Network Working Group   T. Matavka
Internet-Draft
Updates: 1436, 4266 (if approved)   W. Faust
Intended status: Standards Track   June 8, 2015
Expires: December 10, 2015

Gopher-II: The Next Generation Gopher WWIS
draft-matavka-gopher-ii-02

Abstract

The Gopher protocol is over twenty years old. Changing practices and unofficial extensions have caused Gopher as currently used to differ, but remain largely compatible with, the standard established in its official governing document, *The Internet Gopher Protocol (a distributed document search and retrieval protocol)*, known as *RFC 1436*. Therefore, this document attempts to establish a contemporary specification of the Gopher communications protocol, departing as little as possible from current practice.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The over-riding aim of this document is to author a contemporary specification of the Gopher world-wide information system, without falling short of reflecting actual practice and without breaking compliance with RFC 1436 [RFC1436]. This document shall attempt to describe, and, where necessary, update current practice as regards the means of handling errors, line and file terminators, policy files, TITLE selectors, the URL: re-direction scheme, and new selector types not compliant with RFC 1436. This document is not to be construed as a replacement for RFC 1436; it merely complements it.

Gopher is a lightweight, client/server-oriented query/answer protocol, functioning as a world-wide information system (WWIS) and facilitating access to remote servers of any description. The protocol and software permit users of a wide variety of desktop systems to browse, search, and retrieve documents residing on multiple distributed server machines. Gopher is unique among world-wide information systems in that it encourages data to be sent in textual form and that it imposes a strict hierarchy on content, making it a protocol that is fast to transmit, receive, and search. This, in turn, makes it useful in high-latency, low-bandwidth communications, such as mobile links. In fact, Gopher provides the ideal method for transmitting information from and to mobile devices.

1.1. Terminology

NOTE: The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. Furthermore, backticks (`) around a string mean that it is to be interpreted literally.

1.2. Changes from RFC 1436 and 4266

GopherII remains broadly compatible with the original Gopher; a client compatible with the original implementation of Gopher will be able to browse GopherII servers with a minimum of problems. The only difference in strict-compatibility terms arises in the new selector types (including, but not limited to, one distinguishing "plain text" from "ASCII markup document"). That said, although these new selectors are not officially standardised in RFC 1436, most existing Gopher clients will ask for user input when attempting to process an unfamiliar selector, and these selectors have been in de facto use

for some time, such that current Gopher clients will be compatible with them already.

GopherII modifies the original standard in eight ways. Aside from the aforementioned new selector types, GopherII introduces the concept of the so-called policy file. Policy files are configuration files sent from the server to the client in ASCII form with a defined syntax. Three policy files are defined in this document: the capability policy, which defines architectural details of the server (including information about the file system); the administrator contact file, which defines the geographical location of the server as well as the entity responsible for its maintenance, and the robot access restrictions policy, which defines etiquette to be followed by Gopher search engines. The word 'etiquette' is here used because, like human codes of behaviour at mealtime, it is non-binding.

Policy files account for three of the substantive changes from original Gopher. GopherII also adds HTTP-style error codes, a mechanism for titling the Gopher client window or tab, and standardises a backward-compatible method for linking to HTTP addresses. The final change is that GopherII adds support for the metadata system known as Gopher+ (GopherIIbis in this document), although this part of the GopherII specification is entirely optional.

2. Basic Gopher Transactions

There are four broad forms of basic transactions in Gopher:

o Menu Transaction;

o Index Transaction;

o Simple Text Transaction; and

o Binary Transaction.

The precise composition of these transactions is elucidated below.

2.1. Menu Transaction

o Client : [Open Connexion]

o Client : Send [selector<CR><LF>]

o Server : Send <Menu>

o Server : Send .

- o Server : [Close Connexion]

2.2. Index Transaction

- o Client : [Open Connexion]
- o Client : Send [selector<TAB>query parameters<CR><LF>]
- o Server : Send [<Menu>]
- o Server : Send .
- o Server : [Close Connexion]

2.3. Simple Text Transaction

- o Client : [Open Connexion]
- o Client : Send [selector<CR><LF>]
- o Server : Send [<Simple Text>]
- o Server : Send .
- o Server : [Close Connexion]

2.4. Binary Transaction

- o Client : [Open Connexion]
- o Client : Send [selector<CR><LF>]
- o Server : Send [<Raw Binary Data>]
- o Server : DO NOT send .
- o Server : [Close Connexion]

2.5. Details

The fourth step of each transaction, with the exception of the binary type, is OPTIONAL. Servers MAY send a full-stop character after sending a menu, index, or text; if they do, clients MUST accept it. Further information may be found in the appropriate sub-section.

Gopher servers are normally found on TCP port 70. Clients MUST assume this port if no other port is specified. When a client opens a connection to a server, the server MUST accept the connection but

say nothing, waiting for a CR/LF-terminated selector string from the client. The client MAY then send the selector string followed by CR/LF (or nothing to retrieve the root menu from the server, which MUST always be type 1). The server MUST then send the requested content and close the connection.

3. Line Terminators

ASCII, the international standard that governs the interchange of plain-text information between computer systems, is nothing more or less than a table mapping each character (letter, number, space, or symbol) to a numerical code, which is then converted to binary and written to disc. Its necessity was seen long before the advent of the electronic monitor, so some of its more unique quirks must be understood in view of the time period of which it was a product. Historically, input and output was through a specially-adapted typewriter, and the ASCII convention reflects this in the codes it uses to terminate lines of text.

In ASCII, there are two codes, both having physical equivalents in the real world, that signal the end of the line: the Carriage Return (abbreviated C/R, CR, or c/r) and the Line Feed (abbreviated L/F, LF, or l/f). Originally, the term *carriage return* was used for a command that caused the assembly holding the paper (the carriage) to return to the right so the machine was ready to type again on the left side of the paper (assuming a left-to-right language). On the other hand, the *line feed* moved the paper upwards, allowing the carriage to type on the following line.

Different operating systems traditionally signal the end of a line in different ways. UNIX and its descendants (including Mac OS X), the operating systems most likely to run on a server, use the line feed alone. CP/M, DOS, and Microsoft Windows use the sequence of carriage return and line feed (CR/LF). Obsolete versions of Mac OS (up to, and including, System 9) use the carriage return alone.

All programmes using Gopher MUST always use the Microsoft standard of CR/LF, irrespective of the operating system they run on. Both internal Gopher commands and policy files MUST comply with this standard. Other text files SHOULD use standard Gopher format, but this is not strictly required as a matter of technical form; the client MUST be capable of converting to and from all variants of line terminators. The recommendation stands for the benefit of non-compliant clients only.

4. Selector Formats

4.1. Type Codes

The following selectors are defined by RFC 1436:

Type	Treat As	Meaning
0	TEXT	Plain text file
1	MENU	Menu
2	EXTERNAL	CCSO flat database (formerly used as telephone directories); other databases
3	ERROR	Error message
4	TEXT	Macintosh BinHex file
5	BINARY	Binary archive (zip; rar; 7-Zip; gzip; tar)
6	TEXT	UUEncoded archive
7	INDEX	Query a search engine or CGI script
8	EXTERNAL	Telnet to: VT100 series server
9	BINARY	Binary file (see also 5)
+	-	Redundant server
T	EXTERNAL	Telnet to: tn3270 series server
g	BINARY	GIF format graphics file (TODO: Why not use I?)
I	BINARY	Any image file.

The `+` selector indicates a mirror of the previous item in the menu, and MUST behave as though it had the same type as that entry. For example:

```
5Download software /software.zip gopher.example.com 70
+example.net mirror mirror.example.net /example.com/software.zip 70
+Another mirror mirror2.example.com /software.zip 70
```

Additionally, the following selectors have been in common use and are made official here. If a client does not have the capability to display a particular item type, it SHOULD treat it as a more generic item type, passing it off to the operating system (itemtype p "implies" itemtype 0, etc.).

Type	Treat As	Meaning
c	BINARY	Calendar file (Kim Holviala)
d	BINARY	Word-processing document (MS Word; OpenOffice.org; WordPerfect); PDF document
h	TEXT	HTML document
i	-	Informational text (not selectable)
p	TEXT	Page layout or markup document (TeX; LaTeX; PostScript; Rich Text Format)---these documents are all plain text, but contain ASCII tags" that make the document prettier when sent through a special program.
m	BINARY	Electronic mail repository (also known as MBOX) (Kim Holviala)
s	BINARY	Audio recordings (files that consist of audible, but no visible, data) (Wesley Teal)
x	TEXT	eXtensible Markup Language document (Wesley Teal)
;	BINARY	Video files (files that consist of both audible and visible data) (Wesley Teal)

Filetypes `4`, `6`, `h`, `p`, and `x` SHOULD send as text (itemtype 0). This way, the text appears directly on the user's terminal without being downloaded (unless the appropriate command is given to the client, i.e. `CTRL/S`). It is vital to note that text information can be sent via binary (with the minor inconvenience noted above), as binary files contain a greater range of information than ASCII. However, binary files, if sent via text, will be irreparably ruined, as this effectively passes raw eight-bit data through an ASCII filter. In the case of confusion, the owner/ operator of the server should simply mark the file as binary to ensure that it transfers safely.

4.2. GopherIIbis: Metadata in Gopherspace

It is sometimes useful to transmit data about GopherII selectors. This is known as "metadata": the *meta* construction is derived from the Greek for "beyond", and refers to concepts which are abstractions from other concepts intended to complete or add to the latter. For instance, in psychology, metamemory refers to an individual's ability to remember that he has remembered something. In plain English, metadata refers to data about data.

GopherIIbis is an OPTIONAL, but recommended, addition to the basic GopherII specification. That said, it is optional only in the sense that a GopherII client MAY EITHER display the relevant information in accordance with the specification, or ELSE ignore it entirely. To be conformant with GopherII, Gopher clients MUST be capable of handling GopherIIbis metadata. A GopherII client that displays GopherIIbis metadata may be referred to as being compliant with GopherIIbis.

The name of the GopherIIbis extension is pronounced "gopher-two-biss" or "gopher-two-beess". In typeset text, the French word "bis" should be *italicised* so as to set it off visually. The name of the GopherIIbis extension reflects that it is merely an addition, or an iteration, of the GopherII protocol.

5. Gopher Menus

Menu (type 1) content has the following format:

```
T<itemtext>^I<selector>^I<host>^I<port>
```

Where:

- `^I` is the ASCII character corresponding to the `Tab` key
- `T` is the type code, which MUST be run together with the item text
- <selector> is the selector string to send to the specified server
- <host> is the server to send the selector to
- <port> is the port on the server to connect to

If the server understands how to send and receive GopherIIbis metadata, it MUST indicate this fact by adding a fourth tab character (^I) and a plus sign after the port number. For example:

```
T<itemtext>^I<selector>^I<host>^I<port>^I+
```

If the client does not understand GopherIIbis metadata, it MUST ignore the trailing ^I+.

Note on `i` item type: For the `i` item type, Selector, Server, and Port are mostly ignored, but MUST be there anyway. In that case, the host SHOULD be set to placeholder value `example.com`, and the port SHOULD be set to placeholder value `0` (zero). One exception to their being ignored is TITLE entries. These have TITLE as the

selector value; host and port SHOULD again be set to aforementioned placeholder values.

5.1. Note on the terminating full stop

Per RFC 1436, a terminating full stop (.) character followed by CR/LF should be sent on a line by itself after the end of the content, with exceptions for binary data. This terminating full stop has caused no end of trouble ever since. Many, if not most, modern Gopher servers omit this terminating full stop. Therefore, the practice suggested in RFC 1436 is DEPRECATED and the following practice is RECOMMENDED.

- Servers MAY send the full stop; clients MUST accept it

- Servers SHOULD send the full stop after menus and may OPTIONALLY send it after other files

- Clients SHOULD display the full stop at the end of menus, if sent, to notify the user that this is the end of the menu

- Clients SHOULD NOT include the full stop in other output, in case that output has some significance which the full stop may disrupt.

- Clients SHOULD NOT consider a full stop significant, unless it occurs immediately before the connection is terminated.

6. Requesting Data

A standard GopherII client requests data from the server by transmitting the selector string, a carriage return, and a line feed. For instance, to retrieve the file `services.txt`, the client sends

```
services.txt[CR][LF]
```

GopherIIbis handles things in a slightly more complicated way. In addition to a selector string, a GopherIIbis-compliant request contains a *format* string, a data flag indicating the presence or absence of a data block, and an OPTIONAL data block.

The reason for the inclusion of the format string is because GopherIIbis allows one selector to point to multiple versions of the same file, in multiple languages. For instance, the same file in Portable Document Format, PostScript format, Rich Text Format, and plain text may be available, and each of these may be available in British English, American English, Canadian French, and Continental French. The format string, therefore, is the desired MIME type of whichever format is being requested, followed by the ISO country and language codes in the following format:

```
      selector^Imime/type la_CO^I1[CR][LF]datadatadatadata
```

The number 1 above is the file flag. It can be either 1 or 0. If it is 1, it means that the client is not only requesting data, but also *sending* it. This is useful for example when querying a relational database on a Gopher server (this usage is now rare). An example would be:

```
      services.txt^Itext/plain fr_CA^I0[CR][LF]
```

7. **Data Transfer**

When a file is requested by a Gopher client, a Gopher server incompatible with GopherIIbis simply sends the requested data as soon as it gets the request from the client. GopherIIbis servers, on the other hand, have three options when given a GopherIIbis-compliant request (i.e. one that ends in ^I+).

If the size of the file in bytes is known, the server SHOULD transmit a plus sign, the size, and the combination of carriage return and line feed, then the file. For example, if the size of file `report.tex` is known to be 64096 bytes, the server SHOULD transmit:

```
      +64096[CR][LF]\documentclass{article}[CR][LF]\begin{document}...
```

If the size of the file is not known, there are two ways to proceed. One of them is to send the character string `+-1` prior to beginning transmission of the data proper, and end the transmission with a full stop (.) on a line by itself, followed by carriage return and line feed. For example:

```
      +-1[CR][LF]data data[CR][LF]data data data data[CR][LF].[CR][LF]
```

It is RECOMMENDED that most textual data of unknown length be transmitted this way. The exception is when there is a possibility of the full stop appearing on a line by itself; this, of course, would terminate the connexion. There is no choice when sending non-textual (binary) data: it MUST NOT be terminated with a full stop.

In either of the two cases above, the string to send is `+-2`. This instructs the client that the data will be terminated when the connexion is closed, and furthermore, that the length of the data is unknown. For example:

```
      +-2[CR][LF]binarydata <connexion severed by host>
```

8. Requesting and Receiving Metadata

A GopherIIbis client may request the metadata for a specific selector by sending a string in the following form:

```
   <selector>^I![CR][LF]
```

The trailing tab and exclamation mark is what distinguishes a request for data from a request for metadata. The metadata returned is of the following form:

```
+INFO: 0lpryce.txt^IRest in peace, Lane Pryce^Igopher.scdp.com^I70+

+ADMIN:
Admin: Roger Sterling <roger@scdp.com>
Mod-Date: Fri Feb 13 08:22:11 2015 <20130213082211>

+VIEWS:
text/plain: <10k>
application/postscript: <100k>
application/latex: <50k>
application/pdf: <120k>

+ABSTRACT:

Yesterday, our beloved partner Lane Gordon Pryce died of suicide in
his Manhattan home.  He was 55.
```

In general, data intended to be read by the computer will be enclosed in angle brackets (`<` and `>`). A graphical client may, for example, provide a GUI menu of all possible document views with graphical icons of the file type and tool-tips of the file size.

These are far from the only available metadata records; only the `INFO` record is mandatory, and it MUST be transmitted first of all. The `ADMIN` record is RECOMMENDED, and if it is included, it must be transmitted *directly* after the `INFO` record.

It is also possible to retrieve only a *specific* record or range of records. For example, to retrieve only the views and the abstract, a client may send:

```
<selector>^I!+INFO+ADMIN[CR][LF]
```

Finally, it is possible to retrieve metadata for an *entire directory*. Of course, this is relatively bandwidth-intensive (for a 56k link) but a modern Ethernet connexion should have no problem with it. The reason for the requirement of an `INFO` record for every

selector should now be abundantly clear: the `INFO` record serves to separate metadata for one file from metadata for another. For example:

```
<selector>^I&[CR][LF]
```

The only difference between a request for a *single* file's metadata and a request for that of a whole directory is that a single-file request uses an exclamation mark, whereas a whole-directory request uses an ampersand ("and sign", &).

It is even possible to request a specific record from every selector in the directory, by appending the requested fields to the command string as above.

8.1. The `INFO` Record

The `INFO` record is MANDATORY in every metadata listing. It contains the same data as the Gopher selector, with a plus sign at the end, per GopherIIbis style. It MUST always be present, and it MUST always be the first metadata record present. The `INFO` record serves to separate metadata listings when more are sent at the same time.

8.2. The `ADMIN` Record

To promote accountability, the `ADMIN` record is also MANDATORY in every metadata listing. It MUST contain fields for `Admin` (the name and contact information for the administrator of the file) and `Mod-Date` (the date of last modification) as seen in the example below:

```
+ADMIN:
Admin: Roger Sterling <roger@scdp.com>
Mod-Date: 01 January 2015 <YYYYMMDDhhmmss>
```

The time of last modification MUST be in 24-hour format.

If the metadata listing is for the results of a database search, such as Veronica, it SHOULD also include fields for `Score` (a whole-number ranking of the relevance of the result to the search query) and for `Score-Range` (the lowest and highest possible relevance scores), as per the following example:

```
+ADMIN:
Admin: Margaret Olson <m.olson@scdp.com>
Mod-Date: 13 February 2015 <YYYYMMDDhhmmss>
Score: 100
Score-Range: 0 150
```

The first number in the `Score-Range` field is the *lower bound*, and the second number is the *upper bound*.

Several other fields are optional. `Site` is the name of the Gopherhole, `Org` is the name of the business or individual who owns the Gopherhole, `Loc` is the owner's location (city, district, and country), `Geog` is the owner's geographic co-ordinates, and `TZ` is the time zone in the format GMT+[01..11]. For example:

```
+ADMIN:
...
Site: S|C|D|P Main Site
Org: Sterling|Cooper|Draper|Pryce Inc.
Loc: New York, NY, USA
Geog: 40N 173W
TZ: GMT-05
```

The `Author` may also be given, as may be the `Creation-Date` and `Expiration-Date`, in the same format as the `Mod-Date`.

8.3. The `VIEWS` Record

Although the main selector might be for only one format of a file (such as Rich Text Format), the same file may be available in many other formats, such as Plain Text for older systems, LaTeX for typesetters, PDF for displaying on screen, PostScript for printing on a graphical printer, and many more.

The `VIEWS` record in GopherIIbis allows for serving multiple variants of the same file, using what are known as MIME file descriptors, Content-Types, or Internet media types. The `VIEWS` field also allows for viewing the same file in multiple languages and even in multiple dialects of the same language---in this case, the relevant abbreviations are known as ISO-639 language codes and ISO-3166 country codes. These are generally at least somewhat intuitive (`CA` for Canada, `GB` for Great Britain, `en` for English), but a full list may be found on the ISO Web site.

This is an example of a `VIEWS` record allowing for the selection of a plain text, Rich Text, and PDF of the same file in American English, Peninsular Portuguese, and Brazilian Portuguese:

```
+VIEWS
text/plain en_US: <32K>
text/plain pt_PT: <34K>
text/plain pt_BR: <34K>
text/rtf en_US: <55K>
text/rtf pt_PT: <60K>
```

```
text/rtf pt_BR: <66K>
application/pdf en_US: <120K>
application/pdf pt_PT: <132K>
application/pdf pt_BR: <133K>
```

The `VIEWS` record SHOULD be ranked according to the administrator's idea of which view is preferred. On an American site catering to English speakers, the `en_US` files should be listed first of all. Likewise, on a site of any language catering to scientists, LaTeX source should always come first of all.

8.4. The `ABSTRACT` Record

It is RECOMMENDED that every selector on a GopherIIbis-compliant server have an `ABSTRACT` record. The `ABSTRACT` record contains a *brief* description of the item (no more than a paragraph long) to assist the reader in determining its purpose. Similarly, it is also RECOMMENDED that the root directory of every Gopher server (that is, what one gets when one requests metadata for the server itself with no selector) contain an `ABSTRACT` record with the name, postal address, eMail address, and telephone number of the person responsible for the site. For example:

```
+ABSTRACT
The life and times of Professor Albert Einstein, Swiss patents clerk
and discoverer of four great scientific theories in one miraculous
year.
```

9. Errors

Although undesirable in communication, errors do occur in Gopher, and their handling is crucial for a user-friendly, and standards-compliant, Gopher experience.

When an error is encountered, the server MUST return a menu whose first item bears itemtype `3`. All other ways of signalling an error, such as redirecting to a Gopher error menu, an image, or (worst of all) an HTML page, are PROHIBITED.

The selector string for itemtype `3` is the text of the error. It is the responsibility of the server application to have understandable and accurate strings for error handling. As they are well-understood and common, HTTP-style error codes are acceptable and RECOMMENDED; however, they SHOULD also be followed by a clear, legible description of the error in both English and the local language.

Errors are handled in GopherIIbis in a slightly different fashion. When an error occurs in response to a GopherIIbis-compliant query,

the server sends two minus signs, followed by an error code, a description of the error, and a full stop. The error code SHOULD be in the three-digit style elucidated in the next sub-subsection, but the numbers 1, 2, and 3 MUST also be understood and handled correctly, also as defined in "Error Codes". An example of a GopherIIbis error follows:

```
--404[CR][LF]The file requested could not be found.[CR][LF].[CR][LF]
```

The decision of whether to send a GopherII error string or a GopherIIbis error string is governed by the type of query received. If the query was compliant with GopherIIbis, a GopherIIbis error MUST be sent. In all other cases, a GopherII error MUST be sent.

9.1. Error Codes

This is a listing of numeric error codes used in Gopher; due to Gopher's simplicity, it lacks most of the errors possible in HTTP. Codes beginning with 4 can generally be traced to the client; codes beginning with 5 are usually due to the server.

400 Bad Request The request could not be understood by the server due to malformed syntax.

401 Unauthorised The request requires authentication. For example, the received query value (as password) does not match the expected value.

403 Forbidden The request was received, but not filled.

404 Not Found The server could not find anything matching the requested URL. If the condition is known to be permanent, use error code 410 (Gone).

408 Request Time-out The client did not produce a request within the time that the server was prepared to wait.

410 Gone The requested resource is no longer available at the server and no forwarding address is known. This condition is expected to be considered permanent. If this is unknown, use error code 404 (Not Found).

500 Internal Server Error The server encountered an unexpected condition which prevented it from fulfilling the request.

501 Not Implemented The server does not support the functionality required to fulfil the request.

503 Service Unavailable The server is currently unable to handle the
request due to temporary overload/maintenance.

An earlier version of the GopherIIbis extension, known as Gopher+, used error codes `1`, `2`, and `3`. Error code `1` signifies an unavailable item (similar to the 400-series errors), error code `2` signifies an unavailable server (similar to the 500-series errors), and error code `3` signifies an item that has moved. Provision was made to create new error codes. This is now DEPRECATED; the *ad hoc* creation of new errors does not accord with the ethos of a standardised Internet protocol.

10. Titles in Gopher

No mention of menus with titles exists per RFC 1436. When one simply browses about Gopherspace, this does not matter; for bookmarking and Gopher crawlers, such as Veronica-2, however, this presents a large problem.

A Gopher TITLE resource has the following format:

```
i<titletext>^ITITLE^Iexample.com^I0
```

It is identical to a standard informational resource (itemtype `i`); the selector string, however, is set to the specific value, `TITLE`.

The composition of the above format is as follows:

- `^I` is the ASCII character corresponding to a press of the `Tab` key
- The type code MUST be `i` (information)
- The selector string MUST be `TITLE`
- There is no server to connect to; the dummy text used in place of the server SHOULD be `example.com`
- There is no port to connect to; the placeholder number SHOULD therefore be `0` (zero).

A Gopher client that conforms to the above `TITLE` specification SHALL render it in one of two ways, depending on the placement of the resource. If the `TITLE` is the *first* resource in the document, it SHALL be considered its principal `TITLE` and used *wherever a principal title is needed* (window headings, bookmarks, etc.); furthermore, it SHOULD be rendered in a different size, font, and/or colour to the remainder of the document. In *all other* cases, it

SHALL be considered a subordinate `TITLE` and SHOULD be rendered in a different size, font, and/or colour to the remainder of the document, but smaller and/or with less emphasis than the main title.

If a non-compliant Gopher client receives a `TITLE` resource as per above, it will render it as plain informational text. As the main `TITLE` must be on the first line of a menu, it will appear visually similar to a title in any case, although not rendered as such.

11. Linking to Web Addresses

It is now possible, and standard, to link to documents, preferably in HTML, on the World Wide Web, Gopher's younger, more widespread cousin, from Gopher itself, using a two-part system: a `URL:` selector on the Gopher (local) end, and a *redirect page* (following rules as set out below) on the HTTP (remote) end. There are no compliance requirements for Gopher servers, with one exception: servers MUST follow the bulleted list located immediately after the example redirect page.

A Gopher client SHALL, when it sees a selector with a path starting with `URL:`, interpret the path as a URL. It SHALL ignore the host and port components of the Gopher selector, using those components from the URL instead, if applicable.

`URL:` selectors SHOULD NOT be used if it is possible to link to the required content and protocol by any other means. In particular, the following protocols SHALL NOT be used with the URL: selector.

o gopher

o telnet (VT100-compatible)

o tn3270

Authors SHOULD NOT link to any document not of HTML type unless absolutely necessary; linking to non-HTML documents will break compatibility with non-compliant Gopher browsers.

A Gopher `URL:` selector MUST take the following format:

```
h<itemtext>^IURL:<address>^I<localhost>^I<localport>
```

URL:` selectors are, for the most part, identical to standard HTML selectors, but composed of particular data:

- The item type corresponds to the type of document on the remote end. Most typically, this is a Web page authored in HTML; therefore, the item type is most commonly `h`.

- <itemtext> is the text of the link; this can be almost anything.

- <address> is the full URL, preceded by the string `URL:`. For example, this could be `URL:http://www.example.com`

- <localhost> is the server that the link *originated* from; this MUST be ignored by a compliant client, but MUST also be sent by a compliant server

- <localport> is the port that the link *originated* from; this MUST be ignored by a compliant client, but MUST also be sent by a compliant server

It is possible for a non-compliant Gopher client to follow a link to an HTML page, as long as the server is compliant, by the following means: when the client receives a command to follow a `URL:` selector, it will contact the server that provided the menu, as the originating host and port are *mandatory* per this specification.

When a Gopher server receives a request from a client beginning with the string `URL:`, it SHALL write out an HTML document that redirects the browser to the appropriate place. A conforming example of such a document is as follows:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="refresh" content="2;URL=http://www.example.com/">
</HEAD>
<BODY>

You are following an external link to a Web site.  You will be
automatically taken to the site shortly.  If you do not get sent
there, please click <A HREF="http://www.example.com/">here</A> to go
to the web site.
<P>
The URL linked is:http://www.example.com/">
<P>
<A HREF="http://www.example.com/">http://www.example.com/</A>
<P>
Thanks for using Gopher!
</BODY>
</HTML>
```

This document may be any desired by the server authors, but MUST adhere to the following requirements.

- It SHALL provide a refresh of a duration of 10 seconds or less

- It SHALL NOT use `IMG` tags, frames, or have any reference whatsoever to content outside that particular file, with the sole exception of the link to the real destination.

- It SHALL NOT use JavaScript.

- It SHALL adhere to the W3C HTML 4.0 standard.

When a non-compliant Gopher client finds a reference to a HTML file (type `h`), it will open up the file via Gopher, receiving the redirect document using a Web browser. The Web browser will then be redirected to the actual link destination.

Compliant Gopher clients will simply render the target directly.

12. Algorithm to use with selectors

Here is a description for a hypothetical algorithm for parsing item types, splitting them into levels of interaction.

PROTOCOL

Type	Description	What to do
0	Brief text	Render directly line by line.
1	Menu	Request and analyse menu. If it contains '3' error node, print error. Else, render menu in new window.
7	Index/Search Server	

DATA NODES

Type	Description	What to do
4, 9, g, I, c, d, m, s, ;	Binary file	Request and analyse file. If it contains '3' error node, print error. Else, does plug-in exist? If yes, display. If no, save to disc.
6, p, x	Text file	Request and analyse file. If it contains '3' error node, print error. Else, print on screen.
h, 2, 8, T	Link	Treat as URL.
5	Archive File	Request and analyse File. If it contains '3' error node, print error. Else, does plug-in exist? If yes, display. If no, save to disc.

For instance, if the client is incapable of handling images as it is text-only, the algorithm above would have it save to disc.

13. Representation of Gopher Addresses

This section is greatly indebted to RFC 4266 [RFC4266].

A Gopher address, or uniform resource locator, takes the form:

gopher://<host>:<port>/<gopher-path>

where <gopher-path> is one of:

o <gophertype><selector>

o <gophertype><selector>%09<search>

o <gophertype><selector>%09<search>%09<gopher+_string>

If :<port> is omitted, the port defaults to 70. <gophertype> is a single-character field to denote the Gopher type of the resource to which the URL refers. The entire <gopher-path> may also be empty, in which case the delimiting `/` is also optional and the <gophertype> defaults to `1`.

<selector> is the Gopher selector string. Selector strings are arbitrary sequences of characters; they MUST NOT, however, contain the characters corresponding to horizontal tab, line feed, or carriage return. Gopher clients specify which item to retrieve by sending the Gopher selector string to a Gopher server. It is important to know that within the <gopher-path> itself, there are no reserved characters, so one may be arbitrarily creative when creating selector names.

Note that some Gopher <selector> strings begin with a copy of the <gophertype> character, in which case that character will occur twice consecutively. The Gopher selector string may be an empty string; this is how Gopher clients refer to the top-level directory on a Gopher server.

If the URL refers to a search to be submitted to a Gopher search engine, the selector is followed by an encoded tab `%09` and the search string. To submit a search to a Gopher search engine, the Gopher client sends the <selector> string (after decoding), a tab, and the search string to the Gopher server.

14. Gopher Policy Files

It is often useful to provide information to Gopher clients that MAY, but need not, be read by a human being. It is for this reason that policy files exist. This document enumerates two types of policy files, formally known as the Capability Policy and the Robot Access Restriction Policy, but also informally known under their filenames: `caps.txt` and `robots.txt`, respectively.

14.1. Capability Policy

It is RECOMMENDED, when hosting a public-access Gopher server, to include a capability policy. Although it is, ultimately, the choice of the owner or operator of the server, a capability policy (or caps file) can be useful for clients querying the server for certain information without using extensions such as Gopher+.

The purpose of a capability policy is so that a server can instruct a client on how properly to parse selectors in its filesystem; it ensures that the client can understand how files on the server are organised. The scheme used in the current implementation of caps can

handle POSIX (UNIX and related operating systems), FAT/NTFS (used by Microsoft Windows), and HFS (used by all versions of Apple Mac OS, including OS X, which is otherwise POSIX-compatible). For technical reasons, capability policies cannot handle VMS or Files-11 paths; however, owing to their open interface, the specification can be arbitrarily extended.

A capability policy is quite simple in its composition: it is a plain text file with no more than seventy characters per line in the root directory of a Gopher server with the name

```
caps.txt
```

and beginning with the six characters

```
CAPS[CR][LF]
caps.txt
```

Because of the constrained name and location of the policy, it is a trivial matter to verify if one exists or not; the address is always of the form <gopher://gopher.example.com/1/caps.txt>, with the real name of the server substituting for `example`. The server should accept both `caps.txt` and `/caps.txt` as selectors, and return the same content for both.

A caps file contains *keys*, *values*, and *comments*.

Keys can be compared to labelled containers for data; for instance, the key `ServerSoftware` is a container for the name of the Gopher software running on the server. Keys in capability policies are always alphanumeric (i.e., composed of letters and numbers only) and generally are in CamelCase (each individual word within the key capitalised). The data in these containers is called a value; values can use letters, numbers, and symbols. Keys and values are connected by the equals (=) sign. Any amount of whitespace (spaces and tabs) around the equals sign is acceptable.

Anything not conforming to the syntax

```
SomeKey = Value
```

is ignored (treated as a comment). To be compliant with GopherII, comments must begin with a hash (#) sign. More importantly, they must be on a line to their own.

Below is an example caps file.

```
    CAPS
    CapsVersion=1
    ExpireCapsAfter=3600

    PathDelimeter=/
    PathIdentity=.
    PathParent=..
    PathParentDouble=FALSE
    PathEscapeCharacter=\\
    PathKeepPreDelimeter=FALSE

    ServerSoftware=Bucktooth
    ServerSoftwareVersion=0.2.9
    ServerArchitecture=AIX
    ServerDescription=IBM Power 520 Express, 2x4.2GHz POWER6 CPU, 8G RAM
    ServerGeolocationString=Southern California, USA

    ServerSupportsStdinScripts=TRUE

    ServerAdmin=gopher@floodgap.com

    DefaultEncoding=utf-8
```

The `CapsVersion` field is self-explanatory, with one note: it should always be the *first* field in the file, so that an incompatible later format might be detected by the client. The `ExpireCapsAfter` field tells the client the recommended cache expiry time (that is, the time between fetching and re-fetching the caps file) in *seconds*. `3600` as above means one hour, and so on.

The `Path` variables `PathDelimeter` [sic!], `PathIdentity`, `PathParent`, `PathParentDouble`, `PathEscapeCharacter`, and `PathKeepPreDelimeter` [sic!] refer to attributes of the file system. The above example is correct for a UNIX system, including Mac OS X. `PathDelimeter` refers to how the server separates folders from each other; Unix machines use `/`, Microsoft machines use `\`, and obsolete Macs use `:`. `PathIdentity` refers to the shorthand used by an operating system to mean "this directory"; UNIX machines use `.`. `PathParent` refers to the shorthand for "the directory immediately above", and is `..` on UNIX and Microsoft systems. `PathParentDouble` refers to an oddball feature of obsolete Macs: two consecutive path delimiters are used to refer to the parent directory. For all systems other than pre-OS X Macintoshes, `PathParentDouble` should be FALSE. `PathEscapeCharacter` tells the client the escape character for quoting delimiters when they appear in selectors; most of the time, this is `\\`. `PathKeepPreDelimiter`

tells the client not to cut everything up to the first path delimiter; most of the time, this should be `FALSE`.

The `Server` variables `ServerSoftware`, `ServerSoftwareVersion`, `ServerArchitecture`, `ServerDescription`, and `ServerGeolocationString` are freetext descriptions of the server software and version, operating system ("architecture"), server hardware (`Server Description`), and location on the Earth.

Finally, `ServerAdmin` is an eMail contact address for the server administrator, and `DefaultEncoding` is the default text encoding for content types 0 and 1.

14.2. **Robot Access Restrictions Policy**

WWIS robots, also known as spiders, crawlers, or wanderers, are computer programmes that, without human intervention, recursively travel throughout linked pages or directories on an information system (that is, by repeatedly travelling up and down a tree) and store the copies of these files at an independent location. The process of programmatically gathering information in this manner is called crawling or spidering.

Many sites, in particular search engines (such as Google on the World Wide Web, or Veronica on Gopher), use spidering as a means of providing up-to-date data. Robots are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Robots can also be used for redundancy; data can be preserved by a third party in case the original server becomes inaccessible.

In 1993 and 1994, however, there were occasions where robots had visited locations on the Web at which they were not welcome. Inexperienced or heavy-handed use of robots caused situations where servers were swamped with requests at a high rate of speed; or, the same files were retrieved repeatedly. Both could cause denial of service. In other situations, robots traversed parts of servers that were unsuitable, such as temporary information or server-side scripts, especially those with side-effects (such as polls). Abuse of robots was also an issue, and continues to be one now; for instance, electronic mail addresses have been harvested with knowing intent to distribute unsolicited mail ('spam').

These incidents indicated the need for established mechanisms for Gopher servers to indicate to robots which parts of their server should not be accessed. This specification addresses this requirement with an operational solution, adapted from the identical

method used on sites using the Hypertext Transfer and File Transfer Protocols.

The method used to exclude robots from a Gopher server is formally known as the Robot Access Restrictions Policy (RARP) and consists of placing a plain-text file specifying, in simple and user-friendly syntax, which robots may access which directory. The policy file, if it exists, MUST be accessible via Gopher on the local address

```
/robots.txt
```

A possible drawback of this single-file approach is that only a server administrator can maintain such a list, not the individual document maintainers on the server. This can be resolved by a local process to construct the single file from a number of others, but if, or how, this is done is outside of the scope of this document.

Furthermore, Gopher administrators should bear in mind that the Robot Access Restrictions Policy works largely on the honour system. Many crawlers can be set to ignore the policy, and it is trivial to write this capability into a new crawler.

The policy file consists of one or more records, separated by one or more blank lines, terminated by the Gopher-standard CR/LF. Each record contains two or more lines of the form

```
<field>:<value>
```

The field name is not case-sensitive. Comments (lines to be ignored by robots themselves, but useful to robot operators and others) start with the hash (#) character and end with the line terminator (CR/LF). A value can share a line with a comment. A record starts with at least one `User-agent` field, followed by at least one `Disallow` field. There are two further, optional fields: `Crawl-delay`, as well as `Allow`.

The value of the `User-agent` field is the name of the robot whose access policy is being described. If more than one `User-agent` field is present, the record is describing an identical access policy for each robot. This field is to be interpreted broadly. The recommended implementation of access policies in the robot's code is for a case-insensitive sub-string match, without version information. Since one is describing an access policy for at least one robot, at least one `User-agent` field is required. The value `*` (quotes excluded) describes access policy for any robot not matching any previous records; therefore, if listed, it SHOULD be listed last of all. If it is not listed last of all, anything below it will be ignored.

The value of the `Disallow` field specifies a partial URL that is not to be visited. This can be a full path, or a partial path. Any address that begins with this value will not be retrieved; for instance, the line Disallow: /help would disallow `/help/index.html`; `/help/faq.html`; as well as `/help.html`. Conversely, the line Disallow: /help/ would allow `/help.html`, but nothing in the directory `/help/`. An empty `Disallow` field indicates that all addresses can be retrieved. As one is defining policy and not simply listing the names of robots, at least one `Disallow` field is required per record.

One can also add specific exceptions to the locations disallowed by using the `Allow` field.

The `Crawl-delay` field is also supported; this field indicates the number of seconds to wait between successive requests to the same server; the value must be an integer with no units.

The following is an example of a well-built policy file: # Robot Exclusion File for gopher://gopher.scdp.com # If you wish to crawl gopher.scdp.com, please contact # lane.pryce@scdp.com to apply for an exemption. Our terms of # service are available at gopher://gopher.scdp.com/0/tos.txt. User-agent: baiduspider User-agent: googlebot User-agent: msnbot User-agent: bingbot User-agent: naverbot User-agent: seznambot User-agent: slurp User-agent: teoma User-agent: yandex Disallow: /cgi-bin/ # Dynamically generated scripts Disallow: /images/ # This consumes bandwidth! Disallow: /tmp/ # Temporary files---blink, gone! Disallow: /private/ # No peeking! Allow: /images/logo.jpg # Main logo. Mirror this if possible. Crawl-delay: 10 User-agent: * Disallow: / # If you have received authorisation to crawl this site, and are # getting denied, please contact support@scdp.com, or dial # (212) 555 0169. This site is copyright Sterling, Cooper, Draper, # and Pryce, 2012.

In plain terms, this server allows major search engines Baidu, Google, Bing, Naver, Seznam, Teoma, Yahoo, and Yandex to mirror the site freely, with the exception of everything in the directories /cgi-bin/, /tmp/, and /private/, as well as everything with the exception of the single file logo.jpg in the directory /images/. So as to not unduly slow the server down, the policy file requests that search engines wait ten seconds between requests. All other robots are prohibited from accessing the site.

Examples such as the following SHOULD NOT be used except in very rare situations. Robots generally cause more good than harm, and excluding them entirely, as this anti-social user would, does not make Gopher a healthy place.

```
# Piss off!
User-agent: *
Disallow: /
```

14.3. Administrator Contact File

It is worth remembering that computers, like anything else, are fallible and prone to error. When failure occurs in Gopherspace, the person in the best position to rectify it is the system administrator. Furthermore, users may have questions or comments, also best directed to the system administrator. For this reason, each Gopher server MUST have a file in its top-level directory with the name *about.txt* and a RECOMMENDED selector string of *About* or *About this server* (equivalents in the local language are permissible, but an English translation is similarly RECOMMENDED). It is the Gopher equivalent of a Unix user's finger output.

Since this file is intended to be readable by humans and not computers, it does not have a defined file format. However, it should have a short description of the server's contents, as well as the contact details of the server administrator and any other key employees, such as the legal department. A well-structured contact file looks as follows:

```
      Sterling|Cooper|Draper|Pryce
      ============================

      Welcome to SCDP!  We are a full-service advertising and marketing
      agency staffed by a team of diverse, senior professionals with a
      flair for solid strategy and compelling creative output. Our team
      produces unique television, radio, print, and Web advertisements
      for a range of industries.

      Our ability to identify and communicate your greatest benefit to
      your customers is our greatest benefit to you. We find out what
      makes you truly unique. We have built an excellent team: each
      member is an advertising specialist in their own right.
      Photography, programming, writing, design, strategy---you name it,
      we have a creative for that.

      System Administrator: Margaret Olson
      Telephone:            (212) 555 0169 x808
      Address:              13, Madison Avenue,
                            New York, N.Y.,
                            U.S.A.
      eMail:                peggy.olson@scdp.com
      Skype:                peggyXolson

      All prospective clients:
      Please contact Creative Director Donald Draper at extension 069.

      Legal issues:
      For all legal and financial issues, please contact Lane Pryce
      at extension 777.
```

15. IANA Considerations

Nothing within this document should be taken to imply that any actions are to be undertaken by the Internet Assigned Numbers Authority.

16. Security Considerations

Security in GopherII is dependent on the connexion on which it runs. Standard GopherII (that is, running on "straight" TCP) is insecure simply by virtue of the protocol. Sensitive information, such as credit card numbers, must not be sent over a standard Gopher link. It is permissible to run GopherII over SSL, in which case all security considerations that apply for working HTTPS apply also for GopherIIs.

17. Acknowledgements

Thanks go to John Klensin for his invaluable assistance in regards to the IETF process, his constructive criticism, and his calm demeanour even when others just could not keep their tempers in check.

Thanks also go to the members of the Gopher mailing list for keeping the Gopher protocol alive. Thanks go specifically to the Gopher developers: to Matjaz Mesnjak for his Windows-compatible, graphical Gopher client and his simple Motsognir Gopher server; to Dr Cameron Kaiser for Veronica-II, the next generation of Gopher search engine, for the Bucktooth Gopher server, for the Overbite extension for Mozilla Firefox, and for his tireless work on GopherVR, the only full virtual-reality Gopher client; to Kevin Veroneau for his Gopher Application Framework; and to Kim Holviala for the Gophernicus Gopher server.

Finally, my thanks go to Thomas E. Dickey and the others who have put in valuable work on the Lynx browser. I thank them because, rather than remove Gopher support in a misguided attempt to plug security holes, they have in fact continued to improve this side of their software, and they have succeeded in making the finest text-mode Gopher client bar none.

18. References

18.1. Normative References

[Anklesaria1993]
Anklesaria, F., "Gopher+: upward compatible enhancements to the Internet Gopher protocol", 1993.

Anklesaria, Farhad; Lindner, Paul; McCahill, Mark P.; Torrey, Daniel; Johnson, David; Alberti, Bob (1993). ** Retrieved 23 May, 2012, from <gopher://gophernicus.org/0/doc/gopher/gopher+.txt>

[RFC1436] Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D., and B. Alberti, "The Internet Gopher Protocol (a distributed document search and retrieval protocol)", RFC 1436, March 1993.

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

18.2. Informative References

[CapsRef] Kaiser, C., "Welcome to caps!", 2010.

Kaiser, Cameron (2010). *Welcome to caps!* Retrieved 23 May, 2012, from <gopher://gophernicus.org/0/doc/gopher/gopher-caps.txt>

[Floodgap]
Floodgap, "Floodgap's caps file".

<gopher://gopher.floodgap.com/0/caps.txt> - Floodgap's caps file

[Goerzen2012]
Goerzen, J., "Links to URL", 2002.

Goerzen, John (2002). *Links to URL.* Retriever 23 May, 2012, from <gopher://gophernicus.org/0/doc/gopher/hURL.txt>

[GopherHistory]
??, ?., "A paper on the history of Gopher".

<gopher://jgw.mdns.org/9/REFS/56817068-PIL3254-Internet-Gopher-PPL- Alberti.pdf> -- A paper on the history of Gopher

[RelatedDocs]
??, ?., "gophernicus.org".

<gopher://gophernicus.org/1/doc/gopher/> - A number of documents relating to gopher, including the RFCs

[RFC4266] Hoffman, P., "The gopher URI Scheme", RFC 4266, November 2005.

[UpdatedGopher]
??, ?., "The "Updated Gopher RFC" thread", 2012.

The "Updated Gopher RFC" thread (started May 8 2012) on the gopher-project mailing list

Appendix A. Summary of Changes from RFC 1436

In broad strokes, RFC 1436 is compatible with this document; an "old" Gopher client should be fully capable of browsing a GopherII server. GopherII can be considered simply a refinement of the RFC 1436

concept; while RFC 1436 lays out a viable protocol, it leaves a lot of small-scale implementation detail up to the makers of client software. While a sort of gentleman's agreement did manifest itself, and while this gentleman's agreement was in some places almost universal (the `i` itemtype, for example, with only Microsoft Internet Explorer as the nonconforming Gopher client) it did lack standardisation, which is what this document remedies. More specifically:

- c, d, h, i, p, m, s, x, ; itemtypes.
- extension formerly known as Gopher+
- terminating full-stop behaviour
- what to put in the title bar (`TITLE` resource)
- links to HTTP urls
- policy files

Appendix B. Change Log

B.1. Changes from -00 to -01 of this specification

Converted to RFC standard format for legibility; added security considerations section.

B.2. Changes from -01 to -02 of this specification

Added acknowledgements and changes from original Gopher RFC's. Removed placeholder text.

Authors' Addresses

Ted Matavka

Email: n.theodore.matavka.files@gmail.com

Wolfgang Faust

Email: wolfgangmcq@gmail.com