Network Working Group
Internet Draft

Network News Transfer Protocol
Version 2

A Protocol for the Stream-Based Transmission of News

Eliot Lear
IntelliGenetics, Inc.

September, 1991

Status of this Memo

A derivative of this preliminary draft document will be submitted to the RFC editor as a standards document. Distribution of this memo is unlimited. Please send comments to *ietf-nntp@turbo.bio.net*.

What this Document Contains

This draft contains a bunch of new ways for transport servers and clients to communicate with one another. Those interested in transport issues should comment on them without delay. This draft is mildly intertwined with a number of revisions going on in other areas, such as the IETF-SMTP and IETF-822 newsgroups.

Let me be the first to point out my poor grammar. Please send gramatical errors to lear@turbo.bio.net, so that I don’t embarrass myself too much. In addition, there are several formatting errors. Please point them out, as well.

What this Document Doesn’t Contain

This document does not specify additional news reader functionality, because we are considering splitting such functionality into a protocol of its own. No final conclusions have been made about whether this document will contain new reader functionality. Please send your comments to the mailing list on reader related issues, as well.

Table Of Contents

1 Introduction

NNTP[1] specifies a protocol for the distribution, inquiry, retrieval, and posting of netnews articles using a reliable stream-based transmission of news among the Internet community. When used as a news reader protocol, NNTP is designed so that news articles are stored in a central database allowing a subscriber to select only those articles he wishes to read. The netnews model provides for indexing, cross-referencing, and expiration of aged messages.[2]

First deployed on the Internet in 1986, NNTP has enjoyed success in no small part due to the public domain UNIX implementation of both server and client, written by Phil Lapsley, Erik Fair, and Brian Kantor, and supported to date of this publication by Stan Barber with help from the networking community.

NNTP is designed for efficient transmission of news articles over a reliable full duplex communication method. Although designed primarily with the Internet in mind, the original specification has been implemented for DECNET[3] and DATAKIT. While the purpose of this document is to specify an Internet standard for news transmission, the authors have gone to some pains to ensure the mechanism's portability into other environments (e.g. ISO).

Because NNTP is a transport protocol, the scope of this document will be limited strictly to issues of transport. Thus, in as much as content format does not involve the transport, it shall not be addressed herein, and the reader is referred to the appropriate specification for netnews message format, currently RFC-1036.

1.1 Reasons for Change

Since NNTP's introduction, the Internet has changed considerably, growing by several orders of magnitude. New demands of netnews and its transport protocol have brought upon the need for revision of the original document. Netnews on the Internet has grown to encompass over one thousand available interest groups. A variety of information including pictures, sounds, and specialized data such as gene sequence information can now be retrieved via netnews.

Authentication has been added, so that host addresses need not be the sole means to identify a partner in a communication. Binary capability has been added for a number of reasons-- use of alternate forms of character sets, and ease of distribution of binary files are two sample reasons. Batch transfer has been added to reduce the number of protocol turnarounds required, in order to improve performance over long delay connections.

---

[1] RFC 977, *Network News Transfer Protocol*, Kantor & Lapsley, UCSD & UCB, 1986.

[2] RFC 1036, *Standard for Interchange of USENET Messages*, M. Horton & R. Adams, ATT Bell Laboratories / CSC, 1987.

[3] DECNET is a trademark of Digital Equipment Corporation

Several discrepancies have been flushed out; date formats have been modified to use ISO 3307 format, helping the netnews community enter the twenty-first century; distributions have been redefined, and the description of the newsgroup list has been modified to be slightly less confusing.

1.2 Compatibility

Version 2 of the NNTP protocol is designed to be backward compatible with the first version. New functionality must be negotiated by the client. If a client receives an improper response to any of the commands new to this version, it should presume that the server is not running version 2. The server's default behavior is that of version 1.

1.3 Document Layout

This document is broken down into six sections:

Introduction,
Protocol Overview,
Commands and Responses,
State Diagrams,
Comprehensive list of Reply Codes,
Appendix.

1.3.1 Terminal and Non-Terminal Symbols

This document uses a particular format for commands, their parameters and values contained in responses. Any word in CAPITOL LETTERS is meant to be a terminal symbol. Any word in **boldface** is meant to be a non-terminal symbol. All non-terminal symbols are unique throughout this document. Thus **n** will be interpreted as the article number within a group. In addition [brackets] are used for optional arguments and bar (|) is used to indicate ‘‘either or’’. (Parentheses) will be used for grouping, and *italics* will be used to indicate semantic action. Ellipses (...) are used to indicate a repeating pattern.

1.4 Acknowledgments

This document is based on a draft written by Brian Kantor. It also contains wording based on C-News documentation provided by Geoff Collyer and Henry Spencer, and some suggestions from Theodore T'so. It was further developed by the Internet Engineering Task Force NNTP working group. Particular thanks go to Erik Fair, Jim Thompson, Rich Salz, and Jim Galvin.

2 Protocol Overview

Every NNTP session will involve each of the following steps, though possibly not in order:

Connection,
Greeting, exchange of capabilities and requirements,
Authentication (confirmation of the identity of both parties),
News transfer, and
Conclusion/Disconnect.

While authentication for a particular session may not be required, implementations should respond properly to requests for authentication, either by attempting to satisfy or denying the request. The greeting will consist of a banner transmitted by the server, followed by an exchange of negotiations. Once it is determined who is exchanging news, and how it is to be exchanged, the client will either offer articles to or request articles from the server, depending on the negotiated configuration of the connection.

2.1 Server to Server Exchanges

In the context of following discussion a client is defined as the program that initiates the NNTP connection.

NNTP is successful in large part because it eliminates transmission of duplicate articles. In past this has been accomplished through a simple lock-step IHAVE/SENDME protocol, using the USENET Message-ID component as a key. Version 2 of the protocol introduces several variations on this theme, meant to optimize protocol exchanges (turnarounds).

When a client is to distribute netnews articles to a server, it may do so in one of several ways, all of of which involve the server confirming that it does not already have the article stored. This type of negotiation is known as IHAVE/SENDME.

One modification of IHAVE/SENDME is designed to group transactions together. This method is known as BATCH. Under this scheme, the client sends an index of queued articles to the server, and the server replies with a subset of those articles that it wishes to receive. This method reduces the number of turnarounds required during the conversation, and is highly recommended when more than one article is queued for transmission.

A slightly more robust form of IHAVE/SENDME involves the client and server sharing an additional TCP connection. One channel is then used for IHAVE/SENDME, while the other is used to actually transfer the contents of the articles. This method has the benefit of being asynchronous, while also eliminating many protocol turnarounds.

The final method of server to server communications involves the use of the NEWNEWS command. When the client applies this method, it requests that the server send a list of articles received after a certain date. The client will then decide which articles it wishes to receive and requests them using the ARTICLE command.

2.2 News Reader to Server Exchanges

A set of commands are included in the NNTP specification specifically for the benefit of remote news readers. These commands allow for article selection and retrieval, and user information maintenance (e.g. .newsrc files). Traditionally, a news reader would retrieve a list of newsgroups and file numbers from the server to determine which newsgroups have new messages.

When contacted for news reading purposes, the NNTP server will need to retain some state during a session, with regard to referencing the current group and current article. The current group is the group referenced in the most recent GROUP command. The current article is initialized to the first article in a group, and is changed by most of the news reader commands.

2.4 Connection & Greeting

Clients will initiate a TCP connection on port 119 to the server with which they wish to exchange news. NNTP requires a reliable stream protocol, as there are no reliable integrity checks made on a message, once it is introduced into the network. Once the connection is established, the server will respond with a greeting, indicating that it is ready for service, or that it is unable to deal with the client. Because the server is in fact responding to the client, we list this interaction as a command called **Connect**.

2.5 Authentication

One of the additions to the NNTP protocol is the provision for authentication. The purpose of authentication in NNTP is to enable either end of the connection to independently establish and verify the identity of the remote end. In this manner, for example, articles in private newsgroups can be transmitted in either direction with some confidence that the other party is who it claims to be.

Either the server or the client process can initiate a request for authentication. The intent is to allow any form of authentication to occur at any time. In order for there to be interoperability, a table of authentication names is listed in the appendix of this document. It is expected that additional methods will be added through the normal standardization process.

The server requests authentication by transmitting a return code in response to a client command indicates that authentication is required. If the client responds improperly, the server may conclude that the server does not implement authentication. The client requests authentication by issuing the AUTHINFO command. If the server responds improperly, the client may conclude that no authentication exists.

If the server has requested authentication by issuing a response code to a client command, once authentication is completed the client should reissue that command.

2.5.1 Out of Band Authentication

By definition out of band authentication does not occur within NNTP. If either a server or a client wishes to participate in some form of out of band authentication, they must do so strictly on a prearranged basis, so that the NNTP session does not linger forever, waiting for a nonexistent authentication source. In other words, an implementation -client or server- must never send an out of band authentication request without prior arrangements with the other end.

2.6 Commands

Commands consist of a command word followed by zero or more parameters separated by one or more space or tab characters, terminated by a CR-LF pair. Commands and parameters are to be treated as case insensitive, and may contain only one command. All commands are transmitted in NetworkASCII or NETASCII, with the high order bit cleared. Command lines should not contain more than 512 characters, including the CR-LF pair.

2.7 Responses

Responses take several forms, depending on their purpose, but they share a number of characteristics. Every response will begin with a three digit response code, indicated by using either the continuation mark ‘‘-’’ or a space to declare whether the response will be continued on the next line, and end each line with a CR-LF pair.

For example:

```
200 Eyewitness News server is ready and waiting!
```

or

```
200-Eyewitness News server is ready and waiting at
200 7:00PM.  Do you know where your armadillo is?
```

Except where noted, continued lines must have the same response code as the previous line. Under all circumstances, every such line must be terminated by a CR-LF pair.

2.7.1 Simple Responses

Simple response codes contain all information in the three digit response code. Other text may follow on the same line for informational purposes. However, programs should not attempt to parse anything other than the response code in this case.

For example:

```
500 Command not understood.
```

2.7.2 Status Responses

A status response contains useful information in the three digit code as well as the text following the response code. The exact format of each such response code will be specified later in this document.

For example:

```
211 188 14525 14731 misc.legal
```

2.7.3 Extended Responses

There are two forms of extended response. The first, textual response, consists of a three digit response code, some informational text, a CR-LF pair, and then zero or more CR-LF terminated lines of additional information, terminated by a line containing nothing more than a period (‘‘.’’). If a textual response is to be parsed, the receiving side should expect one or more space or tab characters at any point where a space between words is required.

For all examples lines beginning with ‘‘C:’’ originate from the client, lines beginning with ‘‘S:’’ are sent from the server.

For example:

```
C:  OPTION FORMAT=UNIX-IMAGE
S:  104 FORMAT=UNIX-IMAGE
C:  IHAVE
S:  Send list of Message-IDs
C:  <9104181633.AA08820@msw.usc.edu> 24354
C:  <1991Apr19.182734.20692@athena.cs.uga.edu>
C:  .
```

The second form of extended response is byte stream transfer. In certain cases, when the client and server agree on either BINARY or IMAGE transfer, an extended response will involve an eight bit transfer of a certain number of octets. When this document uses the terms ‘‘byte’’ or ‘‘bytecount’’ it refers to octets.

For example:

```
C:  OPTION FORMAT=BINARY
S:  104 FORMAT=BINARY
C:  ARTICLE <9104181633.AA08820@msw.usc.edu>
S:  224 0 4249 <9104181633.AA08820@msw.usc.edu>
Server sends 4249 bytes in binary mode.
```

When a byte count is specified it is of utmost importance that it be accurate, or the server and client will fall out of synchrony, which may cause connections to hang in a state where both ends are waiting for input.

2.7.4 Response Codes

Response codes indicate the results of actions taken by the server as requested by the client, or requests by the server for additional information. The first digit of a response broadly indicates the result of the previous command, as follows:

1xx - Informative message
2xx - Command ok
3xx - Command ok so far, send the rest of it.
4xx - Command was correct, but couldn't be performed for some reason.
5xx - Command unimplemented, or incorrect, or a serious program error occurred.

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
x5x - Authentication
x6x - Batch transfers
x8x - Nonstandard (private implementation) extensions
x9x - Debugging output

The exact response codes that should be expected from each command are detailed in the description of that command. In addition, certain response codes may be printed at any time that the client is expecting a response code. These include the following list:

100 Help text
19x Debug

In addition, there are several response codes that should be used by the server and recognized by the client, as appropriate:

200 server ready - posting allowed
201 server ready - no posting allowed
400 service discontinued
450 authentication required
500 command not recognized
501 command syntax error
503 program fault - command not performed

2.8 Names and Identifiers

Several options and commands require identifiers whose meaning must be shared by hosts on both ends of a connection. For example, the argument to the FORMAT command requires an **data-format** identifier; if this identifier is ‘‘UNIX-IMAGE’’, then both ends might not do LF/CRLF conversion. Similarly, authentication requires an identifier to indicate the method. In each instance where such identifiers are required, a table is provided in the appendix of this document, and may be supplemented or modified by future standards documents. In order to avoid interoperability problems, use of identifiers not listed in the appendix should be by prearrangement only. In addition, it is suggested that such identifiers begin with ‘‘X-’’. Usage of identifiers listed in the appendix should be in accord with the definition.

2.9 Data Representations

This version of NNTP allows information to be transmitted in three different formats. Additional formats may be specified later in separate documents.

2.9.1 TEXT

By default all articles and article information (such as lists of Message-IDs transmitted by either the client or the server) are sent as ASCII text; each such article must be sent as a set of lines separated by carriage return-line feed characters (CR-LF). Transmission of text is terminated by a single period (‘‘.’’) on an otherwise blank line terminated by CR-LF. If a client needs to transmit a data line containing a single period, it must send two periods, and the server must reduce them to one.

2.9.2 BINARY

Binary data is information that contains bytes that may have the high order bit set. Binary transfers between NNTP servers are transfers of a specified quantity of 8-bit octets in network byte order. No matter what the internal representation of this information (big endian/little endian[4] , etc), binary information must always be represented in the same form when being transferred.

---

[4] IEN-137, *On Holy Wars and a Plea for Peace*, Danny Cohen, USC-ISI 1980.

2.9.3 UNIX-IMAGE

Image transfers may contain either BINARY or TEXT information, and must be prearranged between two hosts with identical internal representations. UNIX-IMAGE allows hosts to not perform transformations of any kind on the data.

2.10 Canonical Message Format

To ensure interoperability, NNTP imposes a singular view of how any given article is to be presented over the network. Traditionally, this has meant that all information transmitted must be ASCII, and lines are to be terminated with carriage return-line feed ASCII characters (CR-LF). This is known as the canonical message format. Although a NNTP server is free to store information in any desired format, it must transmit articles in canonical message format. Thus, a UNIX system that stores messages with LF as the end of line character must translate a LF character to CR-LF before transmitting the article.

There is one exception to this rule. If two servers agree on the internal representation by using the FORMAT verb, they may forego any translation, and exchange messages in the selected internal format. Currently only one such internal representation is defined - UNIX-IMAGE.

With the introduction of BINARY article transfers, a new canonical message format will be required for binary articles. This document does not specify or place any restrictions on that format, other than to state that only articles adhere to that format may be transferred in binary mode. It is expected that a standard on the new canonical format for such messages will be published concurrently with this document.

3 NNTP Command Set

3.1 Overview

NNTP has the following command set:

> **Connect**, ARTICLE, AUTHINFO, DATE, HELP, NEWNEWS, OPTION, QUIT, SENDSYS
>
> BATCH, IHAVE, SENDME
>
> BODY, GROUP, HEAD, LAST, LIST, NEWGROUPS, NEXT, POST, STAT

The first set of commands implements general functionality required by all aspects of the protocol; the second set enables transport functionality, and the third set enables remote news reading and posting functionality. At the time of publication of this document, a separate Netnews User Protocol is being developed. As that protocol enters the standards process, news reader clients should use that protocol instead of NNTP. However, due to the installed base of news readers using NNTP, all servers must implement the third set of commands.

Commands are described in the order they appear in the above table.

3.2 Connect

3.2.1 Usage

Connect is a pseudo command, only because it elicits a response from the server.

3.2.2 Responses

200 server ready - posting allowed
201 server ready - no posting allowed
400 Service unavailable.
502 I’m not allowed to talk to you.

3.3 The ARTICLE Command

3.3.1 Usage : ARTICLE [**Message-ID | n**]

The ARTICLE command is used to fetch individual articles in their entirety by sending the headers, a blank line, and the text of the message. **Message-ID** is the contents of the Message-ID header of the requested article.[5]

If a Message-ID is specified, the article referred to by the Message-ID is transmitted as a textual response, or as otherwise negotiated as described previously in this document. If the article does not exist on the server, or is inaccessible for some reason, an error is returned. When used in this fashion, the ARTICLE command shall not alter the current article pointer or the current group pointer, because of semantic difficulties with articles posted to more than one newsgroup. In this context any responses that contain article numbers will return an article number of 0.

If no parameters are specified with the ARTICLE command, the current article is transmitted via textual response or an otherwise negotiated method.

If the parameter given to the ARTICLE command is a number **n**, the article associated with that number on the server is transmitted. The current article pointer is set by this command if a valid article number is specified. Valid article numbers are those listed in the result of a GROUP command.

3.3.2 Responses

220 **n Message-ID** article retrieved - head and body follow
224 **n Message-ID bytecount data-format** article follows
412 no newsgroup has been selected
420 no current article has been selected
423 no such article number in this group
430 no such article found
455 Permission denied.

Codes 220 through 225 require an article number. If the parameter to the ARTICLE command was a message id, then the article number will be returned as 0, indicating no change in the current article pointer.

---

[5] See RFC 1036, for a description of the Message ID.

3.4 The AUTHINFO Command

3.4.1 **Usage** : AUTHINFO (IHAVE|IWANT) **authtype** [**authdata**]

This command is used to exchange authentication credentials with the server. It is used to respond to a server's request for authentication in a 450 response code, or to initiate an authentication request.

The AUTHINFO command is quite general, so that just about any form of authentication exchange may occur within an NNTP session, at either side's request, with any number of exchanges occurring.

authtype is a form of authentication listed in the appendix. This document will only specify the inner workings of one of those mechanisms - SIMPLE, which is intended to be an example.

authdata may be present on the command line. If it is, it must be NETASCII data, and the whole line must not exceed the maximum line length for an NNTP command. Whether or not data is present, the server may require more information through a 350 or 351 response code, or it may send authentication information by preceding it with a 251 response code. In particular, if a binary transfer is desired, it is required that a bytecount be provided on the AUTHINFO line. In the absence of other arrangements, additional information shall be transmitted via textual exchange, ending with a line containing a single period (‘‘.’’).

Note that this standard does not specify how authentication data is to be internally structured. Rather the functionality exists to implement any of the many schemes in existence. How a particular scheme is used within NNTP is beyond the scope of this document, and quite possibly the subject of a separate standard. For example, currently there are efforts under way to produce a common programming interface to authentication technology. It is fully expected that CAT will be made available in NNTP some time in the future.

3.4.2 Responses

250 Authentication accepted
251 **authtype** [**authdata**]
350 Further authentication info needed
351 **authtype** Further authentication info needed
353 Challenge text follows, response requested.
451 **authtype** use **authtype** for authentication
452 no authentication information available
453 authentication rejected

550 Improper authentication sequence.

250 will be used when successful authentication has occurred. 251 is used to indicate the impending transmission of authentication data from the server to the client, with an optional authdata argument. 350 and 351 are to be used as explained previously. 452 and 453 are to be used to indicate authentication failure. Implementors may use 452 if they understand the command, but have not implemented authentication. Implementors must use 550 when they discover that the client has made an authentication scheme specific protocol error.

3.4.3 Example

A Simple Authentication Exchange:

```
S: 200 SomeUniversity.Edu NNTP server ready
S:-at Fri Jun 21 15:41:10 PDT 1991
C: IHAVE <1990Apr101243.lear@genbank.bio.net>
S: 451 SIMPLE Who are you?
C: AUTHINFO IHAVE SIMPLE user=webber,password=x7e37cn46
S: 250 Authentication Accepted, now where were we?
C: IHAVE <1990Apr101243.lear@genbank.bio.net>
```

B. Exchanging Additional Information

```
S:  200 Classified.NSA.Gov NNTP server ready.
C:  IHAVE <1990Apr101243.lear@genbank.bio.net>
S:  450 Further authentication needed.
C:  AUTHINFO IHAVE KERBEROS-5 200
S:  351 KERBEROS Ok.  Send me 200 bytes of KERBEROS data.
Client sends 200 bytes of data.
S:  250 Ok, I believe you.
```

C. Client initiated authentication.

```
S:  200 HARRY-MUDD.IM-LYING.Org ready at Jun 31 1919
C:  AUTHINFO IWANT MD-5
S:  251 MD-5 195 bytes of authentication data on its way
S:  Server sends binary data to client.
C:  IHAVE <1991HushData12374@ATM.CHASE-MANHATTEN.COM>
```

D. A Sample Disagreement

```
S:  200 Agita.Correctness.Org server ready now.
C:  AUTHINFO IWANT X1
S:  452 No authentication available
C:  QUIT
```

3.5 The DATE Command

3.5.1 **Usage** : DATE

DATE returns the GMT date and time as known to the server in ISO 3307 format YYYYMMDDhhmmss[.xxxxxx]. The timestamp is designed to be used in conjunction with the NEWGROUPS and NEWNEWS commands. This command should NOT be used to synchronize time between two computers - NTP[6] is the recommended method for clock synchronization on the Internet. This command merely requests the server's idea of the time, so that the client may keep track of new data since its last communication with this particular server. The client must not interpret the time information returned, but simply pass it back to the server at a later date.

3.5.2 Responses

111 YYYYMMDDhhmmss[.xxxxxx]

3.5.3 Example

```
C:  DATE
S:  111 19911231010203.025
```

[6] RFC-1129, *Internet time synchronization: The Network Time Protocol*, Dave Mills, University of Delaware 1989.

3.6 The HELP Command

3.6.1 **Usage** : HELP [**command**]

Provides a short summary of commands that are understood by this implementation of the server. The help text will be presented as a textual response, terminated by a single period on a line by itself.

If **command** is provided as an argument, the server may optionally give additional information about the command in question.

3.6.2 Responses

100 help text follows
490 help not available

3.7 The NEWNEWS Command

3.7.1 Usage : NEWNEWS **newsgroups date** [GMT] [**distribution**]

This command requests a list of Message-IDs that refer to associated messages that have been received after the specified date.

newsgroups is a comma-separated list of newsgroup patterns specifying the newsgroups the client is interested in receiving. The rules for matching are as follows:

> A pattern and a newsgroup match only if they are identical, except that the ‘‘*’’ character (asterisk) in a pattern shall mean one or more of any NETASCII character.
>
> If a pattern matches a newsgroup, !pattern forces a mismatch of that newsgroup; ie. it negates the match.
>
> A newsgroup matches a pattern list if, and only if, it matches at least one of the patterns and:
>
> > the newsgroup does not mismatch any of the patterns, or
> >
> > the longest matched pattern is longer than the longest mismatched pattern.

Note that order is not significant.

date is a date in ISO 3307 format in Greenwich Mean Time (GMT), as described in the DATE command. For backward compatibility, a server must accept the date and time in the form of yymmdd hhmmss ["GMT"].

distributions is an optional comma separated list of distributions, as listed by the ‘‘Distribution’’ header in a netnews article. As used here, distribution is in no way tied to newsgroup name. Only Message-IDs of articles exactly matching at least one distribution in the list may be returned to the client. If this parameter is not specified, all distributions will be examined and returned if they agree with the other NEWNEWS parameters. If the distribution ‘‘world’’ is used, only Message-IDs of articles with that distribution or with no distribution header may be returned. ‘‘World’’ is the default distribution.

3.7.2 Responses

230 list of new articles by Message-ID follows

3.7.3 Example

```
C:  NEWNEWS *,!comp.*,comp.sys 199106211530.10
S:  230 here comes a list of articles since June 6, 1991
S:  <1234@foo.com>
S:  <1991Jun22.002755.5674@uvm.edu>
S:  <1991Jun24.195403.15700@cbnewsj.att.com>
S:  .
```

3.8 The OPTION Command

3.8.1 **Usage** : OPTION **option**=**value** [**option**=**value**]...

The OPTION command is used to negotiate certain parameters for a connection. **option** is the option name, as listed in this document. **value** may be either the binary switch ON or OFF, an identifier, or a value if one is needed. There may be as many options on a command line as the line length limit allows. It is conceivable that certain sites or implementations may create new options for their own uses. It is suggested that such option names begin with ‘‘X-’’.

The server is to issue a reply that indicates whether an OPTION has been set or not. Initially, all options are in their default states. The server will respond with one reply per option request, and the reply will be of the same form of the request (**option**=**value**). A server rejects an option change by replying with the current value for the negotiated option. All clients and servers must implement the default values of the options listed in this document.

3.8.2 Responses

104 **option**=**value** [**option**=**value**]...

3.8.3 Example

```
C:  OPTION FORMAT=BINARY BATCH=200000
S:  104 FORMAT=BINARY BATCH=200000
```

In this example, the server accepts binary mode, accepts batch mode and sets the maximum batch size to 200000 bytes. and rejects the compression option.

The following set of options must be implemented on all NNTP servers conforming to version 2:

BATCH **bytecount**
FORMAT **data-format**
DATA-PATH **port-information**

3.8.4 The BATCH Option

3.8.4.1 Usage : OPTION BATCH=(**bytecount**|OFF)

If accepted this negotiation changes the behavior of IHAVE to allow for batches of articles to be shipped together. This option exists to reduce the number of protocol turnarounds. For information on how the IHAVE protocol is changed, refer to the IHAVE command in this document. <bytecount> represents the largest batch that the client will transmit, unless the batch contains a single article (otherwise a single article larger than the limit could not be transmitted using batch).

If a server replies with a smaller BATCH limit than the client has requested, the client must not exceed that upper limit.

This option may not be used in conjunction with the DATA-PATH option. Its default value is OFF.

3.8.4.2 Responses

104 BATCH=(**bytecount|**OFF)

<bytecount> in the reply code is the largest batch the server can receive. If the client specifies a batch number smaller than the server, the server may indicate to the client that it is willing to accept larger batches at this time. Of course the client need not increase the size of its batches to accommodate.

3.8.4.3 Example

```
C:  OPTION BATCH=200000
S:  104 BATCH=100000
```

In this example the client may send batches no greater than 100000 bytes.

3.8.5 The DATA-PATH Option

3.8.5.1 Usage : OPTION DATA-PATH=(**port-information|**OFF)

This option allows sites to send commands and data through two different channels, thus allowing different parts of the protocol to operate asynchronously. Before the client issues this option request, it must be prepared to accept connections from the server on a port to be specified that is outside the range legislated by the Internet Address Numbering Authority and on the same interface from which it initiated the initial connection request to the server. If the server accepts this option, it will immediately open a TCP connection to the specified port address on the client, and then send a reply to the option request. That connection will be known as the data connection. If the server is unable to connect to the client, the option remains off. If the option is to be turned off, before replying to the request the server will close the data connection. This option may not be used in conjunction with the BATCH option. See the IHAVE command for more information on how the DATA-PATH option is used.

It is conceivable that various authentication mechanisms will wish to encrypt port information, in order to keep eavesdroppers from gaining access to the server.

The default value of this option is OFF.

3.8.5.2 Responses

104 DATA-PATH=(**port-information**|OFF)

3.8.6 The FORMAT Option

3.8.6.1 Usage : OPTION FORMAT=**data-format**

The format option is used to negotiate a transmission format for netnews message. **data-format** refers to one of a list of different formats that may be optimal or required for certain messages. The default format is ‘‘CANONICAL-TEXT’’, Two other formats are defined in this document - UNIX-IMAGE and BINARY. Other data formats may be added through the normal standards process.

3.8.6.2 Responses

104 FORMAT=**data-format**

3.8.6.3 Example

```
C:  OPTION FORMAT=UNIX-IMAGE
S:  104 FORMAT=CANONICAL-TEXT
```

In the above example, the server is rejecting the UNIX-IMAGE format.

3.9 The QUIT Command

3.9.1 Usage : QUIT

The QUIT Command is the preferred method of closing a connection. Once the server replies, it will immediately close down the connection. If the DATA-PATH option is set, the other TCP channel will be closed, as well, after draining any lingering requests. See the IHAVE command for more information.

3.9.2 Responses

205 closing connection - goodbye!

3.10 The SENDSYS Command

3.10.1 **Usage** : SENDSYS **system-name**

The SENDSYS command requests that the server send information about one or all of the systems that it transmits information to. If <system name> is specified, the server will only transmit information about the name listed in the argument. If warranted, a reply will be transmitted via textual response, or as otherwise negotiated.

The format of the response is as follows:

system-name:**distributions**:**groups**

With one exception, the format for each of these lists may be found under the description of the NEWNEWS command. The exception is that new lines, tabs, and spaces may be inserted after any word.

3.10.2 Responses

207 SENDSYS response follows, terminated with a single period.
208 **n** SENDSYS response follows. Expect **n** bytes.
407 SENDSYS information not available.
507 **system-name**: no such system.

3.10.3 Example

```
C:  SENDSYS bionet
S:  207 Response follows.
S:  bionet:world,na,usa,ba,ca,bionet
S:      :all,!alt.sex.pictures
S:  .
```

3.11 The BATCH Command

3.11.1 Usage : BATCH [**bytecount**]

The BATCH command is used to initiate a transfer of one or more articles that have been queued for transmission via the IHAVE command. It may be used only if the BATCH option has been set for a session. Further, the client may transmit no batch larger than was negotiated with the OPTION command, unless the batch consists of a single article. Thus it is likely that a client might produce a long list of articles in a single IHAVE command, and then transmit multiple batches to the server. The **bytecount** parameter will be present if the data format requires it. (Currently that translates into UNIX-IMAGE and BINARY formats). Once the server replies that it can accept a batch, the client will send the batch of articles in the form of a netnews batch as described in RFC 1036 via textual response, unless otherwise negotiated.

3.11.2 Responses

363 Proceed with BATCH data.
263 Batch DATA received.
463 Inbound news batch garbled.

3.12 IHAVE

The IHAVE command is used to offer the server an article. The character of an IHAVE conversation depends on which options have been negotiated.

3.12.1 IHAVE With No Options

3.12.1.1 Usage : IHAVE **Message-ID**

If there are no options set, the argument will be a Message-ID of a single article being offered to the server. If the server does not have the article and wishes to receive it, it will reply with 335, requesting the article. At that point, the client shall send the article via textual response. After a successful transmission, the server will respond with 235, informing the client that it is OK to dequeue the article.

If the server already has the article, it will return 435, telling the client to NOT transmit the article, and to dequeue it for the server.

If, for some reason, the server does not know if it has the article, probably because some other server is transmitting it at that instant, then the server will respond with 438, requesting that the client ask the server about the article at some point in the future. Thus, if an article is being transmitted during another NNTP session, the server need not keep the connection idle.

3.12.1.2 Responses

235 article transferred ok
335 send article to be transferred. End with <CR-LF>.<CR-LF>
435 article not wanted - do not send it
436 transfer failed - try again later
437 article rejected - do not try again
438 ask me again in a little while

3.12.1.3 Example

A. Successful Exchange

```
C:  IHAVE <1991Jun24.195403.15700@cbnewsj.att.com>
S:  335 send article.  End with <CR-LF>.<CR-LF>
Client sends text article.
C:  .
S:  235 Article Transferred OK.
```

B. Server already has the article.

C: IHAVE <1991Jun24.195403.15700@cbnewsj.att.com>
S: 435 Article not wanted - do not send it.

C. Server doesn't know if it has the article

```
C:  IHAVE <1991Jun24.195403.15700@cbnewsj.att.com>
S:  438 Someone else is sending it - ask me again later.
```

3.12.2 IHAVE With BINARY or UNIX-IMAGE Format s

3.12.2.1 Usage : IHAVE **Message-ID bytecount**

This form of IHAVE is used to transmit articles in either BINARY or UNIX-IMAGE modes. Note that this form may not be used with BATCH or DATA-PATH commands set. **bytecount** is the number of bytes the server should expect, if it wants the article.

3.12.2.2 Responses

235 article transferred ok
332 send **bytecount** bytes
435 article not wanted - do not send it
436 transfer failed - try again later
437 article rejected - do not try again
438 ask me again in a little while

3.12.3 IHAVE With BATCH Option Set

3.12.3.1 Usage : IHAVE

If the BATCH option has been set, the client may offer as many articles as it has in its queue. Once the client issues the command, the server responds with a 361 reply code, requesting a list of Message-ID lines to be transferred. The client will then transmit the list of Message-ID lines via textual response. The server will then reply with the number of reply codes equal to the number of Message-IDs sent by the client, indicating whether the article should be sent, requeued, or dequeued altogether. Once the client receives the list of articles that the server wants, it will transmit them using the BATCH command.

3.12.3.2 Responses

361 Send list of Message-IDs
265 **Message-ID** Queue article for delivery with the batch command.
465 **Message-ID** article not wanted - do not queue it
466 **Message-ID** ask me again in a little while

3.12.3.3 Example

C: OPTION BATCH=100000
S: 104 BATCH=100000
C: IHAVE
S: 361 Send me your list of Message-IDs.
C: <1991Jun24.195403.15700@cbnewsj.att.com>
C: <1991Jul1.214801.287@genbank.bio.net>
C: <June.1.1991.15.14.01.987.pleasant@rutgers.edu>
C: .
S: 265-<1991Jun24.195403.15700@cbnewsj.att.com>
S: 265-<1991Jul1.214801.287@genbank.bio.net>
S: 466 <June.1.1991.15.14.01.987.pleasant@rutgers.edu>
C: BATCH
S: 363 Transmit a batch of articles

```
Client transmits two articles
C:  .
S:  263 Batch received, thanks!
```

3.12.4 IHAVE with the DATA-PATH Option Set

3.12.4.1 Usage : IHAVE

If the DATA-PATH option is set, a second TCP connection should already exist between the client and the server. After receiving an IHAVE command, the server will respond with a 335 message requesting that a stream of Message-IDs begin. The client will then send containing lines of Message-IDs. The server will asynchronously send Message-IDs that it is interested in receiving. The client will queue these messages for transmission on the alternate channel. The client will stop expecting messages from the server only after the server returns a 251 reply code when it has received the last message from the client.

When the client transmits a message to the server on the data channel it will first indicate the transmission format for the forthcoming message. If neither BINARY nor UNIX-IMAGE formats have been negotiated, the client will send one line to the server containing ‘‘TEXT’’ and then transmit the message as it would any textual response, ending each line with a CR-LF pair, and terminating the article with a sole period between CR-LF pairs. If the client is transmitting in BINARY or IMAGE mode, it will transmit a line containing the bytecount of the message, telling the server how many bytes following the CR-LF pair will be sent down the data stream. Once the article has been received, the server will send on the DATA channel 236 **Message-ID** received, indicating to the client that the server has received the article, and that the client may dequeue it from its list of articles to send.

It is likely that the offering of Message-IDs will terminate well before the last article has been transmitted. Until the server indicates that it desires no more articles, the client may not issue any further commands to the server.

If for some reason the data connection is reset before all articles have been transmitted, the client should requeue any articles in the transfer queue in the list of articles whose Message-IDs will be offered at the next transmission.

One area of exploration for this method will be load balancing of different client connections. If multiple clients offer the same article it should be possible for the server to request the article from the client with the best expected response. That client may be one that is currently not transmitting any article, for example.

3.12.4.2 Responses

Control Connection:

237 Leaving IHAVE mode on control channel.
331 Send me Message-IDs on this channel, articles on the other.
431 Problem with DATA port.
531 DATA-PATH option not set.

Data Connection:

236 **Message-ID** article transferred ok
436 **Message-ID** transfer failed - try again later
437 **Message-ID** article rejected - do not try again.

It should be noted that since the server can order its requests as it pleases, there is no need for the client to requeue an offering, so long as the connections remain intact.

3.12.4.3 Example

The control connection communication may look like the following:

```
C:  OPTION BINARY=ON DATA-PATH=56356
S:  104-BINARY=ON
Server opens connection to client on port 56356
S:  104 DATA-PATH=56356
C:  IHAVE
S:  331 Message-IDs on this channel, msgs on the other.
C:  <1991Jun24.195403.15700@cbnewsj.att.com>
C:  <1991Jul1.214801.287@genbank.bio.net>
C:  <1990Apr101243.lear@genbank.bio.net>
C:  <27.Jun.1991.651@groan.berkeley.edu>
S:  <1991Jun24.195403.15700@cbnewsj.att.com>
C:  .
S:  <27.Jun.1991.651@groan.berkeley.edu>
S:  237 Leaving IHAVE mode on Control channel.
```

The DATA Channel would look like the following:

```
C:  34746
Client sends 34,746 bytes of binary data
S:  236 <1991Jun24.195403.15700@cbnewsj.att.com> OK.
```

```
C:  TEXT
C:  Path: bionet!ucbvax!groan!weatherdroid
C:  From: droid@groan.berkeley.edu (The WeatherDroid)
C:  Newsgroups: ba.weather
C:  Subject: weather report for 1546, 27 June 1991
C:  Message-ID: <27.Jun.1991.651@groan.berkeley.edu>
C:  Date: 27 Jun 91 22:46:30 GMT
Client sends the rest of the article.
C:  .
S:  236 <1990Apr101243.lear@genbank.bio.net> OK.
```

3.13 The SENDME Command

3.13.1 Usage : SENDME

The SENDME command is used to request more than one article from the server. This command may only be used when the BATCH option is turned on. Upon receipt of a SENDME command, the server will respond that it is ready to receive a list of Message-IDs from the client. The client will then send that list via textual response.

The server will then return a batch of articles as defined in RFC 1036 to the client.

3.13.2 Responses

361 Send me a list of Message-IDs
267 A batch of articles follow.
268 **bytecount** articles follow in **data-format** mode.
467 No articles follow.
567 BATCH option not set.

3.13.3 Example

```
C:  OPTION BATCH 100000
S:  104 BATCH=100000
C:  NEWNEWS comp.*,!comp.sys.* 199106211530.10
S:  267 here come Message-IDs since June 21, 1991
S:  <1234@foo.com>
S:  <1991Jun22.002755.5674@uvm.edu>
S:  <1991Jun24.195403.15700@cbnewsj.att.com>
S:  .
C:  SENDME
S:  361 Send me a list of Message-IDs
C:  <1234@foo.com>
C:  <1991Jun22.002755.5674@uvm.edu>
C:  <1991Jun24.195403.15700@cbnewsj.att.com>
C:  .
S:  267 A batch of articles follow.
Server sends client a batch of articles.
```

3.14 The BODY Command

3.14.1 **Usage** : BODY [**n | Message-ID**]

The BODY command operates much the same as ARTICLE, except that only the body of a message is returned - that is, only information following the first blank line of a message. The arguments and their meanings are the same as the ARTICLE command.

3.14.2 Responses

222 **n Message-ID** article retrieved - body follows
226 **n Message-ID bytecount data-format** - body follows
412 no newsgroup has been selected
420 no current article has been selected
423 no such article number in this group
430 no such article found

3.15 The GROUP Command

3.15.1 Usage : GROUP **newsgroup**

The required parameter is the name of the newsgroup to be selected. A list of valid newsgroups may be obtained from the LIST command.

The successful selection response will return the article numbers of the first and last articles in the group, and an estimate of the number of articles on file in the group. It is not necessary that the estimate be correct, although that is helpful; it must only be equal to or larger than the actual number of articles on file. (Some implementations will actually count the number of articles on file. Others will just subtract first article number from last to get an
estimate.)

When a valid group is selected by means of this command, the internally maintained "current article pointer" is set to the first article in the group. If an invalid group is specified, the previously selected group and article remain selected. If an empty newsgroup is selected, the "current article pointer" is in an indeterminate state and may not be used.

Note that the name of the newsgroup is not case-dependent. It must otherwise match a newsgroup obtained from the LIST command or an error will result.

3.15.2 Responses

211 **arts first last group**
    arts = number of articles in the group
    first = first article number in the group,
    last = last article number in the group.
411 no such news group

3.15.3 Example

```
C:  GROUP comp.protocols.tcp-ip
S:  211 99 8217 8315 comp.protocols.tcp-ip
```

3.16 The HEAD Command

3.16.1 Usage : HEAD [n | Message-ID]

The HEAD command is used to request all the headers of a message. Its functionality complements that of the BODY command, and its arguments are interpreted in the same manner.

3.16.2 Responses

221 **n Message-ID** article retrieved - head follows
228 **n Message-ID bytecount data-format** - head follows
412 no newsgroup has been selected
420 no current article has been selected
423 no such article number in this group
430 no such article found

3.17 The LAST Command

3.17.1 Usage : LAST

The LAST command moves the current article pointer to the article with the next lowest article number. If already positioned at the first article of the newsgroup, an error message is returned and the current article pointer is unchanged.

3.17.2 Responses

Upon successful execution, a response indicating the current article number and a Message-ID string will be returned. No text is sent in response to this command.

223 **n Message-ID** article retrieved - request text separately
412 no newsgroup has been selected
422 no previous article.

3.18 The LIST Command

3.18.1 Usage : LIST [ACTIVE|NEWSGROUPS|DISTRIBUTIONS]

LIST or LIST ACTIVE returns via textual response a list of valid newsgroups and associated information. Each line contains the following information:

group last first flag

group is the group name. **last** and **first** are numeric fields, possibly containing leading zeros, indicating the last and the first article numbers for a group. **flag** is a one character flag containing either 'y', 'n', or 'm'. If **flag** is set to 'n', the server will not accept postings for the associated group. If **flag** is set to 'm' the group is moderated, and any unapproved postings will be forwarded to an appropriate moderator.

If NEWSGROUPS is the optional argument, LIST will instead return a list of one line descriptions for each newsgroup the server understands in the following format:

group description

If DISTRIBUTIONS is the argument, the list of distributions known to the server will be printed, one distribution per line in the following format:

distribution description

3.18.2 Responses

215 **w** list follows
(**w** = ACTIVE, NEWSGROUPS, or DISTRIBUTIONS)
415 **w** list unavailable

3.18.3 Example

```
C:  LIST DISTRIBUTIONS
S:  215 DISTRIBUTIONS list follows
S:  local            Local to this site
S:  ig               Systems at IntelliGenetics
S:  bionet           All sites carrying bionet newsgroups
S:  ba               Sites in the San Francisco Bay Area
S:  ca               California Sites
S:  usa              United States
S:  na               North America
S:  world            Everywhere on Usenet in the world
S:  .
```

3.19 The NEWGROUPS Command

3.19.1 Usage : NEWGROUPS **date** [**distributions**]

NEWGROUPS returns via textual response a list of groups that have been created since **date**. The format of **date** is the same as that of the NEWNEWS command. The format of the list is the same as that used by the LIST ACTIVE command. The optional **distributions** argument requests that the server limit the list of newsgroups with those associated with the given list of distributions. The format for the list of distributions is also the same as that used with the NEWNEWS command.

If an empty list returned indicates that no newsgroups have been created since the specified date.

3.19.2 Responses

231 list of new newsgroups follows
432 list of new newsgroups unavailable

3.20 The NEXT Command

3.20.1 **Usage** : NEXT

This command advances the current article pointer to the succeeding article in the current newsgroup (i.e., it has the opposite effect of LAST). If the current pointer is located at the last article in the current group, an error is produced and the current article pointer remains unchanged.

3.20.2 Responses

223 **n Message-ID** retrieved; request head and body separately.
412 no newsgroup has been selected.
421 no next article to retrieve.

3.21 The POST Command

3.21.1 **Usage** : POST [**bytecount**]

The POST command is used by clients to submit new articles for posting into the netnews database. The format of such submissions is specified only in general terms by this document, as the posting software may wish to add, delete, or modify message headers based on authentication information. It is the function of the NNTP server to merely provide for transport of the article to software that will enter the article into the news system. Once the article is entered into the system, it must comply with the latest netnews message format standard (currently described by RFC 1036).

If posting is allowed response code 340 or 341 will be returned, depending on the options negotiated. If BINARY or IMAGE mode have been negotiated, the client must provide the **bytecount** argument to the server. Response code 440 indicates that, for some reason, posting is not allowed.

Once the server has issued a 340 or 341 response, the client will transmit the submission either via textual response or an otherwise negotiated mode. It is suggested that any headers be sent first followed by a blank line, followed by the body.

3.21.2 Responses

240 article posted ok
340 Send submission to be posted; end with CR-LF . CR-LF
341 **bytecount** Ok. Send the bits.
440 Posting not allowed.
441 Posting failed.

3.22 The STAT Command

3.22.1 **Usage** : STAT [**n**]

The STAT command is used to set the current article pointer to article **n**. The acknowledgement will contain the article number **n** and the Message-ID.

For backward compatibility the server must also accept and process Message-IDs. If the article exists, the article number in the response will be set to 0. This makes this usage of the stat command somewhat rare and of little value. Thus, this use is deprecated. Even the original specification couldn't find much value in this use of the STAT command.

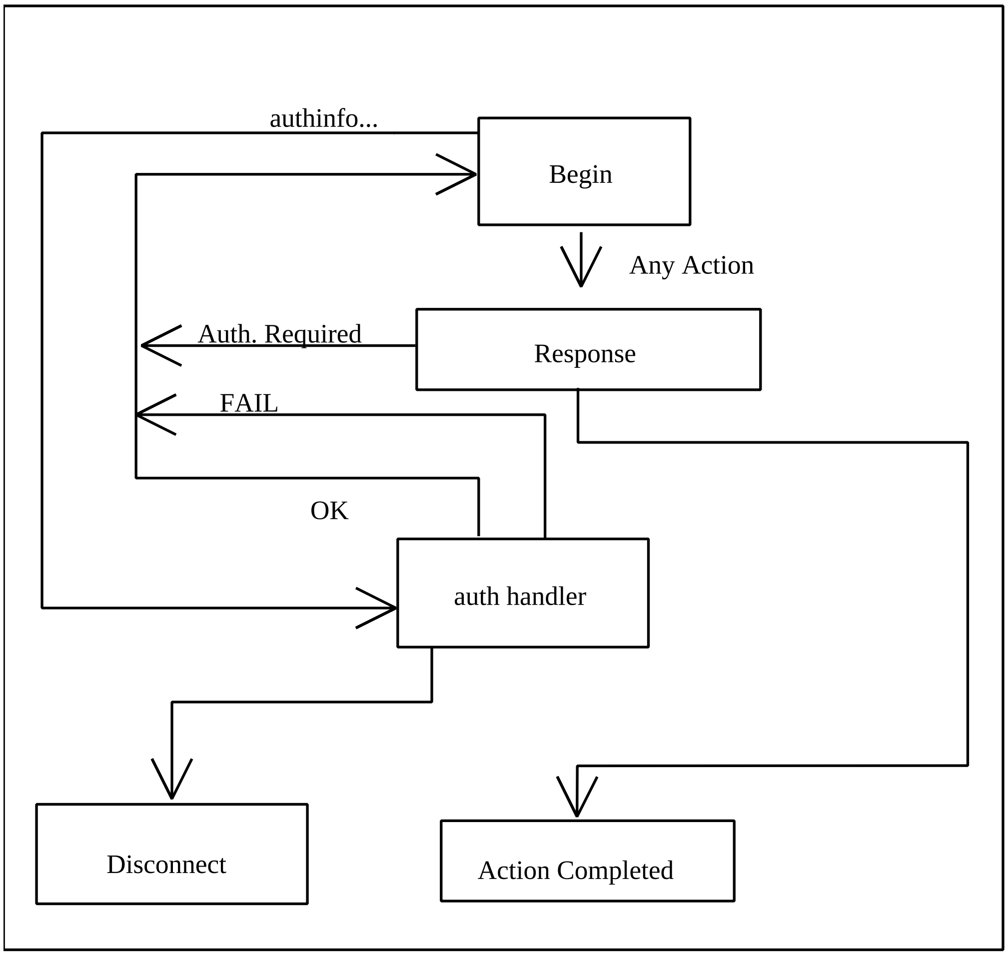
3.22.2 Responses

223 **n Message-ID** retrieved; request head and body separately.
412 no newsgroup has been selected.
423 Not a valid article number.

4 State Diagrams

The following state diagrams represent the conceivable actions of a simple client.

Authorization



5 Response Codes

100 Help text
104 **option**=**value** [**option**=**value**]...
111 YYYYMMDDhhmmss[.xxxxxx]
200 server ready - posting allowed
201 server ready - no posting allowed
205 closing connection - goodbye!
207 SENDSYS response follows, terminated with a single period.
208 **n** SENDSYS response follows. Expect **n** bytes.
211 **arts first last group**
215 **w** list follows
220 **n Message-ID** article retrieved - head and body follow

221 **n Message-ID** article retrieved - head follows
222 **n Message-ID** article retrieved - body follows
223 **n Message-ID** article retrieved - request text separately
224 **n Message-ID bytecount data-format** article follows
226 **n Message-ID bytecount data-format** - body
228 **n Message-ID bytecount data-format** - head follows
230 list of new articles by Message-ID follows
231 list of new newsgroups follows
235 article transferred ok
236 **Message-ID** article transferred ok
237 Leaving IHAVE mode on control channel.
240 article posted ok
250 Authentication Accepted
263 Batch DATA received.
265 **Message-ID** Queue article for delivery with the batch command.
267 A batch of articles follow.
268 **bytecount** articles follow in **data-format** mode.
331 Send me Message-IDs on this channel, articles on the other.
332 send **bytecount** bytes
335 send article to be transferred. End with <CR-LF>.<CR-LF>
340 Send submission to be posted; end with CR-LF . CR-LF
341 **bytecount** Ok. Send the bits.
361 Send list of Message-IDs
363 Proceed with BATCH data.
400 Service unavailable.
407 SENDSYS information not available.
412 no newsgroup has been selected.
415 **w** list unavailable
420 no current article has been selected
421 no next article to retrieve.
422 no previous article.
423 no such article number in this group
430 no such article found
431 Problem with DATA port.
432 list of new newsgroups unavailable
435 article not wanted - do not send it
436 **Message-ID** transfer failed - try again later
437 **Message-ID** article rejected - do not try again.
438 ask me again in a little while
440 Posting not allowed.
441 Posting failed.
450 authentication required
455 Permission denied.
463 Inbound news batch garbled.

465 **Message-ID** article not wanted - do not queue it
466 **Message-ID** ask me again in a little while
467 No articles follow.
490 help not available
500 command not recognized
501 command syntax error
502 I’m not allowed to talk to you.
503 program fault - command not performed
507 **system-name**: no such system.
531 DATA-PATH option not set.
567 BATCH option not set.

6 Sample Sessions

Appendix

A. Authentication Identifiers

It is expected that this section will be periodically updated. The following list contains legal authentication identifiers for NNTP protocol, and where the specification for usage of scheme may be found.

Name	Location
SIMPLE	Appendix of this Document
CAT	Reserved

CAT is reserved for Common Authentication Technology, which is currently in the standardization process.

B. FORMAT Identifiers

It is expected that this section will be periodically updated. The following list contains legal **data-format** identifiers to be used with the FORMAT option, and where their descriptions may be found.

Name	Location
UNIX-IMAGE	This document
TEXT	This document
BINARY	This document

C. The SIMPLE Authentication Method

Usage : AUTHINFO IHAVE SIMPLE **username password**

This method is identified in the AUTHINFO command as ‘‘SIMPLE’’. The two arguments are a username and a password, neither exceeding a length of sixty characters. The password may not be encrypted. All information must be ASCII, and the command line shall be terminated with a CR-LF pair. If the server determines that username and password are acceptable, it will return a 250 return code. The server may not request further information. Otherwise, it may either return 455 indicating authentication failure or simply disconnect.

The server may request this form of authentication at any time during the session, but it may only ask once. Furthermore, this method of authentication provides no means for the server to authenticate itself to the client. Therefore, the client may not request authentication from the server using this scheme.

Responses

250 Authentication Accepted
455 Authentication Failed

Example

```
C: GROUP comp.protocols.tcp-ip
S: 451 SIMPLE
C: AUTHINFO IHAVE SIMPLE lear 5noodlX
S: 250 Authentication OK.
C: GROUP comp.protocols.tcp-ip
S: 211 99 8217 8315 comp.protocols.tcp-ip
```

D. Author’s Address

Eliot Lear

IntelliGenetics, Inc.
700 E. El Camino Real
Mountain View, CA 94040

Phone: +1 (415) 962-7300
EMail: lear@turbo.bio.net