = %x80-BF
   WS = 1*(SP / TAB)
```

The following non-terminals require special consideration. They
represent situations where material SHOULD be restricted to UTF-8,
but implementations MUST be able to cope with other character
encodings. Therefore, there are two sets of definitions for them.

   Implementations MUST accept any content that meets this syntax:

     S-CHAR   = %x21-FF
     S-NONTAB = CTRL / SP / S-CHAR
     S-TEXT   = (CTRL / S-CHAR) *B-CHAR

   and MAY pass such content on unaltered.

   When generating new content or re-encoding existing content,
   implementations SHOULD conform to this syntax:

     S-CHAR   = P-CHAR
     S-NONTAB = U-NONTAB
     S-TEXT   = U-TEXT

9.9. Extensions and Validation

   The specification of a registered extension MUST include formal
   syntax that defines additional forms for the following non-terminals:

   command
      for each new command other than a variant of the LIST command -
      the syntax of each command MUST be compatible with the definition
      of <X-command>;

   command-datastream
      for each new command that immediately streams data;

   command-continuation
      for each new command that sends further material after the initial
      command line - the syntax of each continuation MUST be exactly
      what is sent to the server, including any escape mechanisms such
      as "dot-stuffing";

   initial-response-content
      for each new response code that has arguments - the syntax of each
      response MUST be compatible with the definition of <X-initial-
      response-content>;

   multi-line-response-content
      for each new response code that has a multi-line response - the
      syntax MUST show the response after the lines containing the
      response code and the terminating octet have been removed and any
      "dot-stuffing" undone;

   capability-entry
      for each new capability label - the syntax of each entry MUST be
      compatible with the definition of <X-capability-entry>;

   list-arguments
      for each new variant of the LIST command - the syntax of each
      entry MUST be compatible with the definition of <X-command>;

   list-content
      for each new variant of the LIST command - the syntax MUST show
      the response after the lines containing the 215 response code and
      the terminating octet have been removed and any "dot-stuffing"
      undone.

   The =/ notation of ABNF [RFC4234] and the naming conventions
   described in Section 9.1 SHOULD be used for this.

   When the syntax in this specification, or syntax based on it, is
   validated, it should be noted that:

   o the non-terminals <command-line>, <command-datastream>,
      <command-continuation>, <response>, and
      <multi-line-response-content> describe basic concepts of the
      protocol and are not referred to by any other rule;

   o the non-terminal <base64> is provided for the convenience of
      extension authors and is not referred to by any rule in this
      specification;

   o for the reasons given above, the non-terminals <S-CHAR>,
      <S-NONTAB>, and <S-TEXT> each have two definitions; and

   o the non-terminal <UNDEFINED> is deliberately not defined.

10. Internationalisation Considerations

10.1. Introduction and Historical Situation

   RFC 977 [RFC977] was written at a time when internationalisation was
   not seen as a significant issue. As such, it was written on the
   assumption that all communication would be in ASCII and use only a
   7-bit transport layer, although in practice just about all known
   implementations are 8-bit clean.

   Since then, Usenet and NNTP have spread throughout the world. In the
   absence of standards for handling the issues of language and
   character sets, countries, newsgroup hierarchies, and individuals
   have found a variety of solutions that work for them but that are not
   necessarily appropriate elsewhere. For example, some have adopted a
   default 8-bit character set appropriate to their needs (such as
   ISO/IEC 8859-1 in Western Europe or KOI-8 in Russia), others have
   used ASCII (either US-ASCII or national variants) in headers but

local 16-bit character sets in article bodies, and still others have
gone for a combination of MIME [RFC2045] and UTF-8. With the
increased use of MIME in email, it is becoming more common to find
NNTP articles containing MIME headers that identify the character set
of the body, but this is far from universal.

The resulting confusion does not help interoperability.

One point that has been generally accepted is that articles can
contain octets with the top bit set, and NNTP is only expected to
operate on 8-bit clean transport paths.

10.2. This Specification

Part of the role of this present specification is to eliminate this
confusion and promote interoperability as far as possible. At the
same time, it is necessary to accept the existence of the present
situation and not break existing implementations and arrangements
gratuitously, even if they are less than optimal. Therefore, the
current practice described above has been taken into consideration in
producing this specification.

This specification extends NNTP from US-ASCII [ANSI1986] to UTF-8
[RFC3629]. Except in the two areas discussed below, UTF-8 (which is
a superset of US-ASCII) is mandatory, and implementations MUST NOT
use any other encoding.

Firstly, the use of MIME for article headers and bodies is strongly
recommended. However, given widely divergent existing practices, an
attempt to require a particular encoding and tagging standard would
be premature at this time. Accordingly, this specification allows
the use of arbitrary 8-bit data in articles subject to the following
requirements and recommendations.

o The names of headers (e.g., "From" or "Subject") MUST be in
   US-ASCII.

o Header values SHOULD use US-ASCII or an encoding based on it, such
   as RFC 2047 [RFC2047], until such time as another approach has
   been standardised. At present, 8-bit encodings (including UTF-8)
   SHOULD NOT be used because they are likely to cause
   interoperability problems.

o The character set of article bodies SHOULD be indicated in the
   article headers, and this SHOULD be done in accordance with MIME.

o Where an article is obtained from an external source, an
   implementation MAY pass it on and derive data from it (such as the

response to the HDR command), even though the article or the data
does not meet the above requirements. Implementations MUST
transfer such articles and data correctly and unchanged; they MUST
NOT attempt to convert or re-encode the article or derived data.
(Nevertheless, a client or server MAY elect not to post or forward
the article if, after further examination of the article, it deems
it inappropriate to do so.)

This requirement affects the ARTICLE (Section 6.2.1), BODY
(Section 6.2.3), HDR (Section 8.5), HEAD (Section 6.2.2), IHAVE
(Section 6.3.2), OVER (Section 8.3), and POST (Section 6.3.1)
commands.

Secondly, the following requirements are placed on the newsgroups
list returned by the LIST NEWSGROUPS command (Section 7.6.6):

o Although this specification allows UTF-8 for newsgroup names, they
   SHOULD be restricted to US-ASCII until a successor to RFC 1036
   [RFC1036] standardises another approach. 8-bit encodings SHOULD
   NOT be used because they are likely to cause interoperability
   problems.

o The newsgroup description SHOULD be in US-ASCII or UTF-8 unless
   and until a successor to RFC 1036 standardises other encoding
   arrangements. 8-bit encodings other than UTF-8 SHOULD NOT be used
   because they are likely to cause interoperability problems.

o Implementations that obtain this data from an external source MUST
   handle it correctly even if it does not meet the above
   requirements. Implementations (in particular, clients) MUST
   handle such data correctly.

10.3. Outstanding Issues

While the primary use of NNTP is for transmitting articles that
conform to RFC 1036 (Netnews articles), it is also used for other
formats (see Appendix A). It is therefore most appropriate that
internationalisation issues related to article formats be addressed
in the relevant specifications. For Netnews articles, this is any
successor to RFC 1036. For email messages, it is RFC 2822 [RFC2822].

Of course, any article transmitted via NNTP needs to conform to this
specification as well.

Restricting newsgroup names to UTF-8 is not a complete solution. In
particular, when new newsgroup names are created or a user is asked
to enter a newsgroup name, some scheme of canonicalisation will need
to take place. This specification does not attempt to define that

canonicalization; further work is needed in this area, in conjunction
with the article format specifications. Until such specifications
are published, implementations SHOULD match newsgroup names octet by
octet. It is anticipated that any approved scheme will be applied
"at the edges", and therefore octet-by-octet comparison will continue
to apply to most, if not all, uses of newsgroup names in NNTP.

In the meantime, any implementation experimenting with UTF-8
newsgroup names is strongly cautioned that a future specification may
require that those names be canonicalized when used with NNTP in a
way that is not compatible with their experiments.

Since the primary use of NNTP is with Netnews, and since newsgroup
descriptions are normally distributed through specially formatted
articles, it is recommended that the internationalisation issues
related to them be addressed in any successor to RFC 1036.

11. IANA Considerations

This specification requires IANA to keep a registry of capability
labels. The initial contents of this registry are specified in
Section 3.3.4. As described in Section 3.3.3, labels beginning with
X are reserved for private use, while all other names are expected to
be associated with a specification in an RFC on the standards track
or defining an IESG-approved experimental protocol.

Different entries in the registry MUST use different capability
labels.

Different entries in the registry MUST NOT use the same command name.
For this purpose, variants distinguished by a second or subsequent
keyword (e.g., "LIST HEADERS" and "LIST OVERVIEW.FMT") count as
different commands. If there is a need for two extensions to use the
same command, a single harmonised specification MUST be registered.

12. Security Considerations

This section is meant to inform application developers, information
providers, and users of the security limitations in NNTP as described
by this document. The discussion does not include definitive
solutions to the problems revealed, though it does make some
suggestions for reducing security risks.

12.1. Personal and Proprietary Information

   NNTP, because it was created to distribute network news articles,
   will forward whatever information is stored in those articles.
   Specification of that information is outside this scope of this
   document, but it is likely that some personal and/or proprietary
   information is available in some of those articles. It is very
   important that designers and implementers provide informative
   warnings to users so that personal and/or proprietary information in
   material that is added automatically to articles (e.g., in headers)
   is not disclosed inadvertently. Additionally, effective and easily
   understood mechanisms to manage the distribution of news articles
   SHOULD be provided to NNTP Server administrators, so that they are
   able to report with confidence the likely spread of any particular
   set of news articles.

12.2. Abuse of Server Log Information

   A server is in the position to save session data about a user's
   requests that might identify their reading patterns or subjects of
   interest. This information is clearly confidential in nature, and
   its handling can be constrained by law in certain countries. People
   using this protocol to provide data are responsible for ensuring that
   such material is not distributed without the permission of any
   individuals that are identifiable by the published results.

12.3. Weak Authentication and Access Control

   There is no user-based or token-based authentication in the basic
   NNTP specification. Access is normally controlled by server
   configuration files. Those files specify access by using domain
   names or IP addresses. However, this specification does permit the
   creation of extensions to NNTP for such purposes; one such extension
   is [NNTP-AUTH]. While including such mechanisms is optional, doing
   so is strongly encouraged.

   Other mechanisms are also available. For example, a proxy server
   could be put in place that requires authentication before connecting
   via the proxy to the NNTP server.

12.4. DNS Spoofing

   Many existing NNTP implementations authorize incoming connections by
   checking the IP address of that connection against the IP addresses
   obtained via DNS lookups of lists of domain names given in local
   configuration files. Servers that use this type of authentication
   and clients that find a server by doing a DNS lookup of the server
   name rely very heavily on the Domain Name Service, and are thus

generally prone to security attacks based on the deliberate
misassociation of IP addresses and DNS names. Clients and servers
need to be cautious in assuming the continuing validity of an IP
number/DNS name association.

In particular, NNTP clients and servers SHOULD rely on their name
resolver for confirmation of an IP number/DNS name association,
rather than cache the result of previous host name lookups. Many
platforms already can cache host name lookups locally when
appropriate, and they SHOULD be configured to do so. It is proper
for these lookups to be cached, however, only when the TTL (Time To
Live) information reported by the name server makes it likely that
the cached information will remain useful.

If NNTP clients or servers cache the results of host name lookups in
order to achieve a performance improvement, they MUST observe the TTL
information reported by DNS. If NNTP clients or servers do not
observe this rule, they could be spoofed when a previously accessed
server's IP address changes. As network renumbering is expected to
become increasingly common, the possibility of this form of attack
will increase. Observing this requirement thus reduces this
potential security vulnerability.

This requirement also improves the load-balancing behaviour of
clients for replicated servers using the same DNS name and reduces
the likelihood of a user's experiencing failure in accessing sites
that use that strategy.

12.5. UTF-8 Issues

UTF-8 [RFC3629] permits only certain sequences of octets and
designates others as either malformed or "illegal". The Unicode
standard identifies a number of security issues related to illegal
sequences and forbids their generation by conforming implementations.

Implementations of this specification MUST NOT generate malformed or
illegal sequences and SHOULD detect them and take some appropriate
action. This could include the following:

o Generating a 501 response code.

o Replacing such sequences by the sequence %xEF.BF.BD, which encodes
   the "replacement character" U+FFFD.

o Closing the connection.

o Replacing such sequences by a "guessed" valid sequence (based on
   properties of the UTF-8 encoding).

In the last case, the implementation MUST ensure that any replacement
cannot be used to bypass validity or security checks. For example,
the illegal sequence %xC0.A0 is an over-long encoding for space
(%x20). If it is replaced by the correct encoding in a command line,
this needs to happen before the command line is parsed into
individual arguments. If the replacement came after parsing, it
would be possible to generate an argument with an embedded space,
which is forbidden. Use of the "replacement character" does not have
this problem, since it is permitted wherever non-US-ASCII characters
are. Implementations SHOULD use one of the first two solutions where
the general structure of the NNTP stream remains intact and SHOULD
close the connection if it is no longer possible to parse it
sensibly.

12.6. Caching of Capability Lists

   The CAPABILITIES command provides a capability list, which is
   information about the current capabilities of the server. Whenever
   there is a relevant change to the server state, the results of this
   command are required to change accordingly.

   In most situations, the capabilities list in a given server state
   will not change from session to session; for example, a given
   extension will be installed permanently on a server. Some clients
   may therefore wish to remember which extensions a server supports to
   avoid the delay of an additional command and response, particularly
   if they open multiple connections in the same session.

   However, information about extensions related to security and privacy
   MUST NOT be cached, since this could allow a variety of attacks.

   For example, consider a server that permits the use of cleartext
   passwords on links that are encrypted but not otherwise:

      [Initial connection set-up completed.]
      [S] 200 NNTP Service Ready, posting permitted
      [C] CAPABILITIES
      [S] 101 Capability list:
      [S] VERSION 2
      [S] READER
      [S] NEWNEWS
      [S] POST
      [S] XENCRYPT
      [S] LIST ACTIVE NEWSGROUPS
      [S] .
      [C] XENCRYPT
      [Client and server negotiate encryption on the link]
      [S] 283 Encrypted link established

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] NEWNEWS
[S] POST
[S] XSECRET
[S] LIST ACTIVE NEWSGROUPS
[S] .
[C] XSECRET fred flintstone
[S] 290 Password for fred accepted
```

If the client caches the last capabilities list, then on the next
session it will attempt to use XSECRET on an unencrypted link:

```
[Initial connection set-up completed.]
[S] 200 NNTP Service Ready, posting permitted
[C] XSECRET fred flintstone
[S] 483 Only permitted on secure links
```

This exposes the password to any eavesdropper. While the primary
cause of this is passing a secret without first checking the security
of the link, caching of capability lists can increase the risk.

Any security extension should include requirements to check the
security state of the link in a manner appropriate to that extension.

Caching should normally only be considered for anonymous clients that
do not use any security or privacy extensions and for which the time
required for an additional command and response is a noticeable
issue.

13. Acknowledgements

This document is the result of much effort by the present and past
members of the NNTP Working Group, chaired by Russ Allbery and Ned
Freed. It could not have been produced without them.

The author acknowledges the original authors of NNTP as documented in
RFC 977 [RFC977]: Brian Kantor and Phil Lapsey.

The author gratefully acknowledges the following:

o The work of the NNTP committee chaired by Eliot Lear. The
   organization of this document was influenced by the last available
   version from this working group. A special thanks to Eliot for
   generously providing the original machine-readable sources for
   that document.

o   The work of the DRUMS working group, specifically RFC 1869
    [RFC1869], that drove the original thinking that led to the
    CAPABILITIES command and the extensions mechanism detailed in this
    document.

o   The authors of RFC 2616 [RFC2616] for providing specific and
    relevant examples of security issues that should be considered for
    HTTP. Since many of the same considerations exist for NNTP, those
    examples that are relevant have been included here with some minor
    rewrites.

o   The comments and additional information provided by the following
    individuals in preparing one or more of the progenitors of this
    document:

       Russ Allbery <rra@stanford.edu>
       Wayne Davison <davison@armory.com>
       Chris Lewis <clewis@bnr.ca>
       Tom Limoncelli <tal@mars.superlink.net>
       Eric Schnoebelen <eric@egsner.cirr.com>
       Rich Salz <rsalz@osf.org>

This work was motivated by the work of various news reader authors
and news server authors, including those listed below:

Rick Adams
   Original author of the NNTP extensions to the RN news reader and
   last maintainer of Bnews.

Stan Barber
   Original author of the NNTP extensions to the news readers that
   are part of Bnews.

Geoff Collyer
   Original author of the OVERVIEW database proposal and one of the
   original authors of CNEWS.

Dan Curry
   Original author of the xvnews news reader.

Wayne Davison
   Author of the first threading extensions to the RN news reader
   (commonly called TRN).

Geoff Huston
   Original author of ANU NEWS.

14. References

14.1. Normative References

   [ANSI1986]   American National Standards Institute, "Coded Character
                 Set - 7-bit American Standard Code for Information
                 Interchange", ANSI X3.4, 1986.

   [RFC977]   Kantor, B. and P. Lapsley, "Network News Transfer
                 Protocol", RFC 977, February 1986.

   [RFC2045]   Freed, N. and N. Borenstein, "Multipurpose Internet
                 Mail Extensions (MIME) Part One: Format of Internet
                 Message Bodies", RFC 2045, November 1996.

   [RFC2047]   Moore, K., "MIME (Multipurpose Internet Mail
                 Extensions) Part Three: Message Header Extensions for
                 Non-ASCII Text", RFC 2047, November 1996.

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
                 Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC3629]   Yergeau, F., "UTF-8, a transformation format of ISO
                 10646", STD 63, RFC 3629, November 2003.

   [RFC4234]   Crocker, D., Ed. and P. Overell, "Augmented BNF for
                 Syntax Specifications: ABNF", RFC 4234, October 2005.

   [RFC4648]   Josefsson, S., "The Base16, Base32, and Base64 Data
                 Encodings", RFC 4648, October 2006.

   [TF.686-1]   International Telecommunications Union - Radio,
                 "Glossary, ITU-R Recommendation TF.686-1",
                 ITU-R Recommendation TF.686-1, October 1997.

14.2. Informative References

   [NNTP-AUTH]   Vinocur, J., Murchison, K., and C. Newman, "Network
                 News Transfer Protocol (NNTP) Extension for
                 Authentication",
                 RFC 4643, October 2006.

   [NNTP-STREAM] Vinocur, J. and K. Murchison, "Network News Transfer
                 Protocol (NNTP) Extension for Streaming Feeds",
                 RFC 4644, October 2006.

   [NNTP-TLS]   Murchison, K., Vinocur, J., and C. Newman, "Using
                 Transport Layer Security (TLS) with Network News
                 Transfer Protocol (NNTP)", RFC 4642, October 2006.

   [RFC1036]   Horton, M. and R. Adams, "Standard for interchange of
                 USENET messages", RFC 1036, December 1987.

   [RFC1305]   Mills, D., "Network Time Protocol (Version 3)
                 Specification, Implementation and Analysis", RFC 1305,
                 March 1992.

   [RFC1869]   Klensin, J., Freed, N., Rose, M., Stefferud, E., and D.
                 Crocker, "SMTP Service Extensions", STD 10, RFC 1869,
                 November 1995.

   [RFC2616]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
                 Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
                 Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

   [RFC2629]   Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
                 June 1999.

   [RFC2822]   Resnick, P., "Internet Message Format", RFC 2822, April
                 2001.

   [RFC2980]   Barber, S., "Common NNTP Extensions", RFC 2980, October
                 2000.

   [ROBE1995]   Robertson, R., "FAQ: Overview database / NOV General
                 Information", January 1995.

                 There is no definitive copy of this document known to
                 the author. It was previously posted as the Usenet
                 article <news:nov-faq-1-930909720@agate.Berkeley.EDU>

   [SALZ1992]   Salz, R., "Manual Page for wildmat(3) from the INN 1.4
                 distribution, Revision 1.10", April 1992.

                 There is no definitive copy of this document known to
                 the author.

Appendix A. Interaction with Other Specifications

   NNTP is most often used for transferring articles that conform to
   RFC 1036 [RFC1036] (such articles are called "Netnews articles"
   here). It is also sometimes used for transferring email messages
   that conform to RFC 2822 [RFC2822] (such articles are called "email
   articles" here). In this situation, articles must conform both to
   this specification and to that other one; this appendix describes
   some relevant issues.

A.1. Header Folding

   NNTP allows a header line to be folded (by inserting a CRLF pair)
   before any space or TAB character.

   Both email and Netnews articles are required to have at least one
   octet other than space or TAB on each header line. Thus, folding can
   only happen at one point in each sequence of consecutive spaces or
   TABs. Netnews articles are further required to have the header name,
   colon, and following space all on the first line; folding may only
   happen beyond that space. Finally, some non-conforming software will
   remove trailing spaces and TABs from a line. Therefore, it might be
   inadvisable to fold a header after a space or TAB.

   For maximum safety, header lines SHOULD conform to the following
   syntax rather than to that in Section 9.7.

      header = header-name ":" SP [header-content] CRLF
      header-content = [WS] token *([CRLF] WS token)

A.2. Message-IDs

   Every article handled by an NNTP server MUST have a unique
   message-id. For the purposes of this specification, a message-id is
   an arbitrary opaque string that merely needs to meet certain
   syntactic requirements and is just a way to refer to the article.

   Because there is a significant risk that old articles will be
   reinjected into the global Usenet system, RFC 1036 [RFC1036] requires
   that message-ids are globally unique for all time.

   This specification states that message-ids are the same if and only
   if they consist of the same sequence of octets. Other specifications
   may define two different sequences as being equal because they are
   putting an interpretation on particular characters. RFC 2822
   [RFC2822] has a concept of "quoted" and "escaped" characters. It
   therefore considers the three message-ids:

```
<ab.cd@example.com>
<"ab.cd"@example.com>
<"ab.\cd"@example.com>
```

as being identical. Therefore, an NNTP implementation handing email
articles must ensure that only one of these three appears in the
protocol and that the other two are converted to it as and when
necessary, such as when a client checks the results of a NEWNEWS
command against an internal database of message-ids. Note that
RFC 1036 [RFC1036] never treats two different strings as being
identical. Its successor (as of the time of writing) restricts the
syntax of message-ids so that, whenever RFC 2822 would treat two
strings as equivalent, only one of them is valid (in the above
example, only the first string is valid).

This specification does not describe how the message-id of an article
is determined; it may be deduced from the contents of the article or
derived from some external source. If the server is also conforming
to another specification that contains a definition of message-id
compatible with this one, the server SHOULD use those message-ids. A
common approach, and one that SHOULD be used for email and Netnews
articles, is to extract the message-id from the contents of a header
with name "Message-ID". This may not be as simple as copying the
entire header contents; it may be necessary to strip off comments and
undo quoting, or to reduce "equivalent" message-ids to a canonical
form.

If an article is obtained through the IHAVE command, there will be a
message-id provided with the command. The server MAY either use it
or determine one from the article contents. However, whichever it
does, it SHOULD ensure that, if the IHAVE command is repeated with
the same argument and article, it will be recognized as a duplicate.

If an article does not contain a message-id that the server can
identify, it MUST synthesize one. This could, for example, be a
simple sequence number or be based on the date and time when the
article arrived. When email or Netnews articles are handled, a
Message-ID header SHOULD be added to ensure global consistency and
uniqueness.

Note that, because the message-id might not have been derived from
the Message-ID header in the article, the following example is
legitimate (though unusual):

```
[C] HEAD <45223423@example.com>
[S] 221 0 <45223423@example.com>
[S] Path: pathost!demo!whitehouse!not-for-mail
[S] Message-ID: <1234@example.net>
[S] From: "Demo User" <nobody@example.net>
[S] Newsgroups: misc.test
[S] Subject: I am just a test article
[S] Date: 6 Oct 1998 04:38:40 -0500
[S] Organization: An Example Net, Uncertain, Texas
[S] .
```

A.3. Article Posting

   As far as NNTP is concerned, the POST and IHAVE commands provide the
   same basic facilities in a slightly different way. However, they
   have rather different intentions.

   The IHAVE command is intended for transmitting conforming articles
   between a system of NNTP servers, with all articles perhaps also
   conforming to another specification (e.g., all articles are Netnews
   articles). It is expected that the client will already have done any
   necessary validation (or that it has in turn obtained the article
   from a third party that has done so); therefore, the contents SHOULD
   be left unchanged.

   In contrast, the POST command is intended for use when an end-user is
   injecting a newly created article into a such a system. The article
   being transferred might not be a conforming email or Netnews article,
   and the server is expected to validate it and, if necessary, to
   convert it to the right form for onward distribution. This is often
   done by a separate piece of software on the server installation; if
   so, the NNTP server SHOULD pass the incoming article to that software
   unaltered, making no attempt to filter characters, to fold or limit
   lines, or to process the incoming text otherwise.

   The POST command can fail in various ways, and clients should be
   prepared to re-send an article. When doing so, however, it is often
   important to ensure (as far as possible) that the same message-id is
   allocated to both attempts so that the server, or other servers, can
   recognize the two articles as duplicates. In the case of email or
   Netnews articles, therefore, the posted article SHOULD contain a
   header with the name "Message-ID", and the contents of this header
   SHOULD be identical on each attempt. The server SHOULD ensure that
   two POSTed articles with the same contents for this header are
   recognized as identical and that the same message-id is allocated,
   whether or not those contents are suitable for use as the message-id.

Appendix B. Summary of Commands

   This section contains a list of every command defined in this
   document, ordered by command name and by indicating capability.

                      Ordered by command name:

         +-------------------+----------------------+---------------+
         | Command | Indicating capability | Definition |
         +-------------------+----------------------+---------------+
         | ARTICLE | READER | Section 6.2.1 |
         | BODY | READER | Section 6.2.3 |
         | CAPABILITIES | mandatory | Section 5.2 |
         | DATE | READER | Section 7.1 |
         | GROUP | READER | Section 6.1.1 |
         | HDR | HDR | Section 8.5 |
         | HEAD | mandatory | Section 6.2.2 |
         | HELP | mandatory | Section 7.2 |
         | IHAVE | IHAVE | Section 6.3.2 |
         | LAST | READER | Section 6.1.3 |
         | LIST | LIST | Section 7.6.1 |
         | LIST ACTIVE.TIMES | LIST | Section 7.6.4 |
         | LIST ACTIVE | LIST | Section 7.6.3 |
         | LIST DISTRIB.PATS | LIST | Section 7.6.5 |
         | LIST HEADERS | HDR | Section 8.6 |
         | LIST NEWSGROUPS | LIST | Section 7.6.6 |
         | LIST OVERVIEW.FMT | OVER | Section 8.4 |
         | LISTGROUP | READER | Section 6.1.2 |
         | MODE READER | MODE-READER | Section 5.3 |
         | NEWGROUPS | READER | Section 7.3 |
         | NEWNEWS | NEWNEWS | Section 7.4 |
         | NEXT | READER | Section 6.1.4 |
         | OVER | OVER | Section 8.3 |
         | POST | POST | Section 6.3.1 |
         | QUIT | mandatory | Section 5.4 |
         | STAT | mandatory | Section 6.2.4 |
         +-------------------+----------------------+---------------+

Ordered by indicating capability:

```
+-------------------+----------------------+---------------+
| Command           | Indicating capability | Definition   |
+-------------------+----------------------+---------------+
| CAPABILITIES      | mandatory            | Section 5.2   |
| HEAD              | mandatory            | Section 6.2.2 |
| HELP              | mandatory            | Section 7.2   |
| QUIT              | mandatory            | Section 5.4   |
| STAT              | mandatory            | Section 6.2.4 |
| HDR               | HDR                  | Section 8.5   |
| LIST HEADERS      | HDR                  | Section 8.6   |
| IHAVE             | IHAVE                | Section 6.3.2 |
| LIST              | LIST                 | Section 7.6.1 |
| LIST ACTIVE       | LIST                 | Section 7.6.3 |
| LIST ACTIVE.TIMES | LIST                 | Section 7.6.4 |
| LIST DISTRIB.PATS | LIST                 | Section 7.6.5 |
| LIST NEWSGROUPS   | LIST                 | Section 7.6.6 |
| MODE READER       | MODE-READER          | Section 5.3   |
| NEWNEWS           | NEWNEWS              | Section 7.4   |
| OVER              | OVER                 | Section 8.3   |
| LIST OVERVIEW.FMT | OVER                 | Section 8.4   |
| POST              | POST                 | Section 6.3.1 |
| ARTICLE           | READER               | Section 6.2.1 |
| BODY              | READER               | Section 6.2.3 |
| DATE              | READER               | Section 7.1   |
| GROUP             | READER               | Section 6.1.1 |
| LAST              | READER               | Section 6.1.3 |
| LISTGROUP         | READER               | Section 6.1.2 |
| NEWGROUPS         | READER               | Section 7.3   |
| NEXT              | READER               | Section 6.1.4 |
+-------------------+----------------------+---------------+
```

Appendix C. Summary of Response Codes

   This section contains a list of every response code defined in this
   document and indicates whether it is multi-line, which commands can
   generate it, what arguments it has, and what its meaning is.

   Response code 100 (multi-line)
      Generated by: HELP
      Meaning: help text follows.

   Response code 101 (multi-line)
      Generated by: CAPABILITIES
      Meaning: capabilities list follows.

   Response code 111
      Generated by: DATE
      1 argument: yyyymmddhhmmss
      Meaning: server date and time.

   Response code 200
      Generated by: initial connection, MODE READER
      Meaning: service available, posting allowed.

   Response code 201
      Generated by: initial connection, MODE READER
      Meaning: service available, posting prohibited.

   Response code 205
      Generated by: QUIT
      Meaning: connection closing (the server immediately closes the
      connection).

   Response code 211
      The 211 response code has two completely different forms,
      depending on which command generated it:

         (not multi-line)
         Generated by: GROUP
         4 arguments: number low high group
         Meaning: group selected.

         (multi-line)
         Generated by: LISTGROUP
         4 arguments: number low high group
         Meaning: article numbers follow.

Response code 215 (multi-line)
   Generated by: LIST
   Meaning: information follows.

Response code 220 (multi-line)
   Generated by: ARTICLE
   2 arguments: n message-id
   Meaning: article follows.

Response code 221 (multi-line)
   Generated by: HEAD
   2 arguments: n message-id
   Meaning: article headers follow.

Response code 222 (multi-line)
   Generated by: BODY
   2 arguments: n message-id
   Meaning: article body follows.

Response code 223
   Generated by: LAST, NEXT, STAT
   2 arguments: n message-id
   Meaning: article exists and selected.

Response code 224 (multi-line)
   Generated by: OVER
   Meaning: overview information follows.

Response code 225 (multi-line)
   Generated by: HDR
   Meaning: headers follow.

Response code 230 (multi-line)
   Generated by: NEWNEWS
   Meaning: list of new articles follows.

Response code 231 (multi-line)
   Generated by: NEWGROUPS
   Meaning: list of new newsgroups follows.

Response code 235
   Generated by: IHAVE (second stage)
   Meaning: article transferred OK.

Response code 240
   Generated by: POST (second stage)
   Meaning: article received OK.

Response code 335
   Generated by: IHAVE (first stage)
   Meaning: send article to be transferred.

Response code 340
   Generated by: POST (first stage)
   Meaning: send article to be posted.

Response code 400
   Generic response and generated by initial connection
   Meaning: service not available or no longer available (the server
   immediately closes the connection).

Response code 401
   Generic response
   1 argument: capability-label
   Meaning: the server is in the wrong mode; the indicated capability
   should be used to change the mode.

Response code 403
   Generic response
   Meaning: internal fault or problem preventing action being taken.

Response code 411
   Generated by: GROUP, LISTGROUP
   Meaning: no such newsgroup.

Response code 412
   Generated by: ARTICLE, BODY, GROUP, HDR, HEAD, LAST, LISTGROUP,
   NEXT, OVER, STAT
   Meaning: no newsgroup selected.

Response code 420
   Generated by: ARTICLE, BODY, HDR, HEAD, LAST, NEXT, OVER, STAT
   Meaning: current article number is invalid.

Response code 421
   Generated by: NEXT
   Meaning: no next article in this group.

Response code 422
   Generated by: LAST
   Meaning: no previous article in this group.

Response code 423
   Generated by: ARTICLE, BODY, HDR, HEAD, OVER, STAT
   Meaning: no article with that number or in that range.

   Response code 430
      Generated by: ARTICLE, BODY, HDR, HEAD, OVER, STAT
      Meaning: no article with that message-id.

   Response code 435
      Generated by: IHAVE (first stage)
      Meaning: article not wanted.

   Response code 436
      Generated by: IHAVE (either stage)
      Meaning: transfer not possible (first stage) or failed (second
      stage); try again later.

   Response code 437
      Generated by: IHAVE (second stage)
      Meaning: transfer rejected; do not retry.

   Response code 440
      Generated by: POST (first stage)
      Meaning: posting not permitted.

   Response code 441
      Generated by: POST (second stage)
      Meaning: posting failed.

   Response code 480
      Generic response
      Meaning: command unavailable until the client has authenticated
      itself.

   Response code 483
      Generic response
      Meaning: command unavailable until suitable privacy has been
      arranged.

   Response code 500
      Generic response
      Meaning: unknown command.

   Response code 501
      Generic response
      Meaning: syntax error in command.

   Response code 502
      Generic response and generated by initial connection

      Meaning for the initial connection and the MODE READER command:
      service permanently unavailable (the server immediately closes the
      connection).

      Meaning for all other commands: command not permitted (and there
      is no way for the client to change this).

   Response code 503
      Generic response
      Meaning: feature not supported.

   Response code 504
      Generic response
      Meaning: error in base64-encoding [RFC4648] of an argument.

Appendix D. Changes from RFC 977

   In general every attempt has been made to ensure that the protocol
   specification in this document is compatible with the version
   specified in RFC 977 [RFC977] and the various facilities adopted from
   RFC 2980 [RFC2980]. However, there have been a number of changes,
   some compatible and some not.

   This appendix lists these changes. It is not guaranteed to be
   exhaustive or correct and MUST NOT be relied on.

   o A formal syntax specification (Section 9) has been added.

   o The default character set is changed from US-ASCII [ANSI1986] to
      UTF-8 [RFC3629] (note that US-ASCII is a subset of UTF-8). This
      matter is discussed further in Section 10.

   o All articles are required to have a message-id, eliminating the
      "<0>" placeholder used in RFC 977 in some responses.

   o The newsgroup name matching capabilities already documented in
      RFC 977 ("wildmats", Section 4) are clarified and extended. The
      new facilities (e.g., the use of commas and exclamation marks) are
      allowed wherever wildmats appear in the protocol.

   o Support for pipelining of commands (Section 3.5) is made
      mandatory.

   o The principles behind response codes (Section 3.2) have been
      tidied up. In particular:

      * the x8x response code family, formerly used for private
         extensions, is now reserved for authentication and privacy
         extensions;

      * the x9x response code family, formerly intended for debugging
         facilities, are now reserved for private extensions;

      * the 502 and 503 generic response codes (Section 3.2.1) have
         been redefined;

      * new 401, 403, 480, 483, and 504 generic response codes have
         been added.

   o The rules for article numbering (Section 6) have been clarified
      (also see Section 6.1.1.2).

   o The SLAVE command (which was ill-defined) is removed from the
      protocol.

   o Four-digit years are permitted in the NEWNEWS (Section 7.4) and
      NEWGROUPS (Section 7.3) commands (two-digit years are still
      permitted). The optional distribution parameter to these commands
      has been removed.

   o The LIST command (Section 7.6.1) is greatly extended; the original
      is available as LIST ACTIVE, while new variants include
      ACTIVE.TIMES, DISTRIB.PATS, and NEWSGROUPS. A new "m" status flag
      is added to the LIST ACTIVE response.

   o A new CAPABILITIES command (Section 5.2) allows clients to
      determine what facilities are supported by a server.

   o The DATE command (Section 7.1) is adopted from RFC 2980
      effectively unchanged.

   o The LISTGROUP command (Section 6.1.2) is adopted from RFC 2980.
      An optional range argument has been added, and the 211 initial
      response line now has the same format as the 211 response from the
      GROUP command.

   o The MODE READER command (Section 5.3) is adopted from RFC 2980 and
      its meaning and effects clarified.

   o The XHDR command in RFC 2980 has been formalised as the new HDR
      (Section 8.5) and LIST HEADERS (Section 8.6) commands.

   o The XOVER command in RFC 2980 has been formalised as the new OVER
      (Section 8.3) and LIST OVERVIEW.FMT (Section 8.4) commands. The
      former can be applied to a message-id as well as to a range.

   o The concept of article metadata (Section 8.1) has been formalised,
      allowing the Bytes and Lines pseudo-headers to be deprecated.

   Client authors should note in particular that lack of support for the
   CAPABILITIES command is a good indication that the server does not
   support this specification.

Author's Address

   Clive D.W. Feather
   THUS plc
   322 Regents Park Road
   London
   N3 2QQ
   United Kingdom

   Phone: +44 20 8495 6138
   Fax: +44 870 051 9937
   EMail: clive@demon.net
   URI: http://www.davros.org/

Intellectual Property

   The IETF takes no position regarding the validity or scope of any
   Intellectual Property Rights or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; nor does it represent that it has
   made any independent effort to identify any such rights. Information
   on the procedures with respect to rights in RFC documents can be
   found in BCP 78 and BCP 79.

   Copies of IPR disclosures made to the IETF Secretariat and any
   assurances of licenses to be made available, or the result of an
   attempt made to obtain a general license or permission for the use of
   such proprietary rights by implementers or users of this
   specification can be obtained from the IETF on-line IPR repository at
   http://www.ietf.org/ipr.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights that may cover technology that may be required to implement
   this standard. Please address the information to the IETF at ietf-
   ipr@ietf.org.