Network Working Group Expires May 25, 1993
Internet Draft

Obsoletes: RFC-977

Network News Transfer Protocol
Version 2

A Protocol for the Stream-Based
Transmission of News

Eliot Lear
Silicon Graphics, Inc.

November, 1992

Status of this Memo

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

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months.Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a “working draft” or “work in progress.” Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

Distribution of this memo is unlimited.

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. Although this draft is not the final draft, we are not far from sending this document to the IESG for consideration as a proposed standard. Interested parties are encouraged in the strongest possible terms to contribute.

Let me be the first to point out my poor grammar. Please send gramatical errors to lear@sgi.com, 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 new functionaility specific to news readers, because we are considering splitting such functionality into a protocol of its own. Please send your comments to the mailing list on reader related issues, as well.

Table of Contents

Table of Contents 3
1 Introduction 4
2 Protocol Overview 6
3 NNTP Command Set 12
3.1 The ARTICLE Command 13
3.2 The AUTHINFO Command 14
3.3 The HELP Command 15
3.4 The MARK Command 16
3.5 The NEWNEWS Command 17
3.6 The OPTION Command 18
3.7 The QUIT Command 20
3.8 The VERSION Command 21
3.9 The BATCH Command 22
3.10 IHAVE 23
3.11 The SENDME Command 28
3.12 The BODY Command 29
3.13 The GROUP Command 30
3.14 The HEAD Command 31
3.15 The LAST Command 32
3.16 The LIST Command 33
3.17 The NEWGROUPS Command 34
3.18 The NEXT Command 35
3.19 The POST Command 36
3.20 The STAT Command 37
4 State Diagrams 38
5 Response Codes 39
6 Implementation Comments 41
Appendix 42

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. As a news reader protocol, 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.[2]

As a transport protocol, 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 also been implemented for DECNET[3]™ and DATAKIT.[4]™ 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., OSI).

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 currently by Stan Barber with help from the networking community.

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 on netnews and its transport protocol have brought 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 distributed and retrieved via netnews.

Authentication has been added so that host addresses need not be the sole means to identify a partner in a communication. Multiple message formats are now supported, to allow for the eventual use of new data formats. Batch transfer has been added to reduce the number of protocol turnarounds required, in order to improve performance over long delay connections.

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

Every attempt is made to keep version 2 of the NNTP protocol to be backward compatible with the

---

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..

4. Chesson, G.L., and Fraser, A.G.: ‘‘Datakit Network Architecture’’, Compcon, pp.59-61, Spring 1980.

first version. Generally, new functionality must be negotiated by the client. The server's default behavior is that of version 1. In order to use commands new to this version the client, therefore, must demonstrate its capabilities and discover those of the server with the VERSION command. There are several changes which may require version 1 server modifications. These are highlighted in the back.

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 the 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”. (Grouping) is indicated by parentheses and *italics* will be used to indicate semantic action. Ellipses (...) are used to indicate a repeating pattern.

1.4 Acknowledgments

This document has its roots in 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, Jim Galvin, Neil Katin, Mel Pleasant, and the people at IntelliGenetics.

2 Protocol Overview

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

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

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. While authentication for a particular session may not be required, implementations must respond properly to requests for authentication, either by attempting to satisfy or denying the request.

2.1 Server to Server Exchanges

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 for fewer protocol exchanges (turnarounds).

When a connecting server is to distribute netnews articles to another site, it may do so in one of several ways, all of of which involve the receiving 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 groups transactions together. This method is known as BATCH. Under this scheme, the connecting server sends an index of queued articles to the receiving server, and the latter 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 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 stream 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

Some commands in the NNTP specification are intended 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 retrieves a list of newsgroups and article numbers from the server to determine which newsgroups have new messages.

When contacted for news reading purposes, the NNTP server must 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.

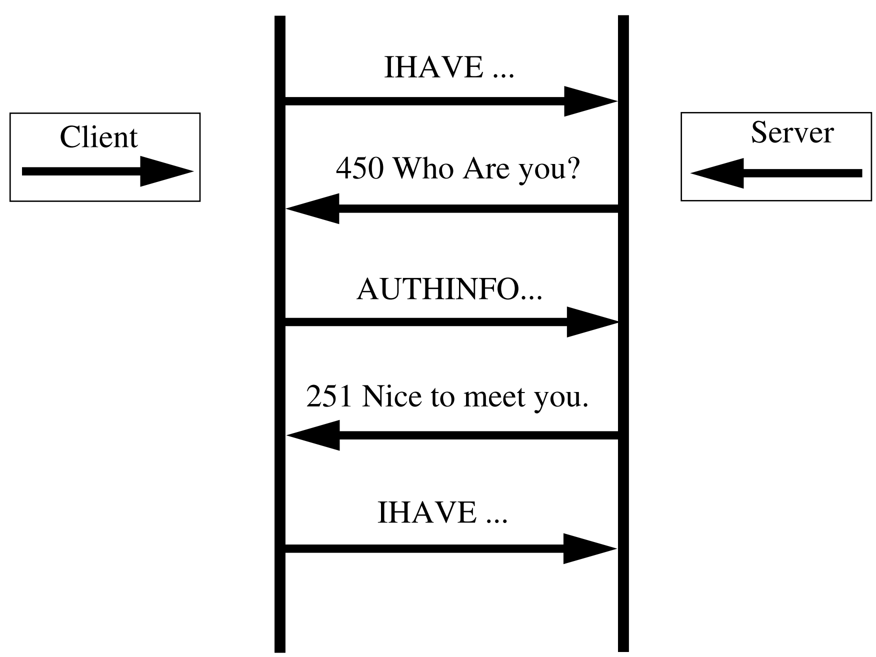
2.3 Connection & Greeting

Clients initiate a TCP connection on the port designated by the Internet Assigned Number Authority (IANA) to the server with which they wish to exchange news. NNTP requires a reliable flow controlled 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 responds with a greeting indicating its readiness to provide service. That response will contain one of the following codes:

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

2.4 Authentication

Version 2 of NNTP protocol provides for host to host bidirectional 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 entity. 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.



Client and Server Authentication Transactions

Either the server or the client 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. Additional methods may be added through the normal standardization process.

The server requests authentication by transmitting a return code in response to a client command indicating that authentication is required. If the client responds improperly, the server may conclude that the client 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.

After the server has requested authentication by issuing a response code to a client command, once authentication is completed the server will expect a new command from the client, and the client must either reissue the command it sent before authentication began, or it must issue some other command.

2.4.1 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.

2.5 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. Only one command may be sent per line. Terminal symbols listed in this document are to be treated as case insensitive. All commands are transmitted in ASCII with the high order bit cleared. Command lines should not contain more than 512 characters, including the CR-LF pair.

2.6 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. Continued response lines must use one response code.

For example:

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

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.

2.6.1 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.
```

2.6.2 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
```

2.6.3 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=PREARRANGED
S:  204 FORMAT=PREARRANGED
C:  IHAVE
S:  361 Send list of Message-IDs
C:  <9104181633.AA08820@msw.usc.edu>
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 a data format requring it, an extended response will involve an integer number of octets to be transferred. When this document uses the terms “byte” or “bytecount” it refers to octets.

For example:

```
C:  OPTION FORMAT=PREARRANGED
S:  204 FORMAT=PREARRANGED
C:  ARTICLE <9104181633.AA08820@msw.usc.edu>
S:  224 0 4249 <9104181633.AA08820@msw.usc.edu>
```

Server sends 4249 bytes in prearranged 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.

2.6.4 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 overall status 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.

5xx - Command unimplemented, incorrect, or program 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
x5x - Authentication
x6x - Batch transfers
x8x - Nonstandard (private implementation) extensions

The exact response codes that should be expected from each command are detailed in the description of that command. In addition, there are several response codes that should be used by the server and recognized by the client at any time when a response code is to be sent.

400 service discontinued
438 service temporarily unavailable
450 authentication required
500 command not recognized
501 command syntax error
503 program fault - command not performed

In addition, the server may send at any time a response code is expected the following debugging codes:

09x - Debugging output

When debugging output is seen by the client it may be ignored or transmitted to a log of some form, but the state of the client side of the connection must not otherwise change.

2.7 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 is a **data-format** identifier; if this identifier is "PREARRANGED", for example, then both ends might not do LF/CR-LF 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 individual prearrangement only. In addition, it is suggested that such identifiers begin with "X-". Usage of identifiers listed in the appendix must be in accord with the definition.

2.8 Data Representations

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

2.8.1 CANONICAL TEXT

By default all articles and article information (such as lists of Message-IDs transmitted by either one side or the other) are sent as ASCII text; each such transmission 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 (e.g., CR-LF . 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.

2.8.2 PREARRANGED

This transfer method allows both sides to use a prearranged data format that will presumably optimize data transfers. For example, if both ends are storing data in a format such as EBCDIC they may choose to agree to a prearrangement to allow for transfer of articles in that format. Implmentations that use prearranged format must insure that messages must remain transferrable in canonical formats.

2.9 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. Data need not be transferred in canonical text if another data format has been negotiated by the FORMAT option.

3 NNTP Command Set

NNTP has the following command set:

ARTICLE, AUTHINFO, HELP, MARK, NEWNEWS, OPTION, QUIT, VERSION

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 retrieval protocol is being developed. As that protocol enters the standards process, news reader clients should migrate to 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.

3.1 The ARTICLE Command

3.1.1 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 specified by the Message-ID is transmitted as a textual response, or as otherwise negotiated as described elsewhere 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. If an invalid article number is specified, an error reply is sent and the current article pointer is left unchanged.

3.1.2 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 and 224 return 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.

3.2 The AUTHINFO Command

3.2.1 Usage: AUTHINFO **authtype**
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. Once the command is given by the client and accepted by the server, both sides exchange authentication data based on the method referred to by **authtype**. When the two sides are done, the server will issue a final return code, and normal processing of commands will once again take place.

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. Other authenication types must be defined through the standards process.

If the client issues an AUTHINFO command, and the server is satisfied with authentication and will not authenticate itself, the server should respond with a 250 return code indicating that no further authentication is desired. The client must then issue NNTP commands, and not attempt to continue the authentication process.

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 many separate standards.

3.2.2 Responses
250 Authentication accepted
350 Entering authentication mode.
450 authentication required.
451 **authtype** authentication wanted.
452 Authentication rejected.

3.2.3 Example

Simple Authentication Exchange
```
S: 200 Podunk.Edu NNTP server ready 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
S: 350 Entering authentication mode.
C: webber x7e37cn46
S: 250 Authentication Accepted, now where were we?
C: IHAVE <1990Apr101243.lear@genbank.bio.net>
```

3.3 The HELP Command

3.3.1 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.

3.3.2 Responses
100 Help text follows

3.4 The MARK Command

3.4.1 **Usage**: MARK

MARK is used in conjunction with the NEWNEWS and NEWGROUPS command to keep track of changes on the server. The text of the MARK command is not to be interpretted by the client, but passed back to the server as an argument to NEWNEWS or NEWGROUPS. Though one obvious return of text information would be the date, it is by no means the only text that MARK may return. It cannot be stated in strong enough terms that clients MUST NOT interpret the data returned from the server.

3.4.2 Responses
206 **cookie-string**

3.4.3 Example
```
C:  MARK
S:  211 19920215010203.025
C:  NEWNEWS comp.* &199205000001.000
S:  230 Here comes newnews...
```

In this example, the client has retrieved a new mark for the next session and is using the one it retained from the previous session as a key for newnews. Note that using only the time with this mechanism allows room for duplicate messages to be transmitted.

3.5 The NEWNEWS Command

3.5.1 Usage: NEWNEWS **newsgroups** [**date**| "&"**cookie-string**] [**distributions**]
This command requests a list of Message-IDs that refer to associated messages that have been received after the specified date or based on information stored in a string obtained by the client from the server through the MARK command.

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 either the newsgroup does not mismatch any of the patterns, or the longest matched pattern is longer than the longest mismatched pattern.
- Order is not significant.

date is a date in ISO 3307 format in Greenwich Mean Time (GMT) of the following form: YYYYMMDDHHMMSS[.x[x*]*. Although this specification places no limitation on the precision of the time, implementors should take care to allow for drift of either of the two clocks involved. For backward compatibility, a server must accept the date and time in the form of yymmdd hhmmss ["GMT"]. Note that the change of date format requires modifications in version 1 servers. At the very least, the robustness principle requires that version 1 servers merely return errors when they see the new format.

cookie-string a string other than a date to be interpretted by the server. It is distinguished from a date by a leading ampersand ("&").

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.

3.5.2 Responses
230 list of new articles by Message-ID follows

3.5.3 Example
```
C:  NEWNEWS *,!comp.*,comp.sys &k8128ddj334fdd
S:  230 here comes a list of articles
S:  <1234@foo.com>
S:  <1991Jun22.002755.5674@uvm.edu>
S:  <1991Jun24.195403.15700@cbnewsj.att.com>
S:  .
```

3.6 The OPTION Command

3.6.1 Usage: OPTION **option=value** [**option=value**]...
The OPTION command is used to negotiate certain parameters for a connection. **option** is the option name, as specified 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. Such option names begin with "X-". Servers must reject unknown options by returning them set to ''UNIMPLEMENTED".

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.

3.6.2 Responses
204 **option=value** [**option=value**]...

3.6.3 Example

```
C:  OPTION FORMAT=PREARRANGED BATCH=200000
S:  204 FORMAT=CANONICAL-TEXT BATCH=200000
```

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

The following set of options and their mandatory default settings must be implemented on all NNTP servers conforming to version 2:

BATCH=OFF
FORMAT=CANONICAL-TEXT
DATA-PATH=OFF

3.6.4 The BATCH Option
3.6.4.1 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 connecting server will transmit. If a larger article is to be transmitted, the connecting server must first turn off batch mode.

If a server replies with a smaller BATCH limit than the client has requested, the client must not exceed that upper limit. Implementors should guard servers against denial-of-service attacks, by not simply returning total space available for spooling of articles.

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

3.6.4.2 Responses
204 BATCH=(**bytecount**|OFF)
bytecount in the reply code is the largest batch the server can receive. The server may reply with a number larger than that requrested by the client, indicating 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.

3.6.4.3 Example

```
C:  OPTION BATCH=200000
S:  204 BATCH=100000
```

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

3.6.5 The DATA-PATH Option

3.6.5.1 Usage: OPTION DATA-PATH=(**ON**|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 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 data port on the client, and then send a reply to the option request. That connection will be used as a 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. Authentication may occur over the second connection only when the client has sent and the server has accepted the AUTHINFO command with a 350 return code.

The default value of this option is OFF.

3.6.5.2 Responses

204 DATA-PATH=(ON|OFF)

3.6.6 The FORMAT Option

3.6.6.1 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”. One other format is defined in this document - PREARRANGED. Other data formats may be added through the normal standards process.

3.6.6.2 Responses

204 FORMAT=**data-format**

3.6.6.3 Example

```
C:  OPTION FORMAT=PREARRANGED
S:  204 FORMAT=CANONICAL-TEXT
```

In the above example, the server is rejecting the PREARRANGED format.

3.7 The QUIT Command

3.7.1 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. See the IHAVE command for more information.

3.7.2 Responses
205 closing connection - goodbye!

3.8 The VERSION Command

3.8.1 Usage: VERSION **version** [**comment**]

This command is used to exchange version information between the client and the server. **version** is an integer number indicating the current version of the NNTP standard being used by the implementation. Implementations using this specification must specify 2 as the version number. **comment** is a text string that may be used to identify the implementation. Implementations relying on earlier versions of the protocol must return a 500 class error (command not recognized).

3.8.2 Responses

207 **version** [**comment**]

3.8.3 Example

The first example below represents two version 2 implementations. The second example is a version 2 client contacting a version 1 server.

```
C: VERSION 2 nnpost6.16
S: 207 2 nntpd 2.0 compiled by lear@sgi.com on Mar 1, 1992.

C: VERSION 2 nnpost6.16
S: 500 Command not recognized.
```

3.9 The BATCH Command

3.9.1 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. 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 the PREARRANGED format). 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.

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

3.10 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.

3.10.1 IHAVE With No Options

3.10.1.1 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 by another NNTP session, the server need not keep the connection idle.

3.10.1.2 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

3.10.1.3 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.
```

3.10.2 IHAVE With Formats Other than CANONICAL-TEXT

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

This form of IHAVE is used to transmit articles in modes other than CANONICAL-TEXT negotiated through the FORMAT option. 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.

3.10.2.2 Responses
235 article transferred ok
332 **bytecount** bytes wanted
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

3.10.3 IHAVE With BATCH Option Set

3.10.3.1 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 a textual response, listing the Message-IDs of the articles it chose to receive. Once the client receives the list of articles that the server wants, it will transmit them using the BATCH command. When a client offers the server an article that is being transmitted by another client, the server may save the Message-ID and request it later in response to another IHAVE of the same client. There is no guarantee that the client will still have the article at that later point.

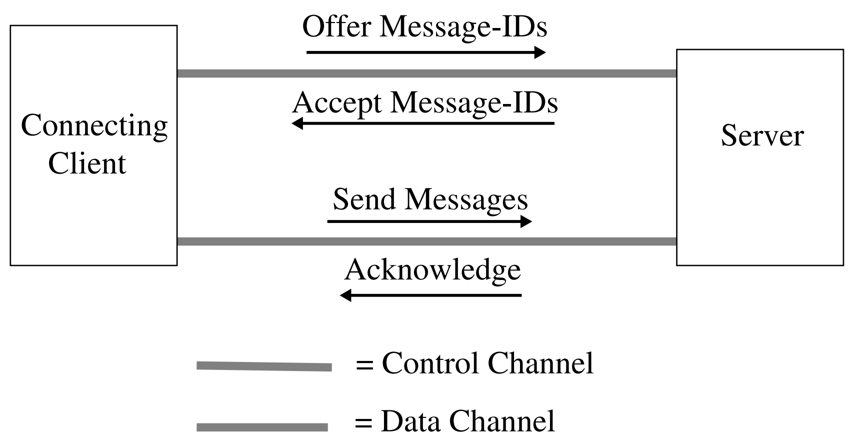
3.10.3.2 Responses
361 Send list of Message-IDs
265 List of Message-IDs follows.

3.10.3.3 Example
```
C:  OPTION BATCH=100000
S:  204 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 List of Message-IDs follows.
S:  <1991Jun24.195403.15700@cbnewsj.att.com>
S:  <1991Jul1.214801.287@genbank.bio.net>
S:  .
C:  BATCH
S:  363 Transmit a batch of articles
```
Client transmits two articles
```
C:  .
S:  263 Batch received, thanks!
```

3.10.4 IHAVE with the DATA-PATH Option Set

3.10.4.1 Usage: IHAVE
If the DATA-PATH option is set, a second stream 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 Message-IDs from the server only after the server returns a 237 reply code when it has

received the last message from the client.



The Dual Channel Model

When the client transmits an article to the server on the data channel it will first indicate the transmission format for the forthcoming message. If no format changes have been negotiated, the client will send one line to the server containing "TEXT" and then transmit the article 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 a non text format, 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. The client need not wait for the confirmation to start sending the next message. If, however, a failure code of 437 is returned, the client should send no more articles after the current one. Under no circumstances should a client transmit a partial message.

If a server requests a message that the client has offered, and the client is unable to service the request, the client must send on the data channel ''NOMSG **Message-ID**" instead of sending the message. For the server's benefit the client MUST notify the server of such errors as soon as possible. For example, if a client has offered the Message-ID of a message that was expired, the client would send NOMSG as soon as it has finished transmitting the current article.

The offering of Message-IDs will likely 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 terminated 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. It may attempt to reset the data connection by renegotiating with the server, using the option command.

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.

3.10.4.2 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.

3.10.4.3 Example
The control connection communication may look like the following:

```
C:  OPTION FORMAT=PREARRANGED DATA-PATH=ON
```
Server opens connection to client on the data port.
```
S:  204 BINARY=ON DATA-PATH=ON
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.
```

3.11 The SENDME Command

3.11.1 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.

3.11.2 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.

3.11.3 Example
```
C:  OPTION BATCH=100000
S:  204 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.

3.12 The BODY Command

3.12.1 Usage: BODY [**n | Message-ID**]
The BODY command operates the same as ARTICLE except that only the body of an article is returned - that is, only information following the first blank line of a message (i.e., no headers). The arguments and their meanings are the same as the ARTICLE command.

3.12.2 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

3.13 The GROUP Command

3.13.1 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 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.

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.

3.13.2 Responses
411 no such news group

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

3.14 The HEAD Command

3.14.1 **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.

3.14.2 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

3.15 The LAST Command

3.15.1 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, error code 422 is returned and the current article pointer is unchanged.

3.15.2 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.

3.16 The LIST Command

3.16.1 Usage: LIST

LIST returns via textual response a list of valid newsgroups and associated information.

group last first flag

group is the group name, which must be a NETASCII newsgroup 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. A client must treat any other character or group of characters in the flag field as if it was set to “y”.

3.16.2 Responses

215 list follows
216 **w bytecount** list follows in **data-format**

3.16.3 Example

```
C:  LIST
S:  215 list follows
S:  alt.business.multi-level 0000000060 00001 y
S:  alt.culture.tuva 0000000043 00001 y
S:  alt.food 0000000015 00001 y
S:  alt.food.mcdonalds 0000000089 00001 y
S:  alt.food.cocacola 0000000169 00001 y
S:  .
```

3.17 The NEWGROUPS Command

3.17.1 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 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 is returned indicates that no newsgroups have been created since the specified date.

The NEWGROUPS command is of little value, given that it does not give the client information on what groups were deleted. Instead of using the NEWGROUPS command, a client should use LIST, and compare differences between the current list and previous lists.

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

3.18 The NEXT Command

3.18.1 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, error code 422 is returned and the current article pointer is unchanged.

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

3.19 The POST Command

3.19.1 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. Posting implementations should obey any requirements for posting of articles mentioned in the current netnews article standard. 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 a format other than canonical text has 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.

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

3.20 The STAT Command

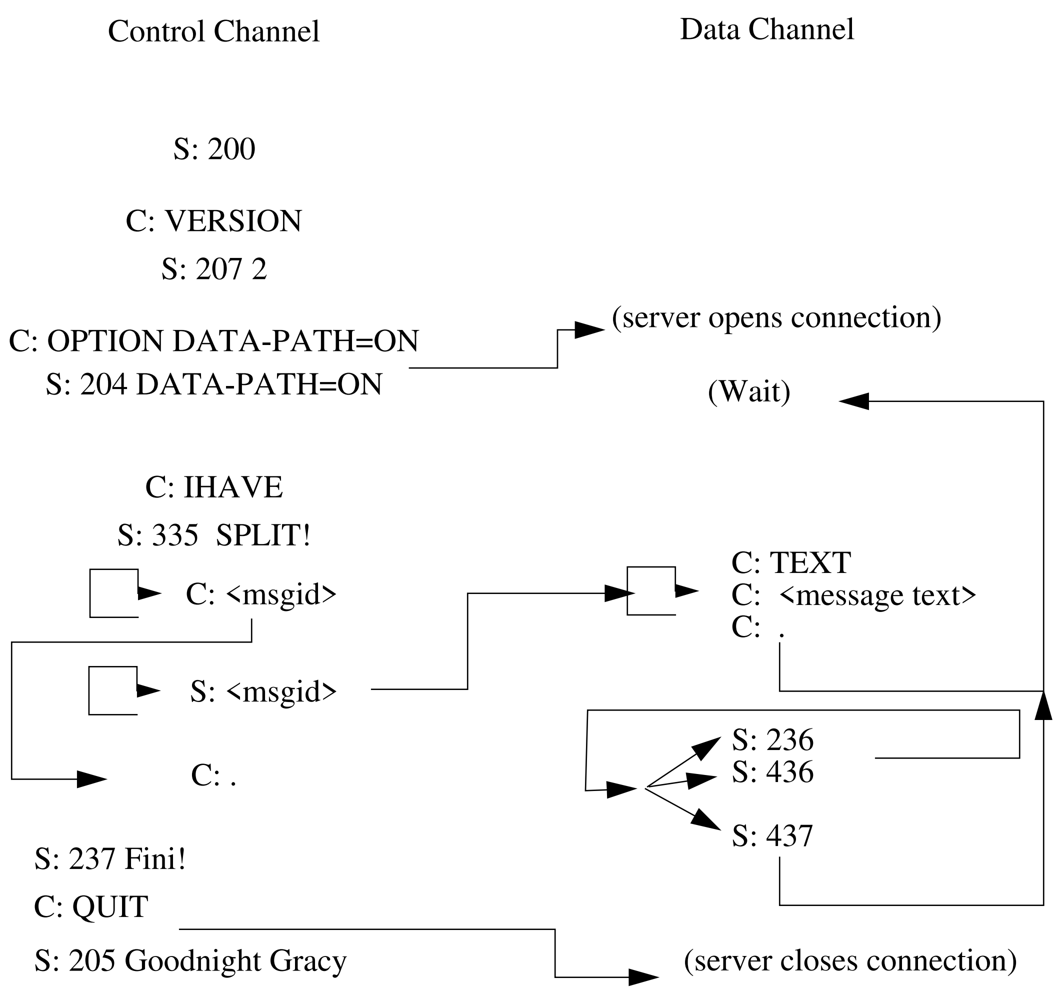
3.20.1 Usage: STAT [**n**]
The STAT command is used to set the current article pointer to article **n**. The acknowledgement will contain the article number **n** and the Message-ID.

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, such use is deprecated, and a client should not issue this command with a Message-ID.

3.20.2 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

Here is a samlple state diagram of a session using the DATA-PATH option. Note that the data channel connection is opened when the OPTION is set, and is closed when QUIT is received. In the diagram blow, state follows from top to bottom, unless indicated otherwise by a line.



5 Response Codes

Response codes are listed in order. Response codes that contain information to be parsed by the connecting implementation must be formatted as listed here. Clients must use as little information returned to process a line containing a return value. In many cases, simply parsing the first character will suffice (e.g., 215 list follows).

100 Help text follows
200 server ready - posting allowed
201 server ready - no posting allowed
204 **option=value** [**option=value**]...
205 closing connection - goodbye!
206 **cookie-string**
207 **version** [**comment**]
211 **arts first last group**
215 list follows
216 **w bytecount** list follows in data-format
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** retrieved; request head and body separately.
224 **n Message-ID bytecount data-format** article follows
226 **n Message-ID bytecount data-format** - body follows
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 List of Message-IDs follows.
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 **bytecount** bytes wanted
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 Send the bits.
361 Send list of Message-IDs
363 Proceed with BATCH data.
400 Service unavailable.
411 no such news group
412 no newsgroup has been selected
420 no current article has been selected
421 no next article to retrieve.
422 no previous article.
423 Not a valid article number.
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
439 service temporarily unavailable
440 Posting not allowed.
441 Posting failed.
450 authentication required
455 Authentication Failed
463 Inbound news batch garbled.
467 No articles follow.
500 command not recognized
501 command syntax error
502 I’m not allowed to talk to you.
503 program fault - command not performed
531 DATA-PATH option not set.
567 BATCH option not set.

6 Implementation Comments

Existing implementations of NNTP senders often connect to the receiving server on a periodic basis. Such periodic connections increase protocol overhead in agreagate when few articles are actually transmitted during each connection. Start-up and tear-down costs should be reduced to a minimal number by keeping connections open and using interprocess communications to offer newly received articles. One such implementation already uses this mechanism. Implementations of version 2 can both improve and exaserbate the situation. The data-path option incurs more start-up/tear-down overhead. However, since communication from the receiving server only occurs when articles are to be accepted, and since transmissions are asynchronous, information can easily be batched by lower layers to increase packet sizes and reduce the number of packets. Thus, over time the start-up/tear-down overhead is mediated.

The previous version of NNTP provided an application level flow control through the IHAVE mechanism. The dual data path transmission method relies on TCP's own flow control mechanism. An overloaded server not accepting more data will cause TCP to flow control the sending side. This mechanism is sufficient and preferable for NNTP over TCP-IP.

At certain instances, it is advantagious transmit information as quickly as possible. On such instance is when the receiving server is attempting to load balance across its connections, in order to optimize network bandwidth. If network writes are buffered on the receiving server, the sending server will idle waiting for a request that is sitting in the receivers cache.

The new version of NNTP enhances reliability pull feeds (i.e., those using NEWNEWS) through the use of MARK. NEWNEWS gives the connecting server selection control of its news feed without interaction with the news backend (e.g., B-News or C-News). An implemention can issue a NEWNEWS command with any feed it wishes to receive, and use MARK to ensure that no articles are lost. A problem occurs when the newsgroups or distributions are added. The MARK that was sent will no longer be valid for any number of reasons. If this is a desirable mechanism for newsgroup feeds, some sort of article cache can be used to alleviate the cost of file opens and text searches, since the criteria used for article selection is present when it is received on the sending server.

The flood nature of NNTP increases protocol overhead, as articles are offered many more times than they are exchanged. A true multicast netnews distribution mechanism could likely reduce net work news traffic by as much as a factor of ten. Even though no such protocol exists at the time of this writing, there are ways to improve NNTP's agreagate network performance which deserve mention in this document.

Appendix

A Changes

The following commands have been added to NNTP in this document:

AUTHINFO, BATCH, MARK, OPTION, SENDME, VERSION

The NEWGROUPS command has been deprecated. The NEWNEWS command has been enhanced with the help of the MARK command.

The STAT command has been clarified and its use with a Message-Id has been deprecated.

Dates may be transmitted in ISO 3307 format. N.B. this may require modifications to version 1 servers.

Several response codes have been added to handle temporary failures, authentication, and options processing. Debugging response codes are clarified.

The use of continuation marks in column four is allowed. N.B. this may require modifications to version 1 clients.

Implementations may now communicate data with the eighth bit set if the message format being transmitted allows for it.

Transmission modes are defined for optimization.

B 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.

Authentication Identifier	Reference
SIMPLE	Appendix of this document

C 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.

Format Name	Reference
PREARRANGED	This document
CANONICAL-TEXT	This document

D The SIMPLE Authentication Method

D.1 Usage: AUTHINFO SIMPLE
This method is identified in the AUTHINFO command as "SIMPLE". Once the server returns a 350 code indicating that it will accept this form of authentication, the client will transmit a username and a password on a single line separated by a space character. If the server is satisfied with the double, it will return 250 indicating acceptance. Otherwise it will return 455 indicating authentication failure, or it may 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.

D.2 Responses
250 Authentication Accepted
455 Authentication Failed

D.3 Example
```
C: GROUP comp.protocols.tcp-ip
S: 451 SIMPLE
C: AUTHINFO SIMPLE
S: 350 send me simple authentication information
C: lear 5noodlX
S: 250 Authentication OK.
C: GROUP comp.protocols.tcp-ip
S: 211 99 8217 8315 comp.protocols.tcp-ip
```

E Author's Address

Eliot Lear
Silicon Graphics, Inc.
P.O. Box 7311
2011 North Shoreline Blvd, MS 15-730
Mountain View, CA 94039-7311