Gopher+

proposed enhancements to the internet Gopher protocol

Bob Alberti, Farhad Anklesaria, Paul Lindner, Mark P. McCahill, Daniel Torrey
University of Minnesota Microcomputer and Workstation Networks Center
Summer 1992

gopher+ n. 1. Hardier strains of mammals of the family Geomyidae. 2. (Amer. colloq.) Native or inhabitant of Minnesota, the Gopher state, in full winter regalia (see PARKA). 3. (Amer. colloq.) Executive secretary. 4. (computer tech.) Software following a simple protocol for burrowing through a TCP/IP internet, made more powerful by simple enhancements (see CREEPING FEATURISM).

Abstract

The internet Gopher protocol was designed for distributed document search and retrieval. The document **The internet Gopher protocol: a distributed document search and retrieval protocol** describes the basic protocol and has an overview of how to implement new client and server applications. This document proposes some enhancements to the syntax, semantics and functionality of the original Gopher protocol.

Distribution of this document is unlimited. Please send comments to the Gopher development team: <gopher@boombox.micro.umn.edu>. Experimentation with the mechanisms described here is encouraged.

1. Introduction

The Internet Gopher protocol was designed primarily to act as a distributed document delivery system. It has enjoyed increasing popularity, and is being used for purposes that were not visualized when the protocol was first outlined. The rest of this document describes the Gopher+ enhancements in a non-rigorous but easily read and understood way. There is a short BNF-like section at the end for exact syntax descriptions. Throughout the document, "#" stands for the ASCII TAB character. There is an implicit carriage return and line-feed at the ends of lines; these will only be explicitly mentioned where necessary to avoid confusion. To understand this document, you really must be familiar with the basic Gopher protocol.

Servers and clients understanding the Gopher+ extensions, transmit extra information at the ends of list and request lines. Old, basic gopher clients ignore such information. New Gopher+ aware servers continue to work at their old level with unenhanced clients. The extra information that can be communicated by Gopher+ clients may be used to summon new capabilities to bridge the most keenly felt shortcomings of the venerable old Gopher.

2. How does Gopher+ work?

Gopher+ enhancements rely on transmitting "extra" tab delimited fields beyond what regular (old) Gopher servers and clients now use. If most existing (old) clients were to encounter extra stuff beyond the "port" field in a list (directory), most would ignore it. Gopher+ servers will return item descriptions in this form:

```
1Display string#selector string#host#port#extra stuff<CRLF>
```

If an existing (old) client has problems with additional information beyond the port, it should not take much more than a simple tweak to have it discard unneeded stuff.

2.1 Advisory issued to client maintainers.

If it does not do this already, your existing client should be modified as soon as possible to ignore extra fields beyond what it expects to find. This will ensure that clients not break when Gopher+ servers enter gopherspace.

All the regular Gopher protocol info remains intact except for:

(1) Instead of just a CRLF after the port field in any item of a list (directory) there may be an optional TAB followed by extra stuff as noted above (explanation to follow).

(2) In the original Gopher protocol, there was provision for a date-time descriptor (sec 3.6) to be sent after the selector (for use by autoindexer beasts). As far as we know, while the descriptor is implemented in the Mac server, it is not in any other server and no clients or daemons use it. This is a good time to withdraw this feature. The basic gopher protocol has been revised for the final time and will be frozen.

2.2 Gopher+ item lists.

Gopher servers that can utilize the Gopher+ enhancements will send some additional stuff (frequently the character "+") after the port field describing any list item. eg:

```
1Some old directory#foo selector#host1#port1
1Some new directory#bar selector#host1#port1#+
0Some file or other#moo selector#host2#port2#+
```

The first line is the regular old gopher item description. The second line is new Gopher+ item description. The third line is a Gopher+ description of a document. Old gopher clients can request the latter two items using old format gopher selector strings and retrieve the items. New, Gopher+ savvy clients will notice the trailing + and know that they can do extra things with these kinds of items.

2.3 Gopher+ data transfer.

If a client sends out a Gopher+ type request to a server (by tagging on a tab and a "+" to the request):

```
bar selector#+
```

The server may return the response in one of three ways; examples below:

```
+5340<CRLF><data>

+-1<CRLF><data><CRLF>.<CRLF>

+-2<CRLF><data>
```

The first response means: I am going to send exactly 5340 bytes at you and they will begin right after this line. The second response means: I have no idea how many bytes I have to send (or I am lazy), but I will send a period on a line by itself when I am done. The third means: I really have no idea how many bytes I have to send, and what's more, they COULD contain the <CRLF>.<CRLF> pattern, so just read until I close the connection.

The first character of a response to a Gopher+ query denotes success (+) or failure (-). Following that is a token to be interpreted as a decimal number. If the number is >= 0, it describes the length of the dataBlock. If = -1, it means the data is period terminated. If = -2, it means the data ends when the connection closes.

The server may return an error also, as in:

```
--1<CRLF><data><CRLF>.<CRLF>
```

The (short!) error message will be in ASCII text in the data part. The first token on the first line of the error text (data) contains an error-code (an integer). It is recommended that the first line also contain the email address of the administrator of the server (in angle brackets). Both the error-code and the email address may easily be extracted by the client. Subsequent lines contain a short error message that may be displayed to the user. Basic error codes are:

1 Item is not available.

2	Try again later ("eg. My load is too high right now.")
3	Item has moved. Following the error-code is the gopher descriptor of where it now lives.

More error codes may be defined as the need arises.

This should be obvious: if the client sends out an "old" Gopher kind of request:

```
bar selector
```

the server will know that it is talking to an old client and will respond in the old way. This means that old gopher clients can still access information on Gopher+ servers.

2.4 Gopher+ client requests.

Clients can send requests to retrieve the contents of an item in this form:

```
selector string#+[representation][#dataFlag]<CRLF>[dataBlock]
```

If dataFlag is '0', or nonexistent, then the client will not send any data besides the selector string. If the dataFlag is '1' then a block of data will follow in the same format as Section 2.3. The client can send a large amount of data to the server in the dataBlock. Representations or alternative views of an item's contents may be discovered by interrogating the server about the item's attribute information; this is explained below.

2.5 Gopher+ Item Attribute Information.

The most basic enhancement of Gopher+ items is the ability to associate information about an item such as size, alternative views, the administrator, an abstract, etc. with the item. To get Attribute Information, a client can send out a request to the gopher server that looks like this:

```
selector string#!<CRLF>
```

(think of "!" as an upside-down i for "information"). To the server this means "Instead of returning the contents of the item, return the item's Attribute Information". The Attribute Information is returned as an ASCII text stream containing blocks of information.For example, a server might return:

```
+INFO: 0Some file or other#moo selector#host2#port2#+
+ADMIN:
 Admin: Frodo Gophermeister <fng@bogus.edu>
 Mod-Date:  15 August 1992 <19920815185503>
+VIEWS:
 Text: <10k>
 Postscript: <100k>
 Text German: <15k>
MacWriteII: <45K>
+ABSTRACT:
 This is a short (but multi-line) abstract about the
 item. Two or three lines ought to be enough
```

The beginning of a block of information is denoted by a "+" in column 1 of a line. Another way to think of it is: the name of each block begins with a + and the rest of the name cannot contain a +. Each line of information within a block begins with a space so that it is easy to locate the beginning of a block.

There can be multiple blocks of information about an item, but the first block *must* be the one-line +INFO block containing the keyword +INFO followed by the gopher item description. This is done to make it easy to associate informational attributes with the gopher items to which they refer (see section 2.7 for some good reasons for doing this). The very first line of Attribute Information for an item contains a one-line +INFO block containing the gopher descriptor for the item. All Gopher+ servers must return an "+INFO" block for all items listed by the server. Also present may be an +ADMIN block that can be many lines long. The server must also send an +ADMIN block when asked to send all the item's attributes (as in the example above). The +ADMIN block must contain at least an Admin attribute and Mod-Date attributes, though there may be many other administrative items also present in the +ADMIN block. The Admin (the administrator of the item) and Date (the date of the item's last modification) attributes are required to be returned by the server, and other optional attributes may be returned as well. In this example, there are two pieces of information within the +ADMIN block (Admin and Mod-Date). The Admin attribute must contain the e-mail address of an administrator inside angle brackets. The Admin line might also contain the administrators name and phone number. The Date line must contain the modification date in angle brackets. The format of the date is <YYYYMMDDhhmmss> where YYYY is year, MM is month, DD is day, hh is hours, mm is minutes, and ss is seconds.

The third block in the example is the +VIEWS block. This block lists different formats in which the document can be retrieved. The first format listed is what the server believes to be the preferred format. A gopher client might display the list of possible view labels of the item to the user and let the user select the view they prefer. Alternatively, a smart client might look at the content of the labels and preferentially retrieve Postscript views of items. Note that the view labels are structured. View labels specify a handler (Postscript, Text, etc.), an optional language (English, German, etc.) and an optional size. The client software can pick off the size of each view IF there are any angle brackets on the line. There might not be a size that the server cares to tell you about. Also this might NOT be the exact size that the server will wind up delivering to you if you ask for it... but it should be reasonably close. This information makes it possible for clever clients to select views based on size, data representation, or language. See section 2.6 for how alternate representations (views) are retrieved.

The next block in the example is an (optional) +ABSTRACT. Here the block consists of lines of text that might be displayed to the user.

Other blocks of information can defined and added as the need arises. For instance, a Neuromancer-esque 3-D cyberspace attribute might be accommodated by including a

3D-ICON block (with an image to display in 3-space) and a 3D-COORDINATE block (with y,x, and z coordinates). More immediate needs can also addressed by defining other information blocks. For instance, a SCRIPT block would be a natural place to put information for scripting telnet sessions. Information blocks give us an extensible way of adding attributes (or what Macintosh programmers call resources) to gopher items.

Some of the really cool ideas we have for information attributes may require sending large amounts of data, some of which may not be easily represented as ASCII text, but the idea of the attributes information is that it is a relatively compact list of attributes. These somewhat conflicting desires can be reconciled by allowing references to gopher items in an attribute. For example, an +ABSTRACT block might be returned this way:

```
 +ABSTRACT: 0long abstract#selector#host2#port2#+
```

In this example, the abstract is a document that resides on a gopher server. By allowing references to to gopher items, we can also accommodate data that must be sent in an 8-bit clear stream by using the Gopher+ methods for retrieving binary data.

If both a reference to an attribute and an explicit value for the attribute are present in an attribute list, the preferred version is the explicit value. In the example below, the preferred version is "the short abstract goes here".

```
 +ABSTRACT: 0long abstract#selector#host2#port2#+
the short abstract goes here
```

Note that if you want to have several views of (for example) an +ABSTRACT this is possible by using a reference to a item residing on a gopher server because the item can have its own attributes.

Attributes names are case sensitive (easier to match and more of them). There is no need to "preregister" all possible attributes since we cannot anticipate all possible future needs. However it would be reasonable to maintain a registry for implementors and administrators so duplication can be avoided. Server implementors or administrators can request that new attributes be included in the registry.

Dream on: What gets us excited are alternate representations for directory lists. Sure, the standard representation for a gopher directory list is known to us all. But isn't hypertext (in a WWW sense) an alternate kind of directory list? We also envisioned a "geographical view" (GView?) mapping servers onto a map of the world (throw up a gif picture and then overlay dots based on latitude and longitude or xy coordinates). OK. Dream off.

Note that interested parties outside gopherspace have long and complex wish-lists for "attributes" that all well-dressed Internet citizens should have. We don't want to comment on the use or value of these laundry-lists. Suffice it to say that nothing precludes server administrators from including whatever attributes they see fit to include. Certainly IAFA blocks are desirable, bearing UDIs, URL's or whatever else is desired. The gopher community will probably arrive at a list of "recommended" attributes that server administrators should try to support. Because not every server administrator sees advantage to cluttering Attribute Info files with information their primary users will never need, it does not seem fair to "force" folks to include them; most will just ignore the harsh protocol guideline and the value of the protocol will be diminished. We want to mandate as little as we possibly can.

2.6 Using Attribute Info: Alternate representations (+VIEWS).

The user may locate a document and wonder if there are representations of it besides, say, the standard Text. Using the appropriate client incantation (Option Double-Click? or whatever) the user indicates a wish to see what's available. The client retrieves the Attribute Information, displays the list of views to the user in some kind of scrolling list dialog. User selects a line and client now requests the document in say, Postscript representation:

```
the selector#+Postscript
```

Smart clients are not precluded from doing things like "Always get Postscript if you can" or "Always get Postscript if that is less than 700K in size." etc. And the "smarter" you make it, the hairier your client will become - unless you are a user interface wizard of awesome proportions. While the example above is of fetching a document's postscript view, there is nothing precluding having different views for directories. In the dream sequence earlier, we imagined a geographic view of a directory. For a client to fetch that view, it would say this:

```
the selector#+GView
```

2.7 Getting attributes for all items in a directory in one transaction.

Heavyweight/clever/special-purpose clients may want to know all the attributes of items in a given directory in one transaction. The "$" command is used to request all the attributes of a directory at once. For instance, a client might sent the request:

```
selector string#$
```

and the server might return this:

```
 +INFO: 0Salmon dogs#some selector#host2#port2#+
 +ADMIN:
  Admin: Frodo Gophermeister <fng@bogus.edu>
  Mod-Date: August 15, 1992 <19920815185503>
 +VIEWS:
  Text: <10k>
  Postscript German: <100k>
 +ABSTRACT:
A great recipe for making salmon
 +INFO: 0Hot pups#other selector#host3#port3#+
 +ADMIN:
  Admin: Bilbo Gophernovice <bng@bogus.edu>
  Date: <19910101080003>
```

In this example, the server returned the attribute lists for two items because there were only two items in the directory.. The client software can easily separate the attributes for the items since each attribute list starts with "+INFO". It is also easy for the client to use the "$" command to get directory listings since the gopher item descriptor is on the +INFO line for each item.

Note that the $ command is the only way to find the administrator of a remote link. To get the full attribute information for a link on another machine may require asking the master machine for the item information. It is possible to append which attributes you are interested in retrieving after the $, eg:

some directory selector`#$+VIEWS`

`or`

other directory selector`#$+VIEWS+ABSTRACT`

The $ command makes it possible for a client that does not mind burning bandwidth to get attribute information for all items as the user navigates gopherspace. Clients using 2400 bps SLIP links will probably not use this method... but clients on Ethernet may not mind. This command may also be useful for building indexes of gopherspace if one wanted to do an Archie-like service for gopherspace. Note that the specific requested attributes are only suggestions to the server that the client would like less than a full set of attributes. The server may choose to ignore the request (if it is not capable of extracting the required attributes) and return the client the full set anyway. Other caveats: even if the attributes requested are not available, the server WILL NOT return an error, but will send whatever IS available. It is the client's responsibility inspect the returned attributes.

Analogous to use of the $ command, the ! command can also be used to request certain attribute blocks.

2.8 Gopher+ Interactive Query items.

The principle here is based on Roland Schemer's "Q/q" type ideas. We're calling it the

Interactive Query enhancements...

The server may list items that have a "?" following the port field:

```
0A file#file selector#host#port#?
1A directory#dir selector#host#port#?
```

Now the fact that there's something there means that they're Gopher+ items. Old clients will still be able to show such items in lists, but if they simply send the old style plain selector string to retrieve them, the server will respond with an old style error telling them to get an updated client. New clients will know that before getting one of these items, it will be necessary to retrieve questions from the server, have the user answer them, and then feed the answers back to the server along with the selector. The questions to be asked of the user are retrieved from the server by looking at the +ASK attribute in the item's attribute information.

So when the user selects a query item, the client quickly connects to the server and requests the Attribute Information for the item. Then the client extracts the information in the +ASK attribute block. Here's an example:

```
  +INFO: 0inquisitive#moo moo selector#host2#port2#+
  +ADMIN
   Admin: Frank Gophermeister <fng@bogus.edu>
   Mod-Date: August 15, 1992 <19920815185503>
+ASK:
 Ask: How many volts?
 Choose: Deliver electric shock to administrator now?#Yes#Not!
```

The client will use all lines in the order they appear in the +ASK attribute block. The content will be presented to the user as questions or prompts or dialogs or something like that.

The "Ask" presents the user with a question, supplies a default text answer if it exists.

The "AskF" requests the user for a new local filename, presumably for stashing the response returned by the server. It may supply a default filename.

The "Choose" presents the user with a few short choices; ideal for a dialog box with a couple or three buttons.

The "ChooseF" requests that the user select an existing local file, presumably for sending to the server. On some systems, the client writer or administrator might want to restrict the selection of such files to the current directory (ie. not allow paths in the filename to prevent sending things like password files).

The n responses harvested from the user are sent on to the server as the first n lines in the dataBlock. There can only be one file sent, and it will be the remainder of the dataBlock if any.

Because of the potential for abuse, client writers should provide a way to turn off +ASK block interpretation, and distribute clients that do not interpret +ASK blocks by default. It is recommended that the +ASK mechanism not be used as a password request, and users be cautioned against typing their password to anything but a very familiar looking login request.

Gopher was originally designed as an essentially anonymous document retrieval protocol to facilitate easy access to information rather than limited access. Various kinds of restrictive mechanisms have been implemented at the server end (for example, access restriction by source IP address); however if you have sensitive information, we emphasize that putting it under a Gopher's nose is not a good idea.

The folks with a hirsute tendency will have noticed that all these interactions are static rather than truly dynamic and interactive. In other words, the server cannot ask different questions in response to different answers. +ASK does not constitute a scripting language by any means.

To do "true" scripting, we have to do one of two things

1. Write a full language parser/interpreter into clients. The server loads a whole script into the client's brain, and the client "runs" it. This rather grossly violates the spirit of simplicity in cross-platform gopher implementation. However, when and if a standard scripting language is adopted, there will be room for it in a SCRIPT attribute block.

2. Client enters a complex back-and-forth transaction with the server. This requires the server, client, or both to save rather a lot of state. NOPE! Server saving state means holding open a connection or (worse) the server retaining tokens between connections. Client saving state means the server has an even worse job to do.

As Opus the Penguin would say: a Hairball.

2.9 Gopher+ Pictures, Sounds, Movies.

A lot of folks need ability to retrieve and display pictures, but there is no real consensus on ONE format for these pictures. We don't want to define a type character for every oddball picture type. Gopher+ handles Pictures, Movies, and Sounds by defining three item types: ":" for bitmap images, ";" for movies, and "<" for sounds (originally I, M, and S were suggested, but they were informally in use in other ways; the only thing magic about ":", ";", and "<", is that they are the first characters after '9') **Note that there is NO default format for Pictures, Movies and Sounds; the specific format of the image, movie, or sound must be gleaned from the +VIEWS information for the item (eg. Gif, PICT, TIFF, etc.).**

Appendix I

Required attributes and suggested attributes.

A1.0 The +INFO attribute block

The +INFO atttribute block is sent whenever an item's attributes are requested. It is required that the Attribute Information list for an item must contain a one-line +INFO attribute, and the +INFO attribute must contain the gopher+ descriptor for the item.

```
+INFO: 1Nice stuff#/selector#host#port#+
```

A2.0 The +ADMIN attribute

A Gopher+ server is required to have an +ADMIN block for every item and the +ADMIN block must contain Admin and a Mod-Date lines:

```
+ADMIN:
 Admin: [comments] <administrator e-mail address>
 Mod-Date: [comments] <YYYYMMDDhhmmss>
```

In addition to the required lines, we recommend that the +ADMIN attribute of items returned by a full-text search engine contain a SCORE attribute. The SCORE attribute should contain the relevance ranking (an integer) of the item.

```
 Score: relevance-ranking
```

We recommend that the +ADMIN attribute of a full-text search engine contain a Score-Range attribute. This attribute is used to specify the range of values taken on by the relevance ranking scores of items returned by the search engine. The Score-Range makes it possible to normalize scores from different search engine technologies. The first number is the lower bound, the second number is the upper bound.

```
 Score-range: lower-bound upper-bound
```

We also recommend that the +ADMIN attribute for the root of the server (i.e. what you get back when you ask for the attributes of the item with the empty selector string) also contain these fields:

```
Site: the name of the site
Org: organization or group owning the site
Loc: city, state, country
Geog: latitude longitude
TZ: timezone as gmt-offset
```

Other useful attributes might include:

```
Provider: who provided this item
  Author: who wrote this item
  Creation-Date: when it was born <YYYYMMDDhhmmss>
  Expiration-Date: when it expires <YYYYMMDDhhmmss>
```

A3.0 The +VIEWS attribute

The +VIEWS attribute is used to specify alternative representations of an item. The form of the +VIEWS attribute is:

```
+VIEWS: [gopher descriptor]
 viewHandler[ viewLanguage]: [<56K>]
 viewHandler[ viewLanguage]: [<93K>]
 viewHandler[ viewLanguage]: [<77K>]
```

Some values for viewhandler are

```
Text, List, Postscript, Gif, Pict, Pict2, Tiff.
```

The Gopher Registry document will list all currently registered views for implementors and administrators. All gopher servers must support the Text view for readable documents and the List view (the basic Gopher+ directory list) for directories. These are the views that must be returned by default. If all a server supports is the default views, then it may omit the +VIEWS attribute block (although we suggest that it not do so).

Some registered values for viewLanguage are

```
English, German, French, Spanish, Swedish, Finnish, Norwegian, Dutch
```

A4.0 The +ABSTRACT attribute

The +ABSTRACT attribute is used to specify a short abstract for the item. The form of the +ABSTRACT attribute is:

```
+ABSTRACT: [gopher reference]
 A line of text<CRLF>
 another line of text<CRLF>
 still another line of text.<CRLF>
```

We recommend that a description of the sorts of information at the site, a postal address, a phone number, and the administrator name for the site be included in the +ABSTRACT attribute for the server root (i.e. what you get when you ask for the attribute list of the server with no selector string).

Appendix II

Paul's NQBNF (Not Quite BNF) for the Gopher+ Enhancements.

Note: This is modified BNF (as used by the Pascal people) with a few English modifiers thrown in. Stuff enclosed in '{}' can be repeated zero or more times. Stuff in '[]' denotes a set of items. The '-' operator denotes set subtraction.

This section is not quite solid yet. Please send us information on any errors you might notice.

Directory Entity

```
CR-LF     ::= Carriage Return Character followed by Line Feed character.
Tab       ::= ASCII Tab character
NUL       ::= ASCII NUL character
PLUS      ::= ASCII '+' character
LEFT      ::= ASCII '<' character
RIGHT      ::= ASCII '>' character
OCTET     ::= $00 -> $ff
UNASCII   ::= OCTET - [Tab CR-LF NUL]
UNASCIINOPLUS ::= UNASCII - [PLUS]
UNASCIINOANGLE ::= UNASCII - [LEFT, RIGHT]
Lastline  ::= '.'CR-LF
TextBlock ::= Block of ASCII text not containing Lastline pattern.
Type      ::= UNASCII
DisplayString ::= {UNASCII}
Selector  ::= {UNASCII}
Otherflds ::= {UNASCII + TAB}
Host      ::= {{UNASCII - ['.']} '.'} {UNASCII - ['.']}
```

Note: This is a Fully Qualified Domain Name as defined in RFC 830. (e.g. gopher.micro.umn.edu) Hosts that have a CR-LF TAB or NUL in their name get what they deserve.

```
Digit     ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
DigitSeq  ::= digit {digit}.
Port      ::= DigitSeq.
```

Note: Port corresponds the the TCP Port Number, its value should be in the range [0..65535]; port 70 is officially assigned to gopher.

```
Bool      ::= '0' | '1'
G+Field   ::= '+' | '?'

Success   ::= '+' | '-'.
Transfer  ::= DigitSeq | '-1' | '-2'
DataHead  ::= Success Transfer CR-LF
DataBlock ::= DataHead {OCTET}
```

```
G1DirEntity ::= Type DisplayString Tab Selector Tab Host Tab Port Tab
Otherflds CR-LF
G+DirEntity ::= Type DisplayString Tab Selector Tab Host Tab Port Tab G+Field
                CR-LF
```

Notes:
It is *highly* recommended that the DisplayString field contain only printable characters, since many different clients will be using it. However if eight bit characters are used, the characters should conform with the ISO Latin1 Character Set. The length of the User displayable line should be less than 70 Characters; longer lines may not fit across some screens. Warning! The Selector string can be longer than 255 characters.

```
Menu Entity
Menu      ::= DataHead {G+DirEntity}.

Continues ::= Bool
Representation ::= 'Text' | 'List'  | 'Postscript' | 'MacWriteII' | 'RTF'
|{UNASCII}
```

Retrieving a document/menu/etc.:

C: Opens Connection
S: Accepts Connection
C: Sends Selector String Tab '+' Representation [Tab DataFlag]
C: [Optional] Client sends a DataBlock depending on value of DataFlag.
S: Sends DataBlock
Connection is closed by either client or server (typically server).

```
Spaceline    ::= ' ' {UNASCII} CR-LF
Blockname        ::=  '+' {UNASCIINOPLUS}
Attrblock    ::= Blockname ' ' [G+Direntry] CR-LF  {Spaceline}
Attrval      ::= SPACE {UNASCII} CR-LF
E-Mail       ::= {UNASCII} s.t. it conforms to RFC 822
```

```
Adminval     ::= ' Admin:' {UNASCII} '<'  E-Mailaddr '>' CR-LF
Dateval      ::= ' Mod-Date:' {UNASCII} '<' YYYYMMDDhhmmss '>' CR-LF
AdminReq     ::= AdminVal Dateval
Infoblock    ::= '+INFO: ' G+Direntry CR-LF
AdminBlock    ::= '+ADMIN: ' {G+Direntry} CR-LF AdminReq {Attrval}
Language     ::= 'English' | 'French' | 'German' | {UNASCII}
ViewVal      ::= ' ' Representation [' ' Language] ":" ASCIINOANGLE '<'
                 Size 'k>' CR-LF
ViewBlock    ::= '+VIEWS: ' {G+Direntry} CR-LF {ViewVal}
AttrBlocks   ::= InfoBlock ViewBlock {AttrBlock}
```

Retrieving item Information.

C: Opens Connection
S: Accepts Connection
C: Sends Selector String Tab '!' {BlockName}CR-LF
S: Sends DataBlock with data in `AttrrBlocks` format.
Connection is closed by either client or server (typically server).

```
Attributes ::= {AttrBlocks}
```

Retrieving all Item Information entries for a directory.

C: Opens Connection
S: Accepts Connection
C: Sends Selector String Tab '$'{BlockName} CR-LF
S: Sends DataBlock with data in `Attributes` format.
Connection is closed by either client or server (typically server).