Help for News

Properties Events Methods

Registration Information

Order Form

Getting Custom Controls Written

Licensing Information

Description

News provides easy access to Network News Transfer Protocol (NNTP) servers as specified in RFC977, and includes support for popular extensions used by nearly all existing news servers. News also supports the user authentication protocol used by many news servers, especially those of commercial news providers.

Mabrys News control also is unique in providing access to non-standard server commands through a Read/Write methods.

Both RFC977 and RFC850 are mandatory reading before attempting to undertake a serious News client program. You should read those before writing any code at all!

File Name

NEWS1.VBX, NEWS32.OCX

Object Type

MabryNews

VBX Compatibility

VB 2.0 and above

Distribution Note When you develop and distribute an application that uses News, you should install the control file into the users Windows SYSTEM directory. News has version information built into it. So, during installation, you should ensure that you are not overwriting a newer version of News.

News Properties

All of the properties that apply to this control are in this table.   Properties that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

- ***About** Property
- ***Action** Property
- ***ArticleCount** Property
- ***ArticleID** Property
- ***ArticleIDs** Property
- ***ArticleIDsCount** Property
- ***Blocking** Property
- ***BodyText** Property
- ***Date** Property
- ***Debug** Property
- ***Distribution** Property
- ***EMailAddress** Property
- ***ExpirationDate** Property
- ***FirstArticle** Property
- ***FollowUpTo** Property
- ***From** Property
- ***Group** Property
- ***GroupCount** Property
- ***Groups** Property
- ***Headers** Property
- ***HeadersCount** Property
- ***HeaderText** Property
- ***Host** Property
- ***LastArticle** Property
- ***Lines** Property
- ***LogonName** Property
- ***LogonPassword** Property
- ***NewsGroups** Property
- ***Organization** Property
- ***PostingHost** Property
- ***ReadData** Property
- ***References** Property
- ***ReplyTo** Property
- ***Subject** Property
- ***TimeOut** Property
- ***Version** Property
- ***WriteData** Property
- ***XOverHeaders** Property

***XOverHeadersCount** Property

News Methods

All of the methods that apply to this control are in this table.   Methods that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***AboutBox** Method
***Connect** Method
***Disconnect** Method
***GetAllGroups** Method
***GetArticle** Method
***GetBody** Method
***GetHeader** Method
***GetHelp** Method
***GetNewGroups** Method
***GetNewNews** Method
***GetStatus** Method
***NextArticle** Method
***PostArticle** Method
***PreviousArticle** Method
***Read** Method
***SelectGroup** Method
***Write** Method
***XOver** Method

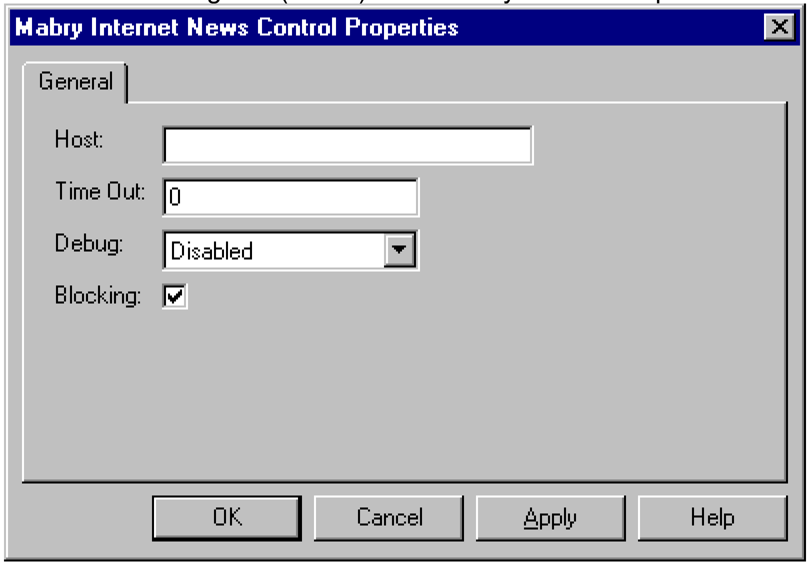
News Events

All of the events that apply to this control are in this table.   Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***AsyncError** Event
***DataReady** Event
***Debug** Event
***Done** Event

News Control Property Page

Click on the dialog box (below) in the field you need help for.



Mabry Internet News Control Properties

General

Host:

Time Out: 0

Debug: Disabled

Blocking: ✓

About Property

See Also

Description

Displays an about box.

Usage

*object.***About**[= *integer*]

Remarks

When you set this property to any value, the News control displays its about box.

This property is only present in the VBX version of News.

See Also

Methods:

AboutBox

AboutBox Method

See Also

Description

Displays the News controls About box.

Syntax

*object.***AboutBox**

Part	Description
object	Required. A News control.

Remarks

When you execute this method, the News control displays its about box.

This property is only present in the OCX-32 version of News.

See Also

Properties:

About

Action Property

Description

Used to invoke news control methods.

Usage

*object.***Action**[= *integer*]

Remarks

Setting this property to one of the values below starts the method associated with that value.

Constant	Value	Method
mNewsNoAction	0	
mNewsConnect	1	Connect
mNewsDisconnect	2	Disconnect
mNewsGetArticle	3	GetArticle
mNewsGetBody	4	GetBody
mNewsGetHeader	5	GetHeader
mNewsGetStatus	6	GetStatus
mNewsGetAllGroups	7	GetAllGroups
mNewsGetNewGroups	8	GetNewGroups
mNewsSelectGroup	9	SelectGroup
mNewsGetHelp	10	GetHelp
mNewsGetNewNews	11	GetNewNews
mNewsNextArticle	12	NextArticle
mNewsPreviousArticle	13	PreviousArticle
mNewsPostArticle	14	PostArticle
mNewsQuit	15	Quit
mNewsWrite	17	Write
mNewsRead	18	Read
mNewsXOver	20	Xover

Data Type

Integer (enumerated)

See Also

Methods:

- Connect
- Disconnect
- GetAllGroups
- GetArticle
- GetBody
- GetHeader
- GetHelp
- GetNewGroups
- GetNewNews
- GetStatus
- NextArticle
- PostArticle
- PreviousArticle
- Read
- SelectGroup
- Write
- Xover

ArticleCount Property

See Also

Description

After a group has been selected by the SelectGroup action this property will contain an estimate of the number of articles in the group.

Usage

*object.***ArticleCount**

Data Type

Integer (long)

See Also

Properties:

- Action

Methods:

- SelectGroup

ArticleID Property

See Also

Description

Contains a message-id or article number.

Usage

*object.***ArticleID**[= *string*]

Remarks

Message-ids are enclosed in angle-brackets (<>) while article numbers are given as a string of decimal digits.

Data Type

String

See Also

Properties:

ArticleIDsCount

ArticleIDs Property

See Also

Description

An array of ArticleIDs.

Usage

*object.***ArticleIDs(** *index* **)**

Remarks

The indices of this array range from zero to ArticleIDsCount - 1.

The elements of the ArticleIDs property are set after the GetNewNews command has finished execution.

Data Type

String

See Also

Properties:

ArticleIDsCount

Methods:

GetNewNews

ArticleIDsCount Property

See Also

Description

Number of elements in the ArticleIDs array.

Usage

*object.***ArticleIDsCount**

Remarks

This property is read-only and is only valid after executing the GetNewNews command at run-time.

Data Type

Integer (long)

See Also

Properties:

ArticleIDs

Methods:

GetNewNews

AsyncError Event

Description

If an error occurs while the control is executing a command asynchronously in non-blocking mode the AsyncError event is fired.

Syntax

Sub *ctlname*_**AsyncError (***ErrorCode* **As Long,** *Description* **As String)**

Remarks

Winsock error codes are specified in winsock.h.   The Description parameter provides an error message you can display if an error occurs.

Blocking Property

Description

Determines whether calls are blocking or non-blocking.

Usage

*object.***Blocking**[= *boolean*]

Remarks

Determines whether methods will execute in blocking (synchronous) or non-blocking (asynchronous) mode.

The Blocking property is read-only at runtime.

Data Type

Integer (boolean)

BodyText Property

See Also

Description

Contains the body of a news message.

Usage

*object.***BodyText**[= *string*]

Remarks

This property holds the body text of an article after the GetBody command is executed.

Data Type

String

See Also

Properties:

HeaderText

Methods:

GetBody

Connect Method

See Also

Description

Connects to the news server specified by the Hostr property.

Syntax

*object.***Connect()**

Remarks

This method connects to the news server defined by the Hostr property. Host can be specified as either a name (i.e., news.microsoft.com) or an IP address (i.e., 123.123.123.123). If the specified news server requires user authentication then the LogonName and LogonPassword properties must be valid.

See Also

Properties:

- LogonName
- LogonPassword

Events:

- Done
- AsyncError

DataReady Event

See Also

Description

Fired when data from the last non-blocking call is available.

Syntax

Sub *ctlname*_**DataReady ()**

Remarks

Some actions (such as GetAllGroups) return large amounts of data in broken up into chunks.   The DataReady event is fired when Blocking= False and just after a chunk has been received if another chunk is expected.   This event allows you to process certain types of data (such as a list of groups) as it arrives.

Note that the last chunk of any action does not fire the DataReady event, the Doneevent notifies you when the last (or only) chunk arrives. You can ignore the DataReady event if you choose and simply rely upon the Doneevent at which time you can process all of the data.   However using the DataReady event allows you to do things like add items to list boxes as the data arrives and display progress bars.

See Also

Events:

Done

Date Property

Description

Message date and time.

Usage

*object.***Date**[= *string*]

Remarks

Date and time message received, or date and time of message to send. You must place correctly formatted dates in this property. To assign the current date and time to the Date property use the following statement:

```
News1.Date = Format(now,"ddd, dd mmm yyyy hh:mm:ss")
```

Upon transmission the control will convert the Date to GMT, as recommended by RFC1036, and upon reception dates are converted to the local time format.

Data Type

String

Debug Event

See Also

Description

Fired when debugging information is available.

Syntax

Sub *ctlname*_**Debug (***DebugText* **As String)**

Remarks

When the Debugproperty is non-zero the Debug event will be fired as significant control events occur.   The DebugText string passed to Debug will contain information which can be listed to the debug window or other tracing facilities.

See Also

Properties:

Debug

Debug Property

See Also

Description

Determines whether debugging information will be provided by the control.

Usage

*object.***Debug**[= *integer*]

Remarks

Debugging information is sent to the control depending upon the following values:

Constant	Value	Description
mNewsNoDebug	0	No debugging information is provided.
mNewsDebug	1	Debugging information is provided via the Debugevent.

Data Type

Integer (enumerated)

See Also

Events:

Debug

Disconnect Method

See Also

Description

Disconnects from the news server.

Syntax

*object.***Disconnect**

Part	Description
object	Required. A News control.

See Also

Methods:

Connect

Distribution Property

Description

A list of distribution groups, enclosed in angle brackets.

Usage

*object.***Distribution**[= *string*]

Remarks

If specified, the distribution portion of a new newsgroup (e.g, 'net' in 'net.wombat') will be examined for a match with the distribution categories listed, and only those new newsgroups which match will be listed.   If more than one distribution group is to be listed, they must be separated by commas within the angle brackets.

Data Type

String

Done Event

See Also

Description

This Done event is fired when any non-blocking method completes.

Syntax

Sub *ctlname***_Done (***ErrorCode* **As Integer)**

Remarks

The ErrorCode parameter will be non-zero if the no errors occured during the execution of the last asynchronous command.

See Also

Events:

DataReady

EMailAddress Property

See Also

Description

Users email address.

Usage

*object.***EMailAddress**[= *string*]

Remarks

When a message is being Posted a From: line of the following form is automatically created:

```
From: EMailAddress
```

EMailAddress should contain a valid email address such as zthomas@mabry.com (Zane Thomas)

Data Type

String

See Also

Properties:

- LogonPassword

Methods:

- Connect
- Post

ExpirationDate Property

See Also

Description

Date the server should expire a posted message.

Usage

*object.***ExpirationDate**[= *string*]

Remarks

If not present, the local default expiration date is used. This field is intended to be used to clean up messages with a limited usefulness, or to keep important messages around for longer than usual. ExpirationDate is formatted exactly the same as Date.

Data Type

String

See Also

Properties:

Date

FirstArticle Property

See Also

Description

First article number in a block of articles.

Usage

*object.***FirstArticle**[= *integer*]

Remarks

Used with methods which return a range of articles or require that a range be specified.

Data Type

Integer (long)

See Also

Properties:

LastArticle

Methods:

XOver

FollowUpTo Property

Description

Specifies the newsgroup(s) for followup posts.

Usage

*object.***FollowUpTo**[= *string*]

Remarks

When someone posts a news message they may specify that followups are posted to one or more other news groups. Your program should use the contents of the FollowUpTo property for responses when it is not an empty string, otherwise you will usually just post responses to the newsgroups from which the message being responded to came.

Data Type

String

From Property

Description

Provides the contents of a messages From: header line.

Usage

*object.***From**

Remarks

From is read-only.   Normally a messages From: line contains the email address of the message author.

Data Type

String

GetAllGroups Method

See Also

Description

Retrieves a list of all valid news groups from the server.

Syntax

*object.***GetAllGroups**

Part	Description
object	Required. A News control.

Remarks

Each newsgroup is sent from the news server as a line of text in the following format:

```
group last first p
```

where <group> is the name of the newsgroup, <last> is the number of the last known article currently in that newsgroup, <first> is the number of the first article currently in the newsgroup, and <p> is either 'y' or 'n' indicating whether posting to this newsgroup is allowed ('y') or prohibited ('n').

The <first> and <last> fields will always be numeric. They may have leading zeros. If the <last> field evaluates to less than the <first> field, there are no articles currently on file in the newsgroup.

Note that posting may still be prohibited to a client even though the GetAllGroups command indicates that posting is permitted to a particular newsgroup. See the PostArticlecommand for an explanation of client prohibitions. The posting flag exists for each newsgroup because some newsgroups are moderated or are digests, and therefore cannot be posted to; that is, articles posted to them must be mailed to a moderator who will post them for the submitter. This is independent of the posting permission granted to a client by the NNTP server.

If Blockingis set to False the DataReady event will be fired as the list of groups arrive.

For both the blocking and non-blocking cases the GroupCount and Groups properties will be set after the list of groups is received. Each element of the Groups property array will contain a line formatted as discussed above.

See Also

Properties:

- GroupCount
- Groups

Events:

- DataReady
- Done

GetArticle Method

See Also

Description

Retrieves a specific article from the news server.

Syntax

*object.***GetArticle**

Part	Description
object	Required. A News control.

Remarks

There are two forms to the GetArticle command (and the related GetBody, GetHeader, and GetStatus commands), each using a different method of specifying which article is to be retrieved. When ArticleID contains a message-id in angle brackets ("<" and ">"), the first form of the command is used; when a numeric parameter is supplied, the second form is invoked. The text of the article is returned in BodyText and the article header is returned in HeaderText. The XOverHeaders and XOverHeadersCount properties can be used to access the individual header lines without parsing the HeaderText.

See Also

Properties:

- ArticleID
- BodyText
- HeaderText
- XOverHeaders
- XOverHeadersCount

Events:

- Done

Methods:

- GetBody
- GetHeader
- GetStatus

GetBody Method

See Also

Description

Retrieves the Body of a specific article from the news server.

Syntax

*object.***GetBody**

Part	Description
object	Required. A News control.

Remarks

There are two forms to the GetBody command (and the related GetArticle, GetHeader, and GetStatus commands), each using a different method of specifying which article is to be retrieved. When ArticleID contains a message-id in angle brackets ("<" and ">"), the first form of the command is used; when a numeric parameter is supplied, the second form is invoked. The text of the article body is returned in BodyText.

See Also

Properties:

- ArticleID
- BodyText

Events:

- Done

Methods:

- GetArticle
- GetHeader

GetHeader Method

See Also

Description

Retrieves the header for a specified article.

Syntax

*object.***GetHeader**

Part	Description
object	Required. A News control.

Remarks

There are two forms to the GetHeader command (and the related GetArticle, GetBody, and GetStatus commands), each using a different method of specifying which article is to be retrieved. When ArticleID contains a message-id in angle brackets ("<" and ">"), the first form of the command is used; when a numeric parameter is supplied, the second form is invoked. The text of the article body is returned in HeaderText. The XOverHeaders and XOverHeadersCount properties can be used to access the individual header lines without parsing the HeaderText.

See Also

Properties:

- ArticleID
- HeaderText
- XOverHeaders
- XOverHeadersCount

Events:

- Done

Methods:

- GetArticle
- GetBody

GetHelp Method

See Also

Description

Reads the news servers help information.

Syntax

*object.***GetHelp**

Part	Description
object	Required. A News control.

Remarks

Most news servers can provide a brief help file describing the supported command set. You can retrieve a servers help file by invoking the GetHelp method, the resulting help information is returned in the BodyText property and if Blocking is False the Done event is fired upon completion. Also, as with other methods which may return a lot of information, the DataReady event is fired as data arrives.

See Also

Properties:

- BodyText

Events:

- Done
- DataReady

GetNewGroups Method

See Also

Description

Retrieves a list of news groups created after the date specified by the Date property.

Syntax

*object.***GetNewGroups**

Part	Description
object	Required. A News control.

Remarks

Only those groups created after the date and time specified by the Date property are returned by this method. All groups on a server can be retrieved using the GetAllGroups method.

Each newsgroup is sent from the news server as a line of text in the following format:

```
group last first p
```

where <group> is the name of the newsgroup, <last> is the number of the last known article currently in that newsgroup, <first> is the number of the first article currently in the newsgroup, and <p> is either 'y' or 'n' indicating whether posting to this newsgroup is allowed ('y') or prohibited ('n').

The <first> and <last> fields will always be numeric. They may have leading zeros. If the <last> field evaluates to less than the <first> field, there are no articles currently on file in the newsgroup.

Note that posting may still be prohibited to a client even though the GetNewGroups command indicates that posting is permitted to a particular newsgroup. See the PostArticle command for an explanation of client prohibitions. The posting flag exists for each newsgroup because some newsgroups are moderated or are digests, and therefore cannot be posted to; that is, articles posted to them must be mailed to a moderator who will post them for the submitter. This is independent of the posting permission granted to a client by the NNTP server.

If Blocking is set to False the DataReady event will be fired as the list of groups arrive.

For both the blocking and non-blocking cases the GroupCount and Groups properties will be set after the list of groups is received. Each element of the Groups property array will contain a line formatted as discussed above.

See Also

Properties:

- Date

Events:

- DataReady
- Done

Methods:

- GetAllGroups

GetNewNews Method

See Also

Description

Retrieves the article ids for articles posted to the specified NewsGroups property after the date and time specified by the Date property.

Syntax

*object.***GetNewNews**

Part	Description
object	Required. A News control.

Remarks

As the new article Ids are received the ArticleIDs property array and ArticleIDsCount properties are updated and the DataReady event is fired for each block of received ArticleIDs.   When all of the article IDs have been received the Done event is fired.

You can retrieve article IDs from more than one group at a time by assigning a comma-separated list of newsgroups to the NewsGroups property.   To retrieve IDs for a single group assign its name without a trailing comma to the NewsGroups property.

See Also

Properties:

- ArticleIDs
- ArticleIDsCount
- NewsGroups

Events:

- Done
- DataReady

Methods:

- XOver

GetStatus Method

See Also

Description

Sets the servers current article number for the newsgroup named by the NewsGroups property.

Syntax

*object.***GetStatus**

Part	Description
object	Required. A News control.

Remarks

The GetStatus command is similar to the GetArticle command except that no text is returned.   When selecting by message number within a group, the GetStatus command serves to set the current article pointer without sending text. The ArticleID property will be updated so that it contains the selected message-id.   Using the GetStatus command to select by message-id is valid but of questionable value, since a selection by message-id does NOT alter the servers "current article pointer".   GetStatus is intended to be used with the NextArticle and PreviousArticle methods with GetStatus being used to establish the servers current article pointer and NextArticle and PreviousArticle being used to move to other articles.

See Also

Properties:

- NewsGroups

Events:

- Done

Methods:

- NextArticle
- PreviousArticle

Group Property

See Also

Description

Used to specify the Group to be used by methods which require a single newsgroup name.

Usage

*object.***Group**[= *string*]

Remarks

The GetNewNews and SelectGroup methods require a single newsgroup name as an argument.

Data Type

String

See Also

Methods:

- GetNewNews
- SelectGroup

GroupCount Property

See Also

Description

Number of elements in the Groups property array.

Usage

*object.***GroupCount**

Data Type

Integer (long)

See Also

Properties:

- GroupCount

Methods:

- GetAllGroups
- GetNewGroups

Groups Property

See Also

Description

Property array containing a collection of newsgroup names.

Usage

*object.***Groups(** *index* **)**

Remarks

The GetAllGroups and GetNewGroups methods retrieve a collection of newsgroup names from the news server. The Groups property array contains the retrieved newsgroups information. See the GetAllGroups for details.

Data Type

String

See Also

Properties:

GroupCount

Methods:

GetAllGroups

GetNewGroups

Headers Property

See Also

Description

Property array used with message headers.

Usage

*object.***Headers(** *index* **)**[= *string*]

Remarks

Messages have a number of required headers, such as Subject: and From: and numerous optional headers which can vary from user to user.

When a message is received the header is parsed and each line of the header is placed in a separate element of the Headers property array.

Before posting a message your program can store as many headers as required in the Headers property array and they will be sent when the message is posted.

An important thing to know about the Headers property is that you may set the element at Headers(HeadersCount). Normally on a zero-based array such as Headers you would only be able to set the elements 0 through Count-1. The Headers array works differently so that you can easily empty the array by setting HeadersCount to zero and then add a number of items to the array without having to count them in advance.

Data Type

String

See Also

Properties:

HeadersCount

Methods:

GetArticle

GetHeader

Post

HeadersCount Property

See Also

Description

Number of elements in the Headers property array.

Usage

*object.***HeadersCount**[= *integer*]

Remarks

Unlike most array count properties you can write to the HeadersCount property.   Writing and number to the HeadersCount property destroys the contents of the Headers property array and creates a new headers array of the specified size.

An important thing to know about the Headers property is that you may set the element at Headers(HeadersCount).   Normally on a zero-based array such as Headers you would only be able to set the elements 0 through Count-1.   The Headers array works differently so that you can easily empty the array by setting HeadersCount to zero and then add a number of items to the array without having to count them in advance.

Data Type

Integer (long)

See Also

Properties:

Headers

HeaderText Property

Description

When a message header is retrieved the header is stored in HeaderText.

Usage

*object.***HeaderText**

Data Type

String

Host Property

Description

IP address or name of a news server.

Usage

*object.***Host**[= *string*]

Remarks

The Host property must be set to a valid news server address prior to using the Connect method.   You may use either a name such as msnews.microsoft.com or the corresponding ip address.

Data Type

LastArticle Property

See Also

Description

Specifies the last article in a range of articles.

Usage

*object.***LastArticle**[= *integer*]

Remarks

Used with methods which return a range of articles or require that a range be specified.

Data Type

Integer (long)

See Also

Properties:

- FirstArticle

Methods:

- GetStatus
- XOver

Lines Property

Description

Number of lines in the article body of a received message.

Usage

*object.***Lines**

Remarks

Read only.

Data Type

Integer (long)

LogonName Property

See Also

Description

Logon name for servers which require user authentication.

Usage

*object.***LogonName**[= *string*]

Remarks

Many news servers require a user name and password before you can read and/or send messages using that server.   See the Connect method for further information.

Data Type

String

See Also

Methods:

Post

LogonPassword Property

See Also

Description

Password for servers which require user authentication.

Usage

*object.***LogonPassword**[= *string*]

Remarks

Many news servers require a user name and password before you can read and/or send messages using that server.   See the Connect method for further information.

Data Type

String

See Also

Properties:

LogonName

NewsGroups Property

See Also

Description

Message news groups.

Usage

*object.***NewsGroups**[= *string*]

Remarks

News messages may be posted to one or more newsgroups, multiple newsgroup names are separated by commas.

Data Type

String

See Also

Methods:

- GetHeader
- GetArticle
- Post

NextArticle Method

See Also

Description

Positions the servers current article pointer to the next article in the selected newsgroup property.

Syntax

*object.***NextArticle**

Part	**Description**
object	Required. A News control.

Remarks

Before using the NextArticle method you must use the SelectGroup method to set the selected group. Once a group has been selected the NextArticle, PreviousArticle, and GetStatus methods may be used to navigate the set of articles in the selected newsgroup.

If Blocking is True and there is no next article in the named newsgroup then an error will be returned. If Blocking is not true and there is no next article the error will be returned by way of the AsyncError event.   Upon successful completion of the NextArticle method the ArticleID property will be contain the articles ID.

See Also

Properties:

- ArticleID

Events:

- AsyncError
- Done

Methods:

- GetStatus
- NextArticle

Organization Property

Description

Contents of a messages Organization header line.

Usage

*object.***Organization**[= *string*]

Remarks

If a received message has an Organization header line the organization name is stored in the Organization property.

Data Type

String

PostArticle Method

See Also

Description

Posts an article to the newsgroups(s) specified by the NewsGroups property.

Syntax

*object.***PostArticle**

Part	Description
object	Required. A News control.

Remarks

You must set the following properties to valid values before invoking the PostArticle method:

PostingHost

EMailAddress

Additionally you must either assign values to the following proporties or a supply a correctly formatted (see RFC977) header line for each in the Headers property array.

NewsGroups

Subject

Message- ID

Date

References

For the above listed properties you do not need to clear a property in order for the corresponding Headers item to be used. For instance if the Subject property contains foo and you add an item Subject: fubar to the Headers property array, then Subject: fubar will be sent and not Subject: foo. If no line beginning with Subject: is in the Headers array then the contents of the Subject property will be appended to Subject: and the resulting string is sent as the messages subject.

The References property must be left blank for new messages and must contain the information specified by RFC977 for responses.

All other properties must be formatted as required by RFC977.

See Also

Properties:

- PostingHost
- EMailAddress
- LogonName
- NewsGroups
- Subject
- ArticleID
- Date
- References

Events:

- AsyncError
- Done

PostingHost Property

Description

Host name of the client machine.

Usage

*object.***PostingHost**[= *string*]

Remarks

When posting articles NNTP requires that the posting host identify itself, thats what the PostingHost property is for.   For instance if your machine-name was zaniac.telebyte.net then that would be what you would assign to the PostingHost property.

Data Type

String

PreviousArticle Method

See Also

Description

Positions the servers current article pointer to the previous article in the currently selected newsgroup.

Syntax

*object.***PreviousArticle**

Part	Description
object	Required. A News control.

Remarks

Before using the NextArticle method you must select a valid newsgroup with the SelectGroup method.

If Blocking is True and there is no previous article in the named newsgroup then an error will be returned.   If Blocking is not true and there is no previous article the error will be returned by way of the AsyncError event.   Upon successful completion of the PreviousArticle method the ArticleID property will be contain the articles ID.

See Also

Properties:

- ArticleID

Events:

- AsyncError
- Done

Methods:

- GetStatus
- NextArticle

Read Method

See Also

Description

Used to do raw reads of server responses.

Syntax

*object.***Read**

Part	Description
object	Required. A News control.

Remarks

The News control has a pair of methods, Read and Write, and corresponding properties ReadData and WriteData which can be used to communicate directly with the news server. These methods and properties are provided so that you can handle non-standard protocols which may be implemented on special-purpose news servers.

See Also

Properties:

- ReadData
- WriteData

Methods:

- Write

ReadData Property

See Also

Description

Contains server response when Write / Read methods are being used for direct access.

Usage

*object.***ReadData**[= *string*]

Remarks

This buffer only has data after a successful Read command.

Data Type

String

See Also

Properties:

WriteData

Methods:

Read

Write

References Property

Description

List of ArticleIDs preceding the current article in a thread.

Usage

*object.***References**[= *string*]

Remarks

The References property can be used to piece together the thread to which the current message belongs.   Its important to remember that no NNTP character string may exceed 500 characters and so referenced ArticleIDs will disappear off the end of the list.   Also, be sure that when you prepend your new ArticleIDs to the list of references belonging to a message youre responding to that you strip off the right end of the string any ArticleIDs which make the text in References longer than 500 characters.

Data Type

String

ReplyTo Property

Description

E-mail address of message poster.

Usage

*object.***ReplyTo**[= *string*]

Remarks

Normally this property contains the e-mail address of the person posting a news message, however most news reader software allows you to put invalid information in this field

Data Type

String

SelectGroup Method

See Also

Description

Selects the newsgroup specified by the NewsGroups property.

Syntax

*object.***SelectGroup**

Part	Description
object	Required. A News control.

Remarks

SelectGroup is used to select a specific newsgroup so that subsequent operations which require a selected group, such as message retrieval by ArticleID, can be exectued.

See Also

Properties:

- NewsGroups

Events:

- AsyncError
- Done

Subject Property

Description

Message subect.

Usage

*object.***Subject**[= *string*]

Remarks

If you respond to a message the NNTP protocol requires that you prepend the original Subject property with a Re: .   So, if the user of your program responds to a message with the subject foo then you should store Re: foo in the Subject property (or Subject: Re: foo as an element of the Headers array) before Posting the response.

Data Type

String

Timeout Property

Description

Determines how long the control will wait for an operation.

Usage

*object.***Timeout**[= *integer*]

Remarks

This propety determines how long the News control will wait for various actions / methods (before declaring an error). This property is measured in milliseconds.

Data Type

Integer

Version Property

Description

Shows the version of the control.

Usage

*object.***Version**

Remarks

This property holds the current version of the control.   It is read-only and available at both design-time and run-time.

Data Type

String

Write Method

See Also

Description

Sends a string to the server.

Syntax

*object.***Write**

Part	Description
object	Required. A News control.

Remarks

The News control has a pair of methods, Read and Write, and corresponding properties ReadData and WriteData which can be used to communicate directly with the news server.   These methods and properties are provided so that you can handle non-standard protocols which may be implemented on special-purpose news servers.

All strings must be terminated with a CR/LF (Chr(13) & Chr(10)) as part of the string.

See Also

Properties:

- ReadData
- WriteData

Methods:

- Read

WriteData Property

See Also

Description

Used with the Write method to send custom commands to the server.

Usage

*object.***WriteData**[= *string*]

Data Type

String

See Also

Properties:

ReadData

Events:

Done

Methods:

Read

Write

XOver Method

See Also

Description

Retrieves a list of headers from the server.

Syntax

*object.***XOver**

Part	Description
object	Required. A News control.

Remarks

Subsequent to RFC977 a number of enhancements have been made to news servers.   The XOver command is a near-universally supported news server enhancement, and with good reason since it drastically reduces the amount of time required to retrieve headers for a news group.

The XOver method retrieves a list of headers for articles numbered FirstArticle thru LastArticle in the current newsgroup.   The headers are stored in the XOverHeaders array and XOverHeadersCount is set to the number of items in the array.

Each element of the XOverHeaders array contains a line of text having the following tab-seperated fields: subject, author, date, message-id, references, byte count, and line count. Other optional fields may follow line count. Where no data exists, a null field is provided (i.e., the text will have two tab characters adjacent to each other).

See Also

Properties:

- FirstArticle
- LastArticle
- XOverHeaders
- XOverHeadersCount

Events:

- Done
- DataReady
- AsyncError

XOverHeaders Property

See Also

Description

XOverHeaders returned from the XOver command.

Usage

*object.***XOverHeaders(** *index* **)**

Remarks

This property holds the text of the headers returned by the XOver command. Each line of header is held in a separate element of this property array. You can determine the number of lines present by accessing the XOverHeadersCount property.

This property is read-only and available only at run-time.

Data Type

String

See Also

Properties:

XOverHeadersCount

Methods:

XOver

XOverHeadersCount Property

See Also

Description

Number of headers in the XOverHeaders property array.

Usage

*object.***XOverHeadersCount**[= *integer*]

Remarks

This property holds the number of headeer lines in the XOverHeaders property array.   This property is meaningful only after a successful XOver command.

This property is read-only and available only at run-time.

Data Type

Integer (long)

See Also

Properties:

XOverHeaders

Methods:

XOver

Registration Information

Credits

News was written by Zane Thomas. Inquiries can be sent to 71231,2066 on CompuServe, or mabry@mabry.com on Internet. Our mailing address is:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA 98103-1926

Registration

You can register this program by sending $35 ($40 for international orders) and your address. You can register News **and** its C source code by sending $90 ($95 for international orders). With your order, you will receive a copy of our manual documenting all of our controls.

Add $5 per order for shipping and handling.

For your convenience, an order form has been provided that you can print out directly from this help file.

Prices are subject to change without notice.

E-mail Discount

You may take a $5 discount for e-mail delivery of this package (CompuServe or Internet). If you choose this option, please note: a printed manual is not included. Be sure to include your full mailing address with your order. Sometimes (on the Internet) the package cannot be e-mailed. So, we are forced to send it through the normal mails.

CompuServe members may also take the $5 e-mail discount by registering this package in the software registration forum (GO SWREG). News SWREG ID number is 6964. The source code version's ID number is 9064. PLEASE NOTE: When you order through SWREG, we send the registered package to your CompuServe account (not your Internet or AOL account) within a few hours.

Credit Card Orders

We accept VISA, Mastercard and American Express. If you e-mail your order to us, please be sure to include your card number, expiration date, complete mailing address, and your phone number (in case we have any questions about your order).

News Order Form

Use the Print Topic.. command from the File menu to print this order form.

Mail this form to:	Mabry Software, Inc. Post Office Box 31926 Seattle, WA 98103-1926 Phone: 206-634-1443 Fax: 206-632-0272 CompuServe: 71231,2066 Internet: mabry@mabry.com Web: www.mabry.com

Where did you get this copy of News?

____

Ship to: ____

____

____

____

____

Phone: ____

Fax: ____

E-Mail: ____

MC/VISA/AMEX: ________________________________ exp. ______________

P.O. # (if any):

____

qty ordered	____	REGISTRATION $35 each ($40 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.
qty ordered	____	SOURCE CODE AND REGISTRATION $90 each ($95 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.

RFC 977

Network Working Group Brian Kantor (U.C. San Diego)

Request for Comments: 977 Phil Lapsley (U.C. Berkeley) - February 1986

Network News Transfer Protocol - A Proposed Standard for the Stream-Based Transmission of News

Status of This Memo

NNTP specifies a protocol for the distribution, inquiry, retrieval, and posting of news articles using a reliable stream-based transmission of news among the ARPA-Internet community. NNTP is designed so that news articles are stored in a central database allowing a subscriber to select only those items he wishes to read. Indexing, cross-referencing, and expiration of aged messages are also provided. This RFC suggests a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

1. Introduction

For many years, the ARPA-Internet community has supported the distribution of bulletins, information, and data in a timely fashion to thousands of participants. We collectively refer to such items of information as "news". Such news provides for the rapid dissemination of items of interest such as software bug fixes, new product reviews, technical tips, and programming pointers, as well as rapid-fire discussions of matters of concern to the working computer professional. News is very popular among its readers.

There are popularly two methods of distributing such news: the Internet method of direct mailing, and the USENET news system.

1.1. Internet Mailing Lists

The Internet community distributes news by the use of mailing lists. These are lists of subscriber's mailbox addresses and remailing sublists of all intended recipients. These mailing lists operate by remailing a copy of the information to be distributed to each subscriber on the mailing list. Such remailing is inefficient when a mailing list grows beyond a dozen or so people, since sending a separate copy to each of the subscribers occupies large quantities of network bandwidth, CPU resources, and significant amounts of disk storage at the destination host. There is also a significant problem in maintenance of the list itself: as subscribers move from one job to another; as new subscribers join and old ones leave; and as hosts come in and out of service.

1.2. The USENET News System

Clearly, a worthwhile reduction of the amount of these resources used can be achieved if articles are stored in a central database on the receiving host instead of in each subscriber's mailbox. The USENET news system provides a method of doing just this. There is a central repository of the news articles in one place (customarily a spool directory of some sort), and a set of programs that allow a subscriber to select those items he wishes to read. Indexing, cross-referencing, and expiration of aged messages are also provided.

1.3. Central Storage of News

For clusters of hosts connected together by fast local area networks (such as Ethernet), it makes even more sense to consolidate news distribution onto one (or a very few) hosts, and to allow access to these news articles using a server and client model. Subscribers may then request only the articles they wish to see, without having to wastefully duplicate the storage of a copy of each item on each host.

1.4. A Central News Server

A way to achieve these economies is to have a central computer system that can provide news service to the other systems on the local area network. Such a server would manage the collection of news articles and index files, with each person who desires to read news bulletins doing so over the LAN. For a large cluster of computer systems, the savings in total disk space is clearly worthwhile. Also, this allows workstations with limited disk storage space to participate in the news without incoming items consuming oppressive amounts of the workstation's disk storage.

We have heard rumors of somewhat successful attempts to provide centralized news service using IBIS and other shared or distributed file systems. While it is possible that such a distributed file system implementation might work well with a group of similar computers running nearly identical operating systems, such a scheme is not general enough to offer service to a wide range of client systems, especially when many diverse operating systems may be in use among a group of clients. There are few (if any) shared or networked file systems that can offer the generality of service that stream connections using Internet TCP provide, particularly when a wide range of host hardware and operating systems are considered.

NNTP specifies a protocol for the distribution, inquiry, retrieval, and posting of news articles using a reliable stream (such as TCP) server-client model. NNTP is designed so that news articles need only be stored on one (presumably central) host, and subscribers on other hosts attached to the LAN may read news articles using stream connections to the news host.

NNTP is modelled upon the news article specifications in RFC 850, which describes the USENET news system. However, NNTP makes few demands upon the structure, content, or storage of news articles, and thus we believe it easily can be adapted to other non-USENET news systems.

Typically, the NNTP server runs as a background process on one host, and would accept connections from other hosts on the LAN. This works well when there are a number of small computer systems (such as workstations, with only one or at most a few users each), and a large central server.

1.5. Intermediate News Servers

For clusters of machines with many users (as might be the case in a university or large industrial environment), an intermediate server might be used. This intermediate or "slave" server runs on each computer system, and is responsible for mediating news reading requests and performing local caching of recently-retrieved news articles.

Typically, a client attempting to obtain news service would first attempt to connect to the news service port on the local machine. If this attempt were unsuccessful, indicating a failed server, an installation might choose to either deny news access, or to permit connection to the central "master" news server.

For workstations or other small systems, direct connection to the master server would probably be the normal manner of operation.

This specification does not cover the operation of slave NNTP servers. We merely suggest that slave servers are a logical addition to NNTP server usage which would enhance operation on large local area networks.

1.6. News Distribution

NNTP has commands which provide a straightforward method of exchanging articles between cooperating hosts. Hosts which are well connected on a local area or other fast network and who wish to actually obtain copies of news articles for local storage might well find NNTP to be a more efficient way to distribute news than more traditional transfer methods (such as UUCP).

In the traditional method of distributing news articles, news is propagated from host to host by flooding - that is, each host will send all its new news articles on to each host that it feeds. These hosts will then in turn send these new articles on to other hosts that they feed. Clearly, sending articles that a host already has obtained a copy of from another feed (many hosts that receive news are redundantly fed) again is a waste of time and communications resources, but for transport mechanisms that are single-transaction based rather than interactive (such as UUCP in the UNIX-world <1>), distribution time is diminished by sending all articles and having the receiving host simply discard the duplicates. This is an especially true when communications sessions are limited to once a day.

Using NNTP, hosts exchanging news articles have an interactive mechanism for deciding which articles are to be transmitted. A host desiring new news, or which has new news to send, will typically contact one or more of its neighbors using NNTP. First it will inquire if any new news groups have been created on the serving host by means of the NEWGROUPS command. If so, and those are appropriate or desired (as established by local site-dependent rules), those new newsgroups can be created.

The client host will then inquire as to which new articles have arrived in all or some of the newsgroups that it desires to receive, using the NEWNEWS command. It will receive a list of new articles from the server, and can request transmission of those articles that it desires and does not already have.

Finally, the client can advise the server of those new articles which the client has recently received. The server will indicate those articles that it has already obtained copies of, and which articles should be sent to add to its collection.

In this manner, only those articles which are not duplicates and which are desired are transferred.

2. The NNTP Specification

2.1. Overview

The news server specified by this document uses a stream connection (such as TCP) and SMTP-like commands and responses. It is designed to accept connections from hosts, and to provide a simple interface to the news database.

This server is only an interface between programs and the news databases. It does not perform any user interaction or presentation-level functions. These "user-friendly" functions are better left to the client programs, which have a better understanding of the environment in which they are operating.

When used via Internet TCP, the contact port assigned for this service is 119.

2.2. Character Codes

Commands and replies are composed of characters from the ASCII character set. When the transport service provides an 8-bit byte (octet) transmission channel, each 7-bit character is transmitted right justified in an octet with the high order bit cleared to zero.

2.3. Commands

Commands consist of a command word, which in some cases may be followed by a parameter. Commands with parameters must separate the parameters from each other and from the command by one or more space or tab characters. Command lines must be complete with all required parameters, and may not contain more than one command.

Commands and command parameters are not case sensitive. That is, a command or parameter word may be upper case, lower case, or any mixture of upper and lower case.

Each command line must be terminated by a CR-LF (Carriage Return - Line Feed) pair.

Command lines shall not exceed 512 characters in length, counting all characters including spaces, separators, punctuation, and the trailing CR-LF (thus there are 510 characters maximum allowed for the command and its parameters). There is no provision for continuation command lines.

2.4. Responses

Responses are of two kinds, textual and status.

2.4.1. Text Responses

Text is sent only after a numeric status response line has been sent that indicates that text will follow. Text is sent as a series of successive lines of textual matter, each terminated with CR-LF pair. A single line containing only a period (.) is sent to indicate the end of the text (i.e., the server will send a CR-LF pair at the end of the last line of text, a period, and another CR-LF pair).

If the text contained a period as the first character of the text line in the original, that first period is doubled. Therefore, the client must examine the first character of each line received, and for those beginning with a period, determine either that this is the end of the text or whether to collapse the doubled period to a single one.

The intention is that text messages will usually be displayed on the user's terminal whereas command/status responses will be interpreted by the client program before any possible display is done.

2.4.2. Status Responses

These are status reports from the server and indicate the response to the last command received from the client.

Status response lines begin with a 3 digit numeric code which is sufficient to distinguish all responses. Some of these may herald the subsequent transmission of text.

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

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
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, below is listed a general set of response codes that may be received at any time.

Certain status responses contain parameters such as numbers and names. The number and type of such parameters is fixed for each response code to simplify interpretation of the response.

Parameters are separated from the numeric response code and from each other by a single space. All numeric parameters are decimal, and may have leading zeros. All string parameters begin after the separating space, and end before the following separating space or the CR-LF pair at the end of the line. (String parameters may not, therefore, contain spaces.) All text, if any, in the response which is not a parameter of the response must follow and be separated from the last parameter by a space. Also, note that the text following a response number may vary in different implementations of the server. The 3-digit numeric code should be used to determine what response was sent.

Response codes not specified in this standard may be used for any installation-specific additional commands also not specified. These should be chosen to fit the pattern of x8x specified above. (Note that debugging is provided for explicitly in the x9x response codes.) The use of unspecified response codes for standard commands is prohibited.

We have provided a response pattern x9x for debugging. Since much debugging output may be classed as "informative messages", we would expect, therefore, that responses 190 through 199 would be used for various debugging outputs. There is no requirement in this specification for debugging output, but if such is provided over the connected stream, it must use these response codes. If appropriate to a specific implementation, other x9x codes may be used for debugging. (An example might be to use e.g., 290 to acknowledge a remote debugging request.)

2.4.3. General Responses

The following is a list of general response codes that may be sent by the NNTP server. These are not specific to any one command, but may be returned as the result of a connection, a failure, or some unusual condition.

In general, 1xx codes may be ignored or displayed as desired; code 200 or 201 is sent upon initial connection to the NNTP server depending upon posting permission; code 400 will be sent when the NNTP server discontinues service (by operator request, for example); and 5xx codes indicate that the command could not be performed for some unusual reason.

100 help text

190 through 199 debug output

200 server ready - posting allowed

201 server ready - no posting allowed

400 service discontinued

500 command not recognized

501 command syntax error

502 access restriction or permission denied

503 program fault - command not performed

3. Command and Response Details

On the following pages are descriptions of each command recognized by the NNTP server and the responses which will be returned by those commands.

Each command is shown in upper case for clarity, although case is ignored in the interpretation of commands by the NNTP server. Any parameters are shown in lower case. A parameter shown in [square brackets] is optional. For example, [GMT] indicates that the triglyph GMT may present or omitted.

Every command described in this section must be implemented by all NNTP servers. There is no prohibition against additional commands being added; however, it is recommended that any such unspecified command begin with the letter "X" to avoid conflict with later revisions of this specification.

Implementors are reminded that such additional commands may not redefine specified status response codes. Using additional unspecified responses for standard commands is also prohibited.

3.1. The ARTICLE, BODY, HEAD, and STAT commands

There are two forms to the ARTICLE command (and the related BODY, HEAD, and STAT commands), each using a different method of specifying which article is to be retrieved. When the ARTICLE command is followed by a message-id in angle brackets ("<" and ">"), the first form of the command is used; when a numeric parameter or no parameter is supplied, the second form is invoked.

The text of the article is returned as a textual response, as described earlier in this document.

The HEAD and BODY commands are identical to the ARTICLE command except that they respectively return only the header lines or text body of the article.

The STAT command is similar to the ARTICLE command except that no text is returned. When selecting by message number within a group, the STAT command serves to set the current article pointer without sending text. The returned acknowledgement response will contain the message-id, which may be of some value. Using the STAT command to select by message-id is valid but of questionable value, since a selection by message-id does NOT alter the "current article pointer".

3.1.1. ARTICLE (selection by message-id)

```
ARTICLE <message-id>
```

Display the header, a blank line, then the body (text) of the specified article. Message-id is the message id of an article as shown in that article's header. It is anticipated that the client will obtain the message-id from a list provided by the NEWNEWS command, from references contained within another article, or from the message-id provided in the response to some other commands.

Please note that the internally-maintained "current article pointer" is NOT ALTERED by this command. This is both to facilitate the presentation of articles that may be referenced within an article being read, and because of the semantic difficulties of determining the proper sequence and membership of an article which may have been posted to more than one newsgroup.

3.1.2. ARTICLE (selection by number)

```
ARTICLE [nnn]
```

Displays the header, a blank line, then the body (text) of the current or specified article. The optional parameter nnn is the

numeric id of an article in the current newsgroup and must be chosen from the range of articles provided when the newsgroup was selected. If it is omitted, the current article is assumed.

The internally-maintained "current article pointer" is set by this command if a valid article number is specified.

[the following applies to both forms of the article command.] A response indicating the current article number, a message-id string, and that text is to follow will be returned.

The message-id string returned is an identification string contained within angle brackets ("<" and ">"), which is derived from the header of the article itself. The Message-ID header line (required by RFC850) from the article must be used to supply this information. If the message-id header line is missing from the article, a single digit "0" (zero) should be supplied within the angle brackets.

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

3.1.3. Responses

220 n <a> article retrieved - head and body follow (n = article number, <a> = message-id)

221 n <a> article retrieved - head follows

222 n <a> article retrieved - body follows

223 n <a> article retrieved - request text separately

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

3.2. The GROUP command

3.2.1. GROUP

```
GROUP ggg
```

The required parameter ggg is the name of the newsgroup to be selected (e.g. "net.news"). 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 should 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.

3.2.2. Responses

211 n f l s group selected (n = estimated number of articles in group, f = first article number in the group, l = last article number in the group, s = name of the group.)

411 no such news group

3.3. The HELP command

3.3.1. HELP

```
HELP
```

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.

3.3.2. Responses

100 help text follows

3.4. The IHAVE command

3.4.1. IHAVE

```
IHAVE <messageid>
```

The IHAVE command informs the server that the client has an article whose id is <messageid>. If the server desires a copy of that article, it will return a response instructing the client to send the entire article. If the server does not want the article (if, for example, the server already has a copy of it), a response indicating that the article is not wanted will be returned.

If transmission of the article is requested, the client should send the entire article, including header and body, in the manner specified for text transmission from the server. A response code indicating success or failure of the transferral of the article will be returned.

This function differs from the POST command in that it is intended for use in transferring already-posted articles between hosts. Normally it will not be used when the client is a personal newsreading program. In particular, this function will invoke the server's news posting program with the appropriate settings (flags, options, etc) to indicate that the forthcoming article is being forwarded from another host.

The server may, however, elect not to post or forward the article if after further examination of the article it deems it inappropriate to do so. The 436 or 437 error codes may be returned as appropriate to the situation.

Reasons for such subsequent rejection of an article may include such problems as inappropriate newsgroups or distributions, disk space limitations, article lengths, garbled headers, and the like. These are typically restrictions enforced by the server host's news software and not necessarily the NNTP server itself.

3.4.2. 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

An implementation note: Because some host news posting software may not be able to decide immediately that an article is inappropriate for posting or forwarding, it is acceptable to acknowledge the successful transfer of the article and to later silently discard it. Thus it is permitted to return the 235 acknowledgement code and later discard the received article. This is not a fully satisfactory solution to the problem. Perhaps some implementations will wish to send mail to the author of the article in certain of these cases.

3.5. The LAST command

3.5.1. LAST

```
LAST
```

The internally maintained "current article pointer" is set to the previous article in the current newsgroup. If already positioned at the first article of the newsgroup, an error message is returned and the current article remains selected.

The internally-maintained "current article pointer" is set by this command.

A response indicating the current article number, and a message-id string will be returned. No text is sent in response to this command.

3.5.2. Responses

223 n a article retrieved - request text separately (n = article number, a = unique article id)

412 no newsgroup selected

420 no current article has been selected

422 no previous article in this group

3.6. The LIST command

3.6.1. LIST

```
LIST
```

Returns a list of valid newsgroups and associated information. Each newsgroup is sent as a line of text in the following format:

group last first p

where <group> is the name of the newsgroup, <last> is the number of the last known article currently in that newsgroup, <first> is the number of the first article currently in the newsgroup, and <p> is either 'y' or 'n' indicating whether posting to this newsgroup is allowed ('y') or prohibited ('n').

The <first> and <last> fields will always be numeric. They may have leading zeros. If the <last> field evaluates to less than the <first> field, there are no articles currently on file in the newsgroup.

Note that posting may still be prohibited to a client even though the LIST command indicates that posting is permitted to a particular newsgroup. See the POST command for an explanation of client prohibitions. The posting flag exists for each newsgroup because some newsgroups are moderated or are digests, and therefore cannot be posted to; that is, articles posted to them must be mailed to a moderator who will post them for the submitter. This is independent of the posting permission granted to a client by the NNTP server.

Please note that an empty list (i.e., the text body returned by this command consists only of the terminating period) is a possible valid response, and indicates that there are currently no valid newsgroups.

3.6.2. Responses

215 list of newsgroups follows

3.7. The NEWGROUPS command

3.7.1. NEWGROUPS

```
NEWGROUPS date time [GMT] [<distributions>]
```

A list of newsgroups created since <date and time> will be listed in the same format as the LIST command.

The date is sent as 6 digits in the format YYMMDD, where YY is the last two digits of the year, MM is the two digits of the month (with leading zero, if appropriate), and DD is the day of the month (with leading zero, if appropriate). The closest century is assumed as part of the year (i.e., 86 specifies 1986, 30 specifies 2030, 99 is 1999, 00 is 2000).

Time must also be specified. It must be as 6 digits HHMMSS with HH being hours on the 24-hour clock, MM minutes 00-59, and SS seconds 00-59. The time is assumed to be in the server's timezone unless the token "GMT" appears, in which case both time and date are evaluated at the 0 meridian.

The optional parameter "distributions" is a list of distribution groups, enclosed in angle brackets. If specified, the distribution portion of a new newsgroup (e.g, 'net' in 'net.wombat') will be examined for a match with the distribution categories listed, and only those new newsgroups which match will be listed. If more than one distribution group is to be listed, they must be separated by commas within the angle brackets.

Please note that an empty list (i.e., the text body returned by this command consists only of the terminating period) is a possible valid response, and indicates that there are currently no new newsgroups.

3.7.2. Responses

231 list of new newsgroups follows

3.8. The NEWNEWS command

3.8.1. NEWNEWS

```
NEWNEWS newsgroups date time [GMT] [<distribution>]
```

A list of message-ids of articles posted or received to the specified newsgroup since "date" will be listed. The format of the listing will be one message-id per line, as though text were being sent. A single line consisting solely of one period followed by CR-LF will terminate the list.

Date and time are in the same format as the NEWGROUPS command.

A newsgroup name containing a "*" (an asterisk) may be specified to broaden the article search to some or all newsgroups. The asterisk will be extended to match any part of a newsgroup name (e.g., net.micro* will match net.micro.wombat, net.micro.apple, etc). Thus if only an asterisk is given as the newsgroup name, all newsgroups will be searched for new news.

(Please note that the asterisk "*" expansion is a general replacement; in particular, the specification of e.g., net.*.unix should be correctly expanded to embrace names such as net.wombat.unix and net.whocares.unix.)

Conversely, if no asterisk appears in a given newsgroup name, only the specified newsgroup will be searched for new articles. Newsgroup names must be chosen from those returned in the listing of available groups. Multiple newsgroup names (including a "*") may be specified in this command, separated by a comma. No comma shall appear after the last newsgroup in the list. [Implementors are cautioned to keep the 512 character command length limit in mind.]

The exclamation point ("!") may be used to negate a match. This can be used to selectively omit certain newsgroups from an otherwise larger list. For example, a newsgroups specification of "net.*,mod.*,!mod.map.*" would specify that all net.<anything> and all mod.<anything> EXCEPT mod.map.<anything> newsgroup names would be matched. If used, the exclamation point must appear as the first character of the given newsgroup name or pattern.

The optional parameter "distributions" is a list of distribution groups, enclosed in angle brackets. If specified, the distribution portion of an article's newsgroup (e.g, 'net' in 'net.wombat') will be examined for a match with the distribution categories listed, and only those articles which have at least one newsgroup belonging to the list of distributions will be listed. If more than one distribution group is to be supplied, they must be separated by commas within the angle brackets.

The use of the IHAVE, NEWNEWS, and NEWGROUPS commands to distribute news is discussed in an earlier part of this document.

Please note that an empty list (i.e., the text body returned by this command consists only of the terminating period) is a possible valid response, and indicates that there is currently no new news.

3.8.2. Responses

230 list of new articles by message-id follows

3.9. The NEXT command

3.9.1. NEXT

```
NEXT
```

The internally maintained "current article pointer" is advanced to the next article in the current newsgroup. If no more articles remain in the current group, an error message is returned and the current article remains selected.

The internally-maintained "current article pointer" is set by this command.

A response indicating the current article number, and the message-id string will be returned. No text is sent in response to this command.

3.9.2. Responses

223 n a article retrieved - request text separately (n = article number, a = unique article id)

412 no newsgroup selected

420 no current article has been selected

421 no next article in this group

3.10. The POST command

3.10.1. POST

```
POST
```

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

If posting is permitted, the article should be presented in the format specified by RFC850, and should include all required header lines. After the article's header and body have been completely sent by the client to the server, a further response code will be returned to indicate success or failure of the posting attempt.

The text forming the header and body of the message to be posted should be sent by the client using the conventions for text received from the news server: A single period (".") on a line indicates the end of the text, with lines starting with a period in the original text having that period doubled during transmission.

No attempt shall be made by the server to filter characters, fold or limit lines, or otherwise process incoming text. It is our intent that the server just pass the incoming message to be posted to the server installation's news posting software, which is separate from this specification. See RFC850 for more details.

Since most installations will want the client news program to allow the user to prepare his message using some sort of text editor, and transmit it to the server for posting only after it is composed, the client program should take note of the herald message that greeted it when the connection was first established. This message indicates whether postings from that client are permitted or not, and can be used to caution the user that his access is read-only if that is the case. This will prevent the user from wasting a good deal of time composing a message only to find posting of the message was denied. The method and determination of which clients and hosts may post is installation dependent and is not covered by this specification.

3.10.2. Responses

240 article posted ok

340 send article to be posted. End with <CR-LF>.<CR-LF>

440 posting not allowed

441 posting failed

(for reference, one of the following codes will be sent upon initial connection; the client program should determine whether posting is generally permitted from these:)

200 server ready - posting allowed

201 server ready - no posting allowed

3.11. The QUIT command

3.11.1. QUIT

```
QUIT
```

The server process acknowledges the QUIT command and then closes the connection to the client. This is the preferred method for a client to indicate that it has finished all its transactions with the NNTP server.

If a client simply disconnects (or the connection times out, or some other fault occurs), the server should gracefully cease its attempts to service the client.

3.11.2. Responses

205 closing connection - goodbye!

3.12. The SLAVE command

3.12.1. SLAVE

```
SLAVE
```

Indicates to the server that this client connection is to a slave server, rather than a user.

This command is intended for use in separating connections to single users from those to subsidiary ("slave") servers. It may be used to indicate that priority should therefore be given to requests from this client, as it is presumably serving more than one person. It might also be used to determine which connections to close when system load levels are exceeded, perhaps giving preference to slave servers. The actual use this command is put to is entirely implementation dependent, and may vary from one host to another. In NNTP servers which do not give priority to slave servers, this command must nonetheless be recognized and acknowledged.

3.12.2. Responses

202 slave status noted

4. Sample Conversations

These are samples of the conversations that might be expected with the news server in hypothetical sessions. The notation C: indicates commands sent to the news server from the client program; S: indicate responses received from the server by the client.

4.1. Example 1 - relative access with NEXT

```
S: (listens at TCP port 119)

C: (requests connection on TCP port 119)

S: 200 wombatvax news server ready - posting ok
```

(client asks for a current newsgroup list)

```
C: LIST
S: 215 list of newsgroups follows
S: net.wombats 00543 00501 y
S: net.unix-wizards 10125 10011 y
```

(more information here)

```
S: net.idiots 00100 00001 n
S: .
```

(client selects a newsgroup)

```
C: GROUP net.unix-wizards
S: 211 104 10011 10125 net.unix-wizards group selected
```

(there are 104 articles on file, from 10011 to 10125)

(client selects an article to read)

```
C: STAT 10110
S: 223 10110 <23445@sdcsvax.ARPA> article retrieved - statistics only
 (article 10110 selected, its message-id is <23445@sdcsvax.ARPA>)
```

(client examines the header)

```
C: HEAD
S: 221 10110 <23445@sdcsvax.ARPA> article retrieved - head
```

follows (text of the header appears here)

```
S: .
```

(client wants to see the text body of the article)

```
C: BODY
S: 222 10110 <23445@sdcsvax.ARPA> article retrieved - body follows (body
text here)
S: .
```

(client selects next article in group)

```
C: NEXT
S: 223 10113 <21495@nudebch.uucp> article retrieved - statistics only
 (article 10113 was next in group)
```

(client finishes session)

```
C: QUIT
S: 205 goodbye.
```

4.2. Example 2 - absolute article access with ARTICLE

```
S: (listens at TCP port 119)
C: (requests connection on TCP port 119)
S: 201 UCB-VAX netnews server ready -- no posting allowed
C: GROUP msgs
S: 211 103 402 504 msgs Your new group is msgs
```

(there are 103 articles, from 402 to 504)

```
C: ARTICLE 401
S: 423 No such article in this newsgroup
C: ARTICLE 402
S: 220 402 <4105@ucbvax.ARPA> Article retrieved, text follows
S: (article header and body follow)
S: .
C: HEAD 403
S: 221 403 <3108@mcvax.UUCP> Article retrieved, header follows
S: (article header follows)
S: .
C: QUIT
S: 205 UCB-VAX news server closing connection. Goodbye.
```

4.3. Example 3 - NEWGROUPS command

```
S: (listens at TCP port 119)
C: (requests connection on TCP port 119)
S: 200 Imaginary Institute News Server ready (posting ok)
```

(client asks for new newsgroups since April 3, 1985)

```
C: NEWGROUPS 850403 020000
S: 231 New newsgroups since 03/04/85 02:00:00 follow
S: net.music.gdead
S: net.games.sources
S: .
C: GROUP net.music.gdead
S: 211 0 1 1 net.music.gdead Newsgroup selected
```

(there are no articles in that newsgroup, and the first and last article numbers should be ignored)

```
C: QUIT
S: 205 Imaginary Institute news server ceasing service. Bye!
```

4.4. Example 4 - posting a news article

```
S: (listens at TCP port 119)
C: (requests connection on TCP port 119)
S: 200 BANZAIVAX news server ready, posting allowed.
C: POST
S: 340 Continue posting; Period on a line by itself to end
C: (transmits news article in RFC850 format)
C: .
S: 240 Article posted successfully.
```

```
C: QUIT

S: 205 BANZAIVAX closing connection. Goodbye.
```

4.5. Example 5 - interruption due to operator request

```
S: (listens at TCP port 119)

C: (requests connection on TCP port 119)

S: 201 genericvax news server ready, no posting allowed.
```

(assume normal conversation for some time, and that a newsgroup has been selected)

```
C: NEXT

S: 223 1013 <5734@mcvax.UUCP> Article retrieved; text separate.

C: HEAD

C: 221 1013 <5734@mcvax.UUCP> Article retrieved; head follows.

S: (sends head of article, but halfway through is interrupted by an
operator request. The following then occurs, without client
intervention.)

S: (ends current line with a CR-LF pair)

S: .

S: 400 Connection closed by operator. Goodbye.

S: (closes connection)
```

4.6. Example 6 - Using the news server to distribute news between systems.

```
S: (listens at TCP port 119)

C: (requests connection on TCP port 119)

S: 201 Foobar NNTP server ready (no posting)
```

(client asks for new newsgroups since 2 am, May 15, 1985)

```
C: NEWGROUPS 850515 020000

S: 235 New newsgroups since 850515 follow

S: net.fluff

S: net.lint

S: .
```

(client asks for new news articles since 2 am, May 15, 1985)

```
C: NEWNEWS * 850515 020000

S: 230 New news since 850515 020000 follows

S: <1772@foo.UUCP>

S: <87623@baz.UUCP>

S: <17872@GOLD.CSNET>

S: .
```

(client asks for article <1772@foo.UUCP>)

```
C: ARTICLE <1772@foo.UUCP>

S: 220 <1772@foo.UUCP> All of article follows
```

```
S: (sends entire message)
S: .
```

(client asks for article <87623@baz.UUCP>

```
C: ARTICLE <87623@baz.UUCP>
S: 220 <87623@baz.UUCP> All of article follows
S: (sends entire message)
S: .
```

(client asks for article <17872@GOLD.CSNET>

```
C: ARTICLE <17872@GOLD.CSNET>
S: 220 <17872@GOLD.CSNET> All of article follows
S: (sends entire message)
S: .
```

(client offers an article it has received recently)

```
C: IHAVE <4105@ucbvax.ARPA>
S: 435 Already seen that one, where you been?
```

(client offers another article)

```
C: IHAVE <4106@ucbvax.ARPA>
S: 335 News to me! <CRLF.CRLF> to end.
C: (sends article)
C: .
S: 235 Article transferred successfully. Thanks.
```

(or)

```
S: 436 Transfer failed.
```

(client is all through with the session)

```
C: QUIT
S: 205 Foobar NNTP server bids you farewell.
```

4.7. Summary of commands and responses.

The following are the commands recognized and responses returned by the NNTP server.

4.7.1. Commands

ARTICLE
BODY
GROUP
HEAD
HELP
IHAVE
LAST
LIST
NEWGROUPS
NEWNEWS
NEXT
POST

QUIT
SLAVE
STAT

4.7.2. Responses

100 help text follows
199 debug output
200 server ready - posting allowed
201 server ready - no posting allowed
202 slave status noted
205 closing connection - goodbye!
211 n f l s group selected
215 list of newsgroups follows
220 n <a> article retrieved - head and body follow 221 n <a> article retrieved - head follows
222 n <a> article retrieved - body follows
223 n <a> article retrieved - request text separately 230 list of new articles by message-id follows
231 list of new newsgroups follows
235 article transferred ok
240 article posted ok
335 send article to be transferred. End with <CR-LF>.<CR-LF>
340 send article to be posted. End with <CR-LF>.<CR-LF>
400 service discontinued
411 no such news group
412 no newsgroup has been selected
420 no current article has been selected
421 no next article in this group
422 no previous article in this group
423 no such article number in this group
430 no such article found
435 article not wanted - do not send it
436 transfer failed - try again later
437 article rejected - do not try again.
440 posting not allowed
441 posting failed
500 command not recognized
501 command syntax error
502 access restriction or permission denied
503 program fault - command not performed

4.8. A Brief Word about the USENET News System

In the UNIX world, which traditionally has been linked by 1200 baud dial-up telephone lines, the USENET News system has evolved to handle central storage, indexing, retrieval, and distribution of news. With the exception of its underlying transport mechanism (UUCP), USENET News is an efficient means of providing news and bulletin service to subscribers on UNIX and other hosts worldwide. The USENET News system is discussed in detail in RFC 850. It runs on most versions of UNIX and on many other operating systems, and is customarily distributed without charge.

USENET uses a spooling area on the UNIX host to store news articles, one per file. Each article consists of a series of heading text, which contain the sender's identification and organizational affiliation, timestamps, electronic mail reply paths, subject, newsgroup (subject category), and the like. A complete news article is reproduced in its entirety below. Please consult RFC 850 for more details.

```
Relay-Version: version B 2.10.3 4.3bsd-beta 6/6/85; site sdcsvax.UUCP
Posting-Version: version B 2.10.1 6/24/83 SMI; site unitek.uucp
Path:sdcsvax!sdcrdcf!hplabs!qantel!ihnp4!alberta!ubc-vision!unitek!honman
From: honman@unitek.uucp (Man Wong)
Newsgroups: net.unix-wizards
Subject: foreground -> background ?
Message-ID: <167@unitek.uucp>
Date: 25 Sep 85 23:51:52 GMT
Date-Received: 29 Sep 85 09:54:48 GMT
Reply-To: honman@unitek.UUCP (Hon-Man Wong)
Distribution: net.all
Organization: Unitek Technologies Corporation
Lines: 12
I have a process (C program) which generates a child and waits for
it to return. What I would like to do is to be able to run the
child process interactively for a while before kicking itself into
the background so I can return to the parent process (while the
child process is RUNNING in the background). Can it be done? And
if it can, how?
Please reply by E-mail. Thanks in advance.
Hon-Man Wong
```

5. References

[1] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", RFC-822, Department of Electrical Engineering, University of Delaware, August, 1982.

[2] Horton, M., "Standard for Interchange of USENET Messages", RFC-850, USENET Project, June, 1983.

[3] Postel, J., "Transmission Control Protocol- DARPA Internet Program Protocol Specification", RFC-793, USC/Information Sciences Institute, September, 1981.

[4] Postel, J., "Simple Mail Transfer Protocol", RFC-821, USC/Information Sciences Institute, August, 1982.

6. Acknowledgements

The authors wish to express their heartfelt thanks to those many people who contributed to this specification, and especially to Erik Fair and Chuq von Rospach, without whose inspiration this whole thing would not have been necessary.

7. Notes

<1> UNIX is a trademark of Bell Laboratories.

RFC 850

RFC 850 June 1983

Standard for Interchange of USENET Messages

Mark R. Horton

[This memo is distributed as an RFC only to make this information easily accessible to researchers in the ARPA community. It does not specify an Internet standard.]

1. Introduction

This document defines the standard format for interchange of Network News articles among USENET sites. It describes the format for articles themselves, and gives partial standards for transmission of news. The news transmission is not entirely standardized in order to give a good deal of flexibility to the individual hosts to choose transmission hardware and software, whether to batch news, and so on.

There are five sections to this document. Section two section defines the format. Section three defines the valid control messages. Section four specifies some valid transmission methods. Section five describes the overall news propagation algorithm.

2. Article Format

The primary consideration in choosing an article format is that it fit in with existing tools as well as possible. Existing tools include both implementations of mail and news. (The notesfiles system from the University of Illinois is considered a news implementation.) A standard format for mail messages has existed for many years on the ARPANET, and this format meets most of the needs of USENET. Since the ARPANET format is extensible, extensions to meet the additional needs of USENET are easily made within the ARPANET standard. Therefore, the rule is adopted that all USENET news articles must be formatted as valid ARPANET mail messages, according to the ARPANET standard RFC 822. This standard is more restrictive than the ARPANET standard, placing additional requirements on each article and forbidding use of certain ARPANET features. However, it should always be possible to use a tool expecting an ARPANET message to process a news article. In any situation where this standard conflicts with the ARPANET standard, RFC 822 should be considered correct and this standard in error.

An example message is included to illustrate the fields.

```
Relay-Version: version B 2.10 2/13/83; site cbosgd.UUCP
Posting-Version: version B 2.10 2/13/83; site eagle.UUCP
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
From: jerry@eagle.uucp (Jerry Schwarz)
Newsgroups: net.general
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.UUCP>
Date: Friday, 19-Nov-82 16:14:55 EST
Followup-To: net.news
Expires: Saturday, 1-Jan-83 00:00:00 EST
Date-Received: Friday, 19-Nov-82 16:59:30 EST
Organization: Bell Labs, Murray Hill
The body of the article comes here, after a blank line.
```

Here is an example of a message in the old format (before the existence of this standard). It is recommended that implementations also accept articles in this format to ease upward conversion.

```
From: cbosgd!mhuxj!mhuxt!eagle!jerry (Jerry Schwarz)
Newsgroups: net.general
```

```
   Title: Usenet Etiquette -- Please Read
   Article-I.D.: eagle.642
   Posted: Fri Nov 19 16:14:55 1982
   Received: Fri Nov 19 16:59:30 1982
   Expires: Mon Jan 1 00:00:00 1990
   The body of the article comes here, after a blank line.
```

Some news systems transmit news in the "A" format, which looks like this:

```
   Aeagle.642
   net.general
   cbosgd!mhuxj!mhuxt!eagle!jerry
   Fri Nov 19 16:14:55 1982
   Usenet Etiquette - Please Read
   The body of the article comes here, with no blank line.
```

An article consists of several header lines, followed by a blank line, followed by the body of the message. The header lines consist of a keyword, a colon, a blank, and some additional information. This is a subset of the ARPANET standard, simplified to allow simpler software to handle it. The "from" line may optionally include a full name, in the format above, or use the ARPANET angle bracket syntax. To keep the implementations simple, other formats (for example, with part of the machine address after the close parenthesis) are not allowed. The ARPANET convention of continuation header lines (beginning with a blank or tab) is allowed.

Certain headers are required, certain headers are optional. Any unrecognized headers are allowed, and will be passed through unchanged. The required headers are Relay-Version, Posting-Version, From, Date, Newsgroups, Subject, Message-ID, Path. The optional headers are Followup-To, Date-Received, Expires, Reply-To, Sender, References, Control, Distribution, Organization.

2.1 Required Headers

2.1.1 Relay-Version

This header line shows the version of the program responsible for the transmission of this article over the immediate link, that is, the program that is relaying the article from the next site. For example, suppose site A sends an article to site B, and site B forwards the article to site C. The message being transmitted from A to B would have a Relay-Version header identifying the program running on A, and the message transmitted from B to C would identify the program running on B. This header can be used to interpret older headers in an upward compatible way. Relay-Version must always be the first in a message; thus, all articles meeting this standard will begin with an upper case "R". No other restrictions are placed on the order of header lines.

The line contains two fields, separated by semicolons. The fields are the version and the full domain name of the site. The version should identify the system program used (e.g., "B") as well as a version number and version date. For example, the header line might contain

```
   Relay-Version: version B 2.10 2/13/83; site cbosgd.UUCP
```

This header should not be passed on to additional sites. A relay program, when passing an article on, should include only its own Relay-Version, not the Relay-Version of some other site. (For upward compatibility with older software, if a Relay-Version is found in a header which is not the first line, it should be assumed to be moved by an older version of news and deleted.)

2.1.2 Posting-Version

This header identifies the software responsible for entering this message into the network. It has the same format as Relay-Version. It will normally identify the same site as the

Message-ID, unless the posting site is serving as a gateway for a message that already contains a message ID generated by mail. (While it is permissible for a gateway to use an externally generated message ID, the message ID should be checked to ensure it conforms to this standard and to RFC 822.)

2.1.3 From

The From line contains the electronic mailing address of the person who sent the message, in the ARPA internet syntax. It may optionally also contain the full name of the person, in parentheses, after the electronic address. The electronic address is the same as the entity responsible for originating the article, unless the Sender header is present, in which case the From header might not be verified. Note that in all site and domain names, upper and lower case are considered the same, thus mark@cbosgd.UUCP, mark@cbosgd.uucp, and mark@CBosgD.UUcp are all equivalent. User names may or may not be case sensitive, for example, Billy@cbosgd.UUCP might be different from BillY@cbosgd.UUCP. Programs should avoid changing the case of electronic addresses when forwarding news or mail.

RFC 822 specifies that all text in parentheses is to be interpreted as a comment. It is common in ARPANET mail to place the full name of the user in a comment at the end of the From line. This standard specifies a more rigid syntax. The full name is not considered a comment, but an optional part of the header line. Either the full name is omitted, or it appears in parentheses after the electronic address of the person posting the article, or it appears before an electronic address enclosed in angle brackets. Thus, the three permissible forms are:

```
From: mark@cbosgd.UUCP
```

```
From: mark@cbosgd.UUCP (Mark Horton)
```

```
From: Mark Horton <mark@cbosgd.UUCP>
```

Full names may contain any printing ASCII characters from space through tilde, with the exceptions that they may not contain parentheses "(" or ")", or angle brackets "<" or ">". Additional restrictions may be placed on full names by the mail standard, in particular, the characters comma ",", colon ":", and semicolon ";" are inadvisable in full names.

2.1.4 Date

The Date line (formerly "Posted") is the date, in a format that must be acceptable both to the ARPANET and to the getdate routine, that the article was originally posted to the network. This date remains unchanged as the article is propagated throughout the network. One format that is acceptable to both is

```
Weekday, DD-Mon-YY HH:MM:SS TIMEZONE
```

Several examples of valid dates appear in the sample article above. Note in particular that ctime format:

```
Wdy Mon DD HH:MM:SS YYYY
```

is not acceptable because it is not a valid ARPANET date. However, since older software still generates this format, news implementations are encouraged to accept this format and translate it into an acceptable format.

The contents of the TIMEZONE field is currently subject to worldwide time zone abbreviations, including the usual American zones (PST, PDT, MST, MDT, CST, CDT, EST, EDT), the other North American zones (Bering through Newfoundland), European zones, Australian zones, and so on. Lacking a complete list at present (and unsure if an unambiguous list exists), authors of software are encouraged to keep this code flexible, and in particular not to assume that time zone names are exactly three letters long. Implementations are free to edit this field, keeping the time the same, but changing the time zone (with an appropriate adjustment to the local time shown) to a known time zone.

2.1.5 Newsgroups

The Newsgroups line specifies which newsgroup or newsgroups the article belongs in. Multiple newsgroups may be specified, separated by a comma. Newsgroups specified must all be the names of existing newsgroups, as no new newsgroups will be created by simply posting to them.

Wildcards (e.g., the word "all") are never allowed in a Newsgroups line. For example, a newsgroup "net.all" is illegal, although a newsgroup name "net.sport.football" is permitted.

If an article is received with a Newsgroups line listing some valid newsgroups and some invalid newsgroups, a site should not remove invalid newsgroups from the list. Instead, the invalid newsgroups should be ignored. For example, suppose site A subscribes to the classes "btl.all" and "net.all", and exchanges news articles with site B, which subscribes to "net.all" but not "btl.all". Suppose A receives an article with "Newsgroups: net.micro,btl.general". This article is passed on to B because B receives net.micro, but B does not receive btl.general. A must leave the Newsgroup line unchanged. If it were to remove "btl.general", the edited header could eventually reenter the "btl.all" class, resulting in an article that is not shown to users subscribing to "btl.general". Also, followups from outside "btl.all" would not be shown to such users. 2.1.6 Subject The Subject line (formerly "Title") tells what the article is about. It should be suggestive enough of the contents of the article to enable a reader to make a decision whether to read the article based on the subject alone. If the article is submitted in response to another article (e.g., is a "followup") the default subject should begin with the four characters "Re: " and the References line is required. (The user might wish to edit the subject of the followup, but the default should begin with "Re: ".)

2.1.7 Message-ID

The Message-ID line gives the article a unique identifier. The same message ID may not be reused during the lifetime of any article with the same message ID. (It is recommended that no message ID be reused for at least two years.) Message ID's have the syntax

```
"<" "string not containing blank or >" ">"
```

In order to conform to RFC 822, the Message-ID must have the format

```
"<" "unique" "@" "full domain name" ">"
```

where "full domain name" is the full name of the host at which the article entered the network, including a domain that host is in, and unique is any string of printing ASCII characters, not including "<", ">", or "@". For example, the "unique" part could be an integer representing a sequence number for articles submitted to the network, or a short string derived from the date and time the article was created. For example, valid message ID for an article submitted from site ucbvax in domain Berkeley.ARPA would be "<4123@ucbvax.Berkeley.ARPA>". Programmers are urged not to make assumptions about the content of message ID fields from other hosts, but to treat them as unknown character strings. It is not safe, for example, to assume that a message ID will be under 14 characters, nor that it is unique in the first 14 characters.

The angle brackets are considered part of the message ID. Thus, in references to the message ID, such as the ihave/sendme and cancel control messages, the angle brackets are included. White space characters (e.g., blank and tab) are not allowed in a message ID. All characters between the angle brackets must be printing ASCII characters.

2.1.8 Path

This line shows the path the article took to reach the current system. When a system forwards the message, it should add its own name to the list of systems in the Path line. The names may be separated by any punctuation character or characters, thus "cbosgd! mhuxj!mhuxt", "cbosgd, mhuxj, mhuxt", and "@cbosgd.uucp,@mhuxj.uucp,@mhuxt.uucp" and even "teklabs, zehntel, sri-unix@cca!decvax" are valid entries. (The latter path indicates a message that passed through decvax, cca, sri-unix, zehntel, and teklabs, in that order.) Additional names should be added from the left, for example, the most recently added name in the third example was "teklabs". Letters, digits, periods and hyphens are considered part of site names; other punctuation, including blanks, are considered separators.

Normally, the rightmost name will be the name of the originating system. However, it is also permissible to include an extra entry on the right, which is the name of the sender. This is for upward compatibility with older system.

The Path line is not used for replies, and should not be taken as a mailing address. It is intended to show the route the message travelled to reach the local site. There are several uses for this information. One is to monitor USENET routing for performance reasons. Another is to establish a path to reach new sites. Perhaps the most important is to cut down on redundant USENET traffic by failing to forward a message to a site that is known to have already received it. In particular, when site A sends an article to site B, the Path line includes "A", so that site B will not immediately send the article back to site A. The site name each site uses to identify itself should be the same as the name by which its neighbors know it, in order to make this optimization possible.

A site adds its own name to the front of a path when it receives a message from another site. Thus, if a message with path A!X!Y!Z is passed from site A to site B, B will add its own name to the path when it receives the message from A, e.g., B!A!X!Y!Z. If B then passes the message on to C, the message sent to C will contain the path B!A!X!Y!Z, and when C receives it, C will change it to C!B!A!X!Y!Z.

Special upward compatibility note: Since the From, Sender, and Reply-To lines are in internet format, and since many USENET sites do not yet have mailers capable of understanding internet format, it would break the reply capability to completely sever the connection between the Path header and the reply function. Thus, sites are required to continue to keep the Path line in a working reply format as much as possible, until January 1, 1984. It is recognized that the path is not always a valid reply string in older implementations, and no requirement to fix this problem is placed on implementations. However, the existing convention of placing the site name and an "!" at the front of the path, and of starting the path with the site name, an "!", and the user name, should be maintained at least until 1984.

2.2 Optional Headers

2.2.1 Reply-To

This line has the same format as From. If present, mailed replies to the author should be sent to the name given here. Otherwise, replies are mailed to the name on the From line. (This does not prevent additional copies from being sent to recipients named by the replier, or on To or Cc lines.) The full name may be optionally given, in parentheses, as in the From line.

2.2.2 Sender

This field is present only if the submitter manually enters a From line. It is intended to record the entity responsible for submitting the article to the network, and should be verified by the software at the submitting site.

For example, if John Smith is visiting CCA and wishes to post an article to the network, using friend Sarah Jones account, the message might read

```
From: smith@ucbvax.uucp (John Smith)

Sender: jones@cca.arpa (Sarah Jones)
```

If a gateway program enters a mail message into the network at site sri-unix, the lines might read

```
From: John.Doe@CMU-CS-A.ARPA

Sender: network@sri-unix.ARPA
```

The primary purpose of this field is to be able to track down articles to determine how they were entered into the network. The full name may be optionally given, in parentheses, as in the From line.

2.2.3 Followup-To

This line has the same format as Newsgroups. If present, follow-up articles are to be posted to the newsgroup(s) listed here. If this line is not present, followups are posted to the newsgroup(s) listed in the Newsgroups line, except that followups to "net.general" should instead go to "net.followup".

2.2.4 Date-Received

This line (formerly "Received") is in a legal USENET date format. It records the date and time that the article was first received on the local system. If this line is present in an article being transmitted from one host to another, the receiving host should ignore it and replace it with the current date. Since this field is intended for local use only, no site is required to support it. However, no site should pass this field on to another site unchanged. 2.2.5 Expires This line, if present, is in a legal USENET date format. It specifies a suggested expiration date for the article. If not present, the local default expiration date is used.

This field is intended to be used to clean up articles with a limited usefulness, or to keep important articles around for longer than usual. For example, a message announcing an upcoming seminar could have an expiration date the day after the seminar, since the message is not useful after the seminar is over. Since local sites have local policies for expiration of news (depending on available disk space, for instance), users are discouraged from providing expiration dates for articles unless there is a natural expiration date associated with the topic. System software should almost never provide a default Expires line. Leave it out and allow local policies to be used unless there is a good reason not to.

2.2.6 References

This field lists the message ID's of any articles prompting the submission of this article. It is required for all follow-up articles, and forbidden when a new subject is raised. Implementations should provide a follow-up command, which allows a user to post a follow-up article. This command should generate a Subject line which is the same as the original article, except that if the original subject does not begin with "Re: " or "re: ", the four characters "Re: " are inserted before the subject. If there is no References line on the original header, the References line should contain the message ID of the original article (including the angle brackets). If the original article does have a References line, the followup article should have a References line containing the text of the original References line, a blank, and the message ID of the original article.

The purpose of the References header is to allow articles to be grouped into conversations by the user interface program. This allows conversations within a newsgroup to be kept together, and potentially users might shut off entire conversations without unsubscribing to a newsgroup. User interfaces may not make use of this header, but all automatically generated followups should generate the References line for the benefit of systems that do use it, and manually generated followups (e.g. typed in well after the original article has been printed by the machine) should be encouraged to include them as well.

2.2.7 Control

If an article contains a Control line, the article is a control message. Control messages are used for communication among USENET host machines, not to be read by users. Control messages are distributed by the same newsgroup mechanism as ordinary messages. The body of the Control header line is the message to the host. For upward compatibility, messages that match the newsgroup pattern "all.all.ctl" should also be interpreted as control messages. If no Control: header is present on such messages, the subject is used as the control message. However, messages on newsgroups matching this pattern do not conform to this standard.

2.2.8 Distribution

This line is used to alter the distribution scope of the message. It has the same format as the Newsgroups line. User subscriptions are still controlled by Newsgroups, but the message is sent to all systems subscribing to the newsgroups on the Distribution line instead of the Newsgroups line. Thus, a car for sale in New Jersey might have headers including

```
Newsgroups: net.auto,net.wanted

Distribution: nj.all
```

so that it would only go to persons subscribing to net.auto or net.wanted within New Jersey. The intent of this header is to further restrict the distribution of a newsgroup, not to increase it. A local newsgroup, such as nj.crazy-eddie, will probably not be propagated by sites outside New Jersey that do not show such a newsgroup as valid. Wildcards in newsgroup names in the Distribution line are allowed. Followup articles should default to the same Distribution line as the original article, but the user can change it to a more limited one, or escalate the distribution if it was originally restricted and a more widely distributed reply is appropriate.

2.2.9 Organization

The text of this line is a short phrase describing the organization to which the sender belongs, or to which the machine belongs. The intent of this line is to help identify the person posting the message, since site names are often cryptic enough to make it hard to recognize the organization by the electronic address.

3. Control Messages

This section lists the control messages currently defined. The body of the Control header is the control message. Messages are a sequence of zero or more words, separated by white space (blanks or tabs). The first word is the name of the control message, remaining words are parameters to the message. The remainder of the header and the body of the message are also potential parameters; for example, the From line might suggest an address to which a response is to be mailed. Implementors and administrators may choose to allow control messages to be automatically carried out, or to queue them for manual processing. However, manually processed messages should be dealt with promptly.

3.1 Cancel

```
cancel <message ID>
```

If an article with the given message ID is present on the local system, the article is cancelled. This mechanism allows a user to cancel an article after the article has been distributed over the network.

Only the author of the article or the local super user is allowed to use this message. The verified sender of a message is the Sender line, or if no Sender line is present, the From line. The verified sender of the cancel message must be the same as either the Sender or From field of the original message. A verified sender in the cancel message is allowed to match an unverified From in the original message.

3.2 Ihave/Sendme

```
ihave <message ID list> <remotesys>

sendme <message ID list> <remotesys>
```

This message is part of the "ihave/sendme" protocol, which allows one site (say "A") to tell another site ("B") that a particular message has been received on A. Suppose that site A receives article "ucbvax.1234", and wishes to transmit the article to site B. A sends the control message "ihave ucbvax.1234 A" to site B (by posting it to newsgroup "to.B"). B responds with the control message "sendme ucbvax.1234 B" (on newsgroup to.A) if it has not already received the article. Upon receiving the Sendme message, A sends the article to B.

This protocol can be used to cut down on redundant traffic between sites. It is optional and should be used only if the particular situation makes it worthwhile. Frequently, the outcome is that, since most original messages are short, and since there is a high overhead to start sending a new message with UUCP, it costs as much to send the Ihave as it would cost to send the article itself.

One possible solution to this overhead problem is to batch requests. Several message ID's may be announced or requested in one message. If no message ID's are listed in the control message, the body of the message should be scanned for message ID's, one per line.

3.3 Newgroup

```
newgroup <groupname>
```

This control message creates a new newsgroup with the name given. Since no articles may be posted or forwarded until a newsgroup is created, this message is required before a newsgroup can be used. The body of the message is expected to be a short paragraph describing the intended use of the newsgroup.

3.4 Rmgroup

```
rmgroup <groupname>
```

This message removes a newsgroup with the given name. Since the newsgroup is removed from every site on the network, this command should be used carefully by a responsible administrator.

3.5 Sendsys

```
sendsys (no arguments)
```

The "sys" file, listing all neighbors and which newsgroups are sent to each neighbor, will be mailed to the author of the control message (Reply-to, if present, otherwise From). This information is considered public information, and it is a requirement of membership in USENET that this information be provided on request, either automatically in response to this control message, or manually, by mailing the requested information to the author of the message. This information is used to keep the map of USENET up to date, and to determine where netnews is sent.

The format of the file mailed back to the author should be the same as that of the "sys" file. This format has one line per neighboring site (plus one line for the local site), containing four colon separated fields. The first field has the site name of the neighbor, the second field has a newsgroup pattern describing the newsgroups sent to the neighbor. The third and fourth fields are not defined by this standard. A sample response:

```
From cbosgd!mark Sun Mar 27 20:39:37 1983
Subject: response to your sendsys request
To: mark@cbosgd.UUCP
Responding-System: cbosgd.UUCP
cbosgd:osg,cb,btl,bell,net,fa,to,test
ucbvax:net,fa,to.ucbvax:L:
cbosg:net,fa,bell,btl,cb,osg,to.cbosg:F:/usr/spool/outnews/cbosg
cbosgb:osg,to.cbosgb:F:/usr/spool/outnews/cbosgb
sescent:net,fa,bell,btl,cb,to.sescent:F:/usr/spool/outnews/sescent
npois:net,fa,bell,btl,ug,to.npois:F:/usr/spool/outnews/npois
mhuxi:net,fa,bell,btl,ug,to.mhuxi:F:/usr/spool/outnews/mhuxi
```

3.6 Senduuname

```
senduuname (no arguments)
```

The "uuname" program is run, and the output is mailed to the author of the control message (Reply-to, if present, otherwise From). This program lists all uucp neighbors of the local site. This information is used to make maps of the UUCP network. The sys file is not the same as the UUCP L.sys file. The L.sys file should never be transmitted to another party without the consent of the sites whose passwords are listed therein.

It is optional for a site to provide this information. Some reply should be made to the author of the control message, so that a transmission error won't be blamed. It is also permissible for a site to run the uuname program (or in some other way determine the uucp neighbors) and edit the output, either automatically or manually, before mailing the reply back to the author. The file should contain one site per line, beginning with the uucp site name. Additional information may be included, separated from the site name by a blank or tab. The phone number or password for the site should NOT be included, as the reply is considered to be in the public domain. (The uuname program will send only the site name and not the entire contents of the L.sys file, thus, phone numbers and passwords are not transmitted.)

The purpose of this message is to generate and maintain UUCP mail routing maps. Thus, connections over which mail can be sent using the site!user syntax should be included, regardless of whether the link is actually a UUCP link at the physical level. If a mail router should use it, it should be included. Since all information sent in response to this message is optional, sites are free to edit the list, deleting secret or private links they do not wish to publicise.

3.7 Version

```
version (no arguments)
```

The name and version of the software running on the local system is to be mailed back to the author of the article (Reply-to if present, otherwise From).

4. Transmission Methods

USENET is not a physical network, but rather a logical network resting on top of several existing physical networks. These networks include, but are not limited to, UUCP, the ARPANET, an Ethernet, the BLICN network, an NSC Hyperchannel, and a Berknet. What is important is that two neighboring systems on USENET have some method to get a new article, in the format listed here, from one system to the other, and once on the receiving system, processed by the netnews software on that system. (On UNIX systems, this usually means the "rnews" program being run with the article on the standard input.)

It is not a requirement that USENET sites have mail systems capable of understanding the ARPA Internet mail syntax, but it is strongly recommended. Since From, Reply-To, and Sender lines use the Internet syntax, replies will be difficult or impossible without an internet mailer. A site without an internet mailer can attempt to use the Path header line for replies, but this field is not guaranteed to be a working path for replies. In any event, any site generating or forwarding news messages must have an internet address that allows them to receive mail from sites with internet mailers, and they must include their internet address on their From line.

4.1 Remote Execution

Some networks permit direct remote command execution. On these networks, news may be forwarded by spooling the rnews command with the article on the standard input. For example, if the remote system is called "remote", news would be sent over a UUCP link with the command "uux - remote!rnews", and on a Berknet, "net -mremote rnews". It is important that the article be sent via a reliable mechansim, normally involving the possibility of spooling, rather than direct real-time remote execution. This is because, if the remote system is down, a direct execution command will fail, and the article will never be delivered. If the article is spooled, it will eventually be delivered when both systems are up.

4.2 Transfer by Mail

On some systems, direct remote spooled execution is not possible. However, most systems support electronic mail, and a news article can be sent as mail. One approach is to send a mail message which is identical to the news message: the mail headers are the news headers, and the mail body is the news body. By convention, this mail is sent to the user "newsmail" on the remote machine.

One problem with this method is that it may not be possible to convince the mail system that the From line of the message is valid, since the mail message was generated by a program on a system different from the source of the news article. Another problem is that error messages caused by the mail transmission would be sent to the originator of the news article, who has no control over news transmission between two cooperating hosts and does not know who to contact. Transmission error messages should be directed to a responsible contact person on the sending machine.

A solution to this problem is to encapsulate the news article into a mail message, such that the entire article (headers and body) are part of the body of the mail message. The convention here is that such mail is sent to user "rnews" on the remote system. A mail message body is generated by prepending the letter "N" to each line of the news article, and then attaching whatever mail headers are convenient to generate. The N's are attached to prevent any special lines in the news article from interfering with mail transmission, and to prevent any extra lines inserted by the mailer (headers, blank lines, etc.) from becoming part of the news article. A program on the receiving machine receives mail to "rnews", extracting the article itself and invoking the "rnews" program. An example in this format might look like this:

```
Date: Monday, 3-Jan-83 08:33:47 MST
From: news@cbosgd.UUCP
Subject: network news article
To: rnews@npois.UUCP
NRelay-Version: B 2.10 2/13/83 cbosgd.UUCP
NPosting-Version: B 2.9 6/21/82 sask.UUCP
NPath: cbosgd!mhuxj!harpo!utah-cs!sask!derek
NFrom: derek@sask.UUCP (Derek Andrew)
NNewsgroups: net.test
NSubject: necessary test
NMessage-ID: <176@sask.UUCP>
NDate: Monday, 3-Jan-83 00:59:15 MST
N
NThis really is a test. If anyone out there more than 6
Nhops away would kindly confirm this note I would
Nappreciate it. We suspect that our news postings
Nare not getting out into the world.
N
```

Using mail solves the spooling problem, since mail must always be spooled if the destination host is down. However, it adds more overhead to the transmission process (to encapsulate and extract the article) and makes it harder for software to give different priorities to news and mail.

4.3 Batching

Since news articles are usually short, and since a large number of messages are often sent between two sites in a day, it may make sense to batch news articles. Several articles can be combined into one large article, using conventions agreed upon in advance by the two sites. One such batching scheme is described here; its use is still considered experimental.

News articles are combined into a script, separated by a header of the form:

```
##! rnews 1234
```

where 1234 is the length, in bytes, of the article. Each such line is followed by an article containing the given number of bytes. (The newline at the end of each line of the article is counted as one byte, for purposes of this count, even if it is stored as CRLF.) For example, a batch of articles might look like this:

```
#! rnews 374
   Relay-Version: version B 2.10 2/13/83; site cbosgd.UUCP
   Posting-Version: version B 2.10 2/13/83; site eagle.UUCP
   Path: cbosgd!mhuxj!mhuxt!eagle!jerry
   From: jerry@eagle.uucp (Jerry Schwarz)
   Newsgroups: net.general
   Subject: Usenet Etiquette -- Please Read
   Message-ID: <642@eagle.UUCP>
   Date: Friday, 19-Nov-82 16:14:55 EST
   Here is an important message about USENET Etiquette.
   #! rnews 378
   Relay-Version: version B 2.10 2/13/83; site cbosgd.UUCP
   Posting-Version: version B 2.10 2/13/83; site eagle.UUCP
   Path: cbosgd!mhuxj!mhuxt!eagle!jerry
   From: jerry@eagle.uucp (Jerry Schwarz)
   Newsgroups: net.followup
   Subject: Notes on Etiquette article
   Message-ID: <643@eagle.UUCP>
   Date: Friday, 19-Nov-82 17:24:12 EST
   There was something I forgot to mention in the last message.
```

Batched news is recognized because the first character in the message is "#". The message is then passed to the unbatcher for interpretation.

5. The News Propagation Algorithm

This section describes the overall scheme of USENET and the algorithm followed by sites in propagating news to the entire network. Since all sites are affected by incorrectly formatted articles and by propagation errors, it is important for the method to be standardized.

USENET is a directed graph. Each node in the graph is a host computer, each arc in the graph is a transmission path from one host to another host. Each arc is labelled with a newsgroup pattern, specifying which newsgroup classes are forwarded along that link. Most arcs are bidirectional, that is, if site A sends a class of newsgroups to site B, then site B usually sends the same class of newsgroups to site A. This bidirectionality is not, however, required.

USENET is made up of many subnetworks. Each subnet has a name, such as "net" or "btl". The special subnet "net" is defined to be USENET, although the union of all subnets may be a superset of USENET (because of sites that get local newsgroup classes but do not get net.all). Each subnet is a connected graph, that is, a path exists from every node to every other node in the subnet. In addition, the entire graph is (theoretically) connected. (In practice, some political considerations have caused some sites to be unable to post articles reaching the rest of the network.)

An article is posted on one machine to a list of newsgroups. That machine accepts it locally, then forwards it to all its neighbors that are interested in at least one of the newsgroups of the message. (Site A deems site B to be "interested" in a newsgroup if the newsgroup matches the pattern on the arc from A to B. This pattern is stored in a file on the A machine.) The sites receiving the incoming article examine it to make sure they really want the article, accept it locally, and then in turn forward the article to all their interest neighbors. This process continues until the entire network has seen the article.

An important part of the algorithm is the prevention of loops. The above process would cause a message to loop along a cycle forever. In particular, when site A sends an article to site B, site B will send it back to site A, which will send it to site B, and so on. One solution to this is the history mechanism. Each site keeps track of all articles it has seen (by their message ID) and whenever an article comes in that it has already seen, the incoming article is discarded immediately. This solution is sufficient to prevent loops, but additional optimizations can be made to avoid sending articles to sites that will simply throw them away.

One optimization is that an article should never be sent to a machine listed in the Path line of the header. When a machine name is in the Path line, the message is known to have passed through the machine. Another optimization is that, if the article originated on site A, then site A has already seen the article. (Origination can be determined by the Posting-Version line.)

Thus, if an article is posted to newsgroup "net.misc", it will match the pattern "net.all" (where "all" is a metasymbol that matches any string), and will be forwarded to all sites that subscribe to net.all (as determined by what their neighbors send them). These sites make up the "net" subnetwork. An article posted to "btl.general" will reach all sites receiving "btl.all", but will not reach sites that do not get "btl.all". In effect, the articles reaches the "btl" subnetwork. An article posted to newsgroups "net.micro,btl.general" will reach all sites subscribing to either of the two classes.

Getting Custom Controls Written

If you or your organization would like to have custom controls written, you can contact us at the following:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA   98103-1926

Phone: 206-634-1443
Fax: 206-632-0272

CompuServe: 71231,2066
Internet: mabry@mabry.com

You can also contact Zane Thomas.   He can be reached at:

Zane Thomas
Post Office Box 121
Indianola, WA   98342

Internet: zane@mabry.com

Licensing Information

Legalese Version

Mabry Software grants a license to use the enclosed software to the original purchaser. Copies may be made for back-up purposes only. Copies made for any other purpose are expressly prohibited, and adherence to this requirement is the sole responsibility of the purchaser.

Customer written executable applications containing embedded Mabry products may be freely distributed, without royalty payments to Mabry Software, provided that such distributed Mabry product is bound into these applications in such a way so as to prohibit separate use in design mode, and that such Mabry product is distributed only in conjunction with the customers own software product. The Mabry Software product may not be distributed by itself in any form.

Neither source code for Mabry Software products nor modified source code for Mabry Software products may be distributed under any circumstances, nor may you distribute .OBJ, .LIB, etc. files that contain our routines. This control may be used as a constituent control only if the compound control thus created is distributed with and as an integral part of an application. Permission to use this control as a constituent control does not grant a right to distribute the license (LIC) file or any other file other than the control executable itself.This license may be transferred to a third party only if all existing copies of the software and its documentation are also transferred.

This product is licensed for use by only one developer at a time. Mabry Software expressly prohibits installing this product on more than one computer if there is any chance that both copies will be used simultaneously. This restriction also extends to installation on a network server, if more than one workstation will be accessing the product. All developers working on a project which includes a Mabry Software product, even though not working directly with the Mabry product, are required to purchase a license for that Mabry product.

This software is provided as is. Mabry Software makes no warranty, expressed or implied, with regard to the software. All implied warranties, including the warranties of merchantability and fitness for a particular use, are hereby excluded.

MABRY SOFTWARE'S LIABILITY IS LIMITED TO THE PURCHASE PRICE. Under no circumstances shall Mabry Software or the authors of this product be liable for any incidental or consequential damages, nor for any damages in excess of the original purchase price.

To be eligible for free technical support by telephone, the Internet, CompuServe, etc. and to ensure that you are notified of any future updates, please complete the enclosed registration card and return it to Mabry Software.

English Version

We require that you purchase one copy of a control per developer on a project. If this is met, you may distribute the control with your application royalty free. You may never distribute the LIC file. You may not change the product in any way that removes or changes the requirement of a license file.

We encourage the use of our controls as constituent controls when the compound controls you create are an integral part of your application. But we don't allow distribution of our controls as constituents of other controls when the compound control is not part of an application. The reason we need to have this restriction is that without it someone might decide to use our control as a constituent, add some trivial (or even non-trivial) enhancements and then sell the compound control. Obviously there would be little difference between that and just plain reselling our control.

If you have purchased the source code, you may not re-distribute the source code either (nor may you copy it into your own project). Mabry Software retains the copyright to the