Network Working Group
Request for Comments: 1738
Category: Standards Track
T. Berners-Lee
CERN
L. Masinter
Xerox Corporation
M. McCahill
University of Minnesota
Editors
December 1994

Uniform Resource Locators (URL)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies a Uniform Resource Locator (URL), the syntax and semantics of formalized information for location and access of resources via the Internet.

1. Introduction

This document describes the syntax and semantics for a compact string representation for a resource available via the Internet. These strings are called "Uniform Resource Locators" (URLs).

The specification is derived from concepts introduced by the World-Wide Web global information initiative, whose use of such objects dates from 1990 and is described in "Universal Resource Identifiers in WWW", RFC 1630. The specification of URLs is designed to meet the requirements laid out in "Functional Requirements for Internet Resource Locators" [12].

This document was written by the URI working group of the Internet Engineering Task Force. Comments may be addressed to the editors, or to the URI-WG <uri@bunyip.com>. Discussions of the group are archived at <URL:http://www.acl.lanl.gov/URI/archive/uri-archive.index.html>

2. General URL Syntax

Just as there are many different methods of access to resources, there are several schemes for describing the location of such resources.

The generic syntax for URLs provides a framework for new schemes to be established using protocols other than those defined in this document.

URLs are used to ‘locate’ resources, by providing an abstract identification of the resource location. Having located a resource, a system may perform a variety of operations on the resource, as might be characterized by such words as ‘access’, ‘update’, ‘replace’, ‘find attributes’. In general, only the ‘access’ method needs to be specified for any URL scheme.

2.1. The main parts of URLs

A full BNF description of the URL syntax is given in Section 5.

In general, URLs are written as follows:

```
<scheme>:<scheme-specific-part>
```

A URL contains the name of the scheme being used (<scheme>) followed by a colon and then a string (the <scheme-specific-part>) whose interpretation depends on the scheme.

Scheme names consist of a sequence of characters. The lower case letters "a"--"z", digits, and the characters plus ("+"), period ("."), and hyphen ("-") are allowed. For resiliency, programs interpreting URLs should treat upper case letters as equivalent to lower case in scheme names (e.g., allow "HTTP" as well as "http").

2.2. URL Character Encoding Issues

URLs are sequences of characters, i.e., letters, digits, and special characters. A URLs may be represented in a variety of ways: e.g., ink on paper, or a sequence of octets in a coded character set. The interpretation of a URL depends only on the identity of the characters used.

In most URL schemes, the sequences of characters in different parts of a URL are used to represent sequences of octets used in Internet protocols. For example, in the ftp scheme, the host name, directory name and file names are such sequences of octets, represented by parts of the URL. Within those parts, an octet may be represented by the chararacter which has that octet as its code within the US-ASCII [20] coded character set.

In addition, octets may be encoded by a character triplet consisting of the character "%" followed by the two hexadecimal digits (from "0123456789ABCDEF") which forming the hexadecimal value of the octet. (The characters "abcdef" may also be used in hexadecimal encodings.)

Octets must be encoded if they have no corresponding graphic character within the US-ASCII coded character set, if the use of the corresponding character is unsafe, or if the corresponding character is reserved for some other interpretation within the particular URL scheme.

No corresponding graphic US-ASCII:

URLs are written only with the graphic printable characters of the US-ASCII coded character set. The octets 80-FF hexadecimal are not used in US-ASCII, and the octets 00-1F and 7F hexadecimal represent control characters; these must be encoded.

Unsafe:

Characters can be unsafe for a number of reasons. The space character is unsafe because significant spaces may disappear and insignificant spaces may be introduced when URLs are transcribed or typeset or subjected to the treatment of word-processing programs. The characters "<" and ">" are unsafe because they are used as the delimiters around URLs in free text; the quote mark (""") is used to delimit URLs in some systems. The character "#" is unsafe and should always be encoded because it is used in World Wide Web and in other systems to delimit a URL from a fragment/anchor identifier that might follow it. The character "%" is unsafe because it is used for encodings of other characters. Other characters are unsafe because gateways and other transport agents are known to sometimes modify such characters. These characters are "{", "}", "|", "\", "^", "~", "[", "]", and "`".

All unsafe characters must always be encoded within a URL. For example, the character "#" must be encoded within URLs even in systems that do not normally deal with fragment or anchor identifiers, so that if the URL is copied into another system that does use them, it will not be necessary to change the URL encoding.

Reserved:

Many URL schemes reserve certain characters for a special meaning: their appearance in the scheme-specific part of the URL has a designated semantics. If the character corresponding to an octet is reserved in a scheme, the octet must be encoded. The characters ";", "/", "?", ":", "@", "=" and "&" are the characters which may be reserved for special meaning within a scheme. No other characters may be reserved within a scheme.

Usually a URL has the same interpretation when an octet is represented by a character and when it encoded. However, this is not true for reserved characters: encoding a character reserved for a particular scheme may change the semantics of a URL.

Thus, only alphanumerics, the special characters "$-_.+!*'(),", and reserved characters used for their reserved purposes may be used unencoded within a URL.

On the other hand, characters that are not required to be encoded (including alphanumerics) may be encoded within the scheme-specific part of a URL, as long as they are not being used for a reserved purpose.

2.3 Hierarchical schemes and relative links

In some cases, URLs are used to locate resources that contain pointers to other resources. In some cases, those pointers are represented as relative links where the expression of the location of the second resource is in terms of "in the same place as this one except with the following relative path". Relative links are not described in this document. However, the use of relative links depends on the original URL containing a hierarchical structure against which the relative link is based.

Some URL schemes (such as the ftp, http, and file schemes) contain names that can be considered hierarchical; the components of the hierarchy are separated by "/".

3. Specific Schemes

The mapping for some existing standard and experimental protocols is outlined in the BNF syntax definition. Notes on particular protocols follow. The schemes covered are:

ftp	File Transfer protocol
http	Hypertext Transfer Protocol
gopher	The Gopher protocol
mailto	Electronic mail address
news	USENET news
nntp	USENET news using NNTP access
telnet	Reference to interactive sessions
wais	Wide Area Information Servers
file	Host-specific file names
prospero	Prospero Directory Service

Other schemes may be specified by future specifications. Section 4 of this document describes how new schemes may be registered, and lists some scheme names that are under development.

3.1. Common Internet Scheme Syntax

While the syntax for the rest of the URL may vary depending on the particular scheme selected, URL schemes that involve the direct use of an IP-based protocol to a specified host on the Internet use a common syntax for the scheme-specific data:

```
//<user>:<password>@<host>:<port>/<url-path>
```

Some or all of the parts "<user>:<password>@", ":<password>", ":<port>", and "/<url-path>" may be excluded. The scheme specific data start with a double slash "//" to indicate that it complies with the common Internet scheme syntax. The different components obey the following rules:

user
: An optional user name. Some schemes (e.g., ftp) allow the specification of a user name.

password
: An optional password. If present, it follows the user name separated from it by a colon.

The user name (and password), if present, are followed by a commercial at-sign "@". Within the user and password field, any ":", "@", or "/" must be encoded.

Note that an empty user name or password is different than no user name or password; there is no way to specify a password without specifying a user name. E.g., <URL:ftp://@host.com/> has an empty user name and no password, <URL:ftp://host.com/> has no user name, while <URL:ftp://foo:@host.com/> has a user name of "foo" and an empty password.

host
: The fully qualified domain name of a network host, or its IP address as a set of four decimal digit groups separated by ".". Fully qualified domain names take the form as described in Section 3.5 of RFC 1034 [13] and Section 2.1 of RFC 1123 [5]: a sequence of domain labels separated by ".", each domain label starting and ending with an alphanumerical character and possibly also containing "-" characters. The rightmost domain label will never start with a digit, though, which syntactically distinguishes all domain names from the IP addresses.

port
: The port number to connect to. Most schemes designate protocols that have a default port number. Another port number may optionally be supplied, in decimal, separated from the host by a colon. If the port is omitted, the colon is as well.

url-path
: The rest of the locator consists of data specific to the scheme, and is known as the "url-path". It supplies the details of how the specified resource can be accessed. Note that the "/" between the host (or port) and the url-path is NOT part of the url-path.

The url-path syntax depends on the scheme being used, as does the manner in which it is interpreted.

3.2. FTP

The FTP URL scheme is used to designate files and directories on Internet hosts accessible using the FTP protocol (RFC959).

A FTP URL follow the syntax described in Section 3.1. If :<port> is omitted, the port defaults to 21.

3.2.1. FTP Name and Password

A user name and password may be supplied; they are used in the ftp "USER" and "PASS" commands after first making the connection to the FTP server. If no user name or password is supplied and one is requested by the FTP server, the conventions for "anonymous" FTP are to be used, as follows:

The user name "anonymous" is supplied.

The password is supplied as the Internet e-mail address of the end user accessing the resource.

If the URL supplies a user name but no password, and the remote server requests a password, the program interpreting the FTP URL should request one from the user.

3.2.2. FTP url-path

The url-path of a FTP URL has the following syntax:

```
<cwd1>/<cwd2>/.../<cwdN>/<name>;type=<typecode>
```

Where <cwd1> through <cwdN> and <name> are (possibly encoded) strings and <typecode> is one of the characters "a", "i", or "d". The part ";type=<typecode>" may be omitted. The <cwdx> and <name> parts may be empty. The whole url-path may be omitted, including the "/" delimiting it from the prefix containing user, password, host, and port.

The url-path is interpreted as a series of FTP commands as follows:

Each of the <cwd> elements is to be supplied, sequentially, as the argument to a CWD (change working directory) command.

If the typecode is "d", perform a NLST (name list) command with <name> as the argument, and interpret the results as a file directory listing.

Otherwise, perform a TYPE command with <typecode> as the argument, and then access the file whose name is <name> (for example, using the RETR command.)

Within a name or CWD component, the characters "/" and ";" are reserved and must be encoded. The components are decoded prior to their use in the FTP protocol. In particular, if the appropriate FTP sequence to access a particular file requires supplying a string containing a "/" as an argument to a CWD or RETR command, it is necessary to encode each "/".

For example, the URL <URL:ftp://myname@host.dom/%2Fetc/motd> is interpreted by FTP-ing to "host.dom", logging in as "myname" (prompting for a password if it is asked for), and then executing "CWD /etc" and then "RETR motd". This has a different meaning from <URL:ftp://myname@host.dom/etc/motd> which would "CWD etc" and then "RETR motd"; the initial "CWD" might be executed relative to the default directory for "myname". On the other hand, <URL:ftp://myname@host.dom//etc/motd>, would "CWD " with a null argument, then "CWD etc", and then "RETR motd".

FTP URLs may also be used for other operations; for example, it is possible to update a file on a remote file server, or infer information about it from the directory listings. The mechanism for doing so is not spelled out here.

3.2.3. FTP Typecode is Optional

The entire ;type=<typecode> part of a FTP URL is optional. If it is omitted, the client program interpreting the URL must guess the appropriate mode to use. In general, the data content type of a file can only be guessed from the name, e.g., from the suffix of the name; the appropriate type code to be used for transfer of the file can then be deduced from the data content of the file.

3.2.4 Hierarchy

For some file systems, the "/" used to denote the hierarchical structure of the URL corresponds to the delimiter used to construct a file name hierarchy, and thus, the filename will look similar to the URL path. This does NOT mean that the URL is a Unix filename.

3.2.5. Optimization

Clients accessing resources via FTP may employ additional heuristics to optimize the interaction. For some FTP servers, for example, it may be reasonable to keep the control connection open while accessing multiple URLs from the same server. However, there is no common hierarchical model to the FTP protocol, so if a directory change command has been given, it is impossible in general to deduce what sequence should be given to navigate to another directory for a second retrieval, if the paths are different. The only reliable algorithm is to disconnect and reestablish the control connection.

3.3. HTTP

The HTTP URL scheme is used to designate Internet resources accessible using HTTP (HyperText Transfer Protocol).

The HTTP protocol is specified elsewhere. This specification only describes the syntax of HTTP URLs.

An HTTP URL takes the form:

```
http://<host>:<port>/<path>?<searchpart>
```

where <host> and <port> are as described in Section 3.1. If :<port> is omitted, the port defaults to 80. No user name or password is allowed. <path> is an HTTP selector, and <searchpart> is a query string. The <path> is optional, as is the <searchpart> and its preceding "?". If neither <path> nor <searchpart> is present, the "/" may also be omitted.

Within the <path> and <searchpart> components, "/", ";", "?" are reserved. The "/" character may be used within HTTP to designate a hierarchical structure.

3.4. GOPHER

The Gopher URL scheme is used to designate Internet resources accessible using the Gopher protocol.

The base Gopher protocol is described in RFC 1436 and supports items and collections of items (directories). The Gopher+ protocol is a set of upward compatible extensions to the base Gopher protocol and is described in [2]. Gopher+ supports associating arbitrary sets of attributes and alternate data representations with Gopher items. Gopher URLs accommodate both Gopher and Gopher+ items and item attributes.

3.4.1. Gopher URL syntax

A Gopher URL takes the form:

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

where <gopher-path> is one of

```
<gophertype><selector>
<gophertype><selector>%09<search>
<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. In the Gopher protocol, Gopher selector strings are a sequence of octets which may contain any octets except 09 hexadecimal (US-ASCII HT or tab) 0A hexadecimal (US-ASCII character LF), and 0D (US-ASCII character CR).

Gopher clients specify which item to retrieve by sending the Gopher selector string to a Gopher server.

Within the <gopher-path>, no characters are reserved.

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.

3.4.2 Specifying URLs for Gopher Search Engines

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.

3.4.3 URL syntax for Gopher+ items

URLs for Gopher+ items have a second encoded tab (%09) and a Gopher+ string. Note that in this case, the %09<search> string must be supplied, although the <search> element may be the empty string.

The <gopher+_string> is used to represent information required for retrieval of the Gopher+ item. Gopher+ items may have alternate views, arbitrary sets of attributes, and may have electronic forms associated with them.

To retrieve the data associated with a Gopher+ URL, a client will connect to the server and send the Gopher selector, followed by a tab and the search string (which may be empty), followed by a tab and the Gopher+ commands.

3.4.4 Default Gopher+ data representation

When a Gopher server returns a directory listing to a client, the Gopher+ items are tagged with either a "+" (denoting Gopher+ items) or a "?" (denoting Gopher+ items which have a +ASK form associated with them). A Gopher URL with a Gopher+ string consisting of only a "+" refers to the default view (data representation) of the item while a Gopher+ string containing only a "?" refer to an item with a Gopher electronic form associated with it.

3.4.5 Gopher+ items with electronic forms

Gopher+ items which have a +ASK associated with them (i.e. Gopher+ items tagged with a "?") require the client to fetch the item's +ASK attribute to get the form definition, and then ask the user to fill out the form and return the user's responses along with the selector string to retrieve the item. Gopher+ clients know how to do this but depend on the "?" tag in the Gopher+ item description to know when to handle this case. The "?" is used in the Gopher+ string to be consistent with Gopher+ protocol's use of this symbol.

3.4.6 Gopher+ item attribute collections

To refer to the Gopher+ attributes of an item, the Gopher URL's Gopher+ string consists of "!" or "$". "!" refers to the all of a Gopher+ item's attributes. "$" refers to all the item attributes for all items in a Gopher directory.

3.4.7 Referring to specific Gopher+ attributes

To refer to specific attributes, the URL's gopher+_string is "!<attribute_name>" or "$<attribute_name>". For example, to refer to the attribute containing the abstract of an item, the gopher+_string would be "!+ABSTRACT".

To refer to several attributes, the gopher+_string consists of the attribute names separated by coded spaces. For example, "!+ABSTRACT%20+SMELL" refers to the +ABSTRACT and +SMELL attributes of an item.

3.4.8 URL syntax for Gopher+ alternate views

Gopher+ allows for optional alternate data representations (alternate views) of items. To retrieve a Gopher+ alternate view, a Gopher+ client sends the appropriate view and language identifier (found in the item's +VIEW attribute). To refer to a specific Gopher+ alternate view, the URL's Gopher+ string would be in the form:

```
+<view_name>%20<language_name>
```

For example, a Gopher+ string of "+application/postscript%20Es_ES" refers to the Spanish language postscript alternate view of a Gopher+ item.

3.4.9 URL syntax for Gopher+ electronic forms

The gopher+_string for a URL that refers to an item referenced by a Gopher+ electronic form (an ASK block) filled out with specific values is a coded version of what the client sends to the server. The gopher+_string is of the form:

```
+%091%0D%0A+-1%0D%0A<ask_item1_value>%0D%0A<ask_item2_value>%0D%0A.%0D%0A
```

To retrieve this item, the Gopher client sends:

```
<a_gopher_selector><tab>+<tab>1<cr><lf>
+-1<cr><lf>
<ask_item1_value><cr><lf>
<ask_item2_value><cr><lf>
.<cr><lf>
```

to the Gopher server.

3.5. MAILTO

The mailto URL scheme is used to designate the Internet mailing address of an individual or service. No additional information other than an Internet mailing address is present or implied.

A mailto URL takes the form:

```
mailto:<rfc822-addr-spec>
```

where <rfc822-addr-spec> is (the encoding of an) addr-spec, as specified in RFC 822 [6]. Within mailto URLs, there are no reserved characters.

Note that the percent sign ("%") is commonly used within RFC 822 addresses and must be encoded.

Unlike many URLs, the mailto scheme does not represent a data object to be accessed directly; there is no sense in which it designates an object. It has a different use than the message/external-body type in MIME.

3.6. NEWS

The news URL scheme is used to refer to either news groups or individual articles of USENET news, as specified in RFC 1036.

A news URL takes one of two forms:

```
news:<newsgroup-name>
news:<message-id>
```

A <newsgroup-name> is a period-delimited hierarchical name, such as "comp.infosystems.www.misc". A <message-id> corresponds to the Message-ID of section 2.1.5 of RFC 1036, without the enclosing "<" and ">"; it takes the form <unique>@<full_domain_name>. A message identifier may be distinguished from a news group name by the presence of the commercial at "@" character. No additional characters are reserved within the components of a news URL.

If <newsgroup-name> is "*" (as in <URL:news:*>), it is used to refer to "all available news groups".

The news URLs are unusual in that by themselves, they do not contain sufficient information to locate a single resource, but, rather, are location-independent.

3.7. NNTP

The nntp URL scheme is an alternative method of referencing news articles, useful for specifying news articles from NNTP servers (RFC 977).

A nntp URL take the form:

```
nntp://<host>:<port>/<newsgroup-name>/<article-number>
```

where <host> and <port> are as described in Section 3.1. If :<port> is omitted, the port defaults to 119.

The <newsgroup-name> is the name of the group, while the <article-number> is the numeric id of the article within that newsgroup.

Note that while nntp: URLs specify a unique location for the article resource, most NNTP servers currently on the Internet today are configured only to allow access from local clients, and thus nntp URLs do not designate globally accessible resources. Thus, the news: form of URL is preferred as a way of identifying news articles.

3.8. TELNET

The Telnet URL scheme is used to designate interactive services that may be accessed by the Telnet protocol.

A telnet URL takes the form:

```
telnet://<user>:<password>@<host>:<port>/
```

as specified in Section 3.1. The final "/" character may be omitted. If :<port> is omitted, the port defaults to 23. The :<password> can be omitted, as well as the whole <user>:<password> part.

This URL does not designate a data object, but rather an interactive service. Remote interactive services vary widely in the means by which they allow remote logins; in practice, the <user> and <password> supplied are advisory only: clients accessing a telnet URL merely advise the user of the suggested username and password.

3.9. WAIS

The WAIS URL scheme is used to designate WAIS databases, searches, or individual documents available from a WAIS database. WAIS is described in [7]. The WAIS protocol is described in RFC 1625 [17]; Although the WAIS protocol is based on Z39.50-1988, the WAIS URL scheme is not intended for use with arbitrary Z39.50 services.

A WAIS URL takes one of the following forms:

```
wais://<host>:<port>/<database>
wais://<host>:<port>/<database>?<search>
wais://<host>:<port>/<database>/<wtype>/<wpath>
```

where <host> and <port> are as described in Section 3.1. If :<port> is omitted, the port defaults to 210. The first form designates a WAIS database that is available for searching. The second form designates a particular search. <database> is the name of the WAIS database being queried.

The third form designates a particular document within a WAIS database to be retrieved. In this form <wtype> is the WAIS designation of the type of the object. Many WAIS implementations require that a client know the "type" of an object prior to retrieval, the type being returned along with the internal object identifier in the search response. The <wtype> is included in the URL in order to allow the client interpreting the URL adequate information to actually retrieve the document.

The <wpath> of a WAIS URL consists of the WAIS document-id, encoded as necessary using the method described in Section 2.2. The WAIS document-id should be treated opaquely; it may only be decomposed by the server that issued it.

3.10 FILES

The file URL scheme is used to designate files accessible on a particular host computer. This scheme, unlike most other URL schemes, does not designate a resource that is universally accessible over the Internet.

A file URL takes the form:

```
file://<host>/<path>
```

where <host> is the fully qualified domain name of the system on which the <path> is accessible, and <path> is a hierarchical directory path of the form <directory>/<directory>/.../<name>.

For example, a VMS file

```
DISK$USER:[MY.NOTES]NOTE123456.TXT
```

might become

```
<URL:file://vms.host.edu/disk$user/my/notes/note12345.txt>
```

As a special case, <host> can be the string "localhost" or the empty string; this is interpreted as 'the machine from which the URL is being interpreted'.

The file URL scheme is unusual in that it does not specify an Internet protocol or access method for such files; as such, its utility in network protocols between hosts is limited.

3.11 PROSPERO

The Prospero URL scheme is used to designate resources that are accessed via the Prospero Directory Service. The Prospero protocol is described elsewhere [14].

A prospero URLs takes the form:

```
prospero://<host>:<port>/<hsoname>;<field>=<value>
```

where <host> and <port> are as described in Section 3.1. If :<port> is omitted, the port defaults to 1525. No username or password is allowed.

The <hsoname> is the host-specific object name in the Prospero protocol, suitably encoded. This name is opaque and interpreted by the Prospero server. The semicolon ";" is reserved and may not appear without quoting in the <hsoname>.

Prospero URLs are interpreted by contacting a Prospero directory server on the specified host and port to determine appropriate access methods for a resource, which might themselves be represented as different URLs. External Prospero links are represented as URLs of the underlying access method and are not represented as Prospero URLs.

Note that a slash "/" may appear in the <hsoname> without quoting and no significance may be assumed by the application. Though slashes may indicate hierarchical structure on the server, such structure is not guaranteed. Note that many <hsoname>s begin with a slash, in which case the host or port will be followed by a double slash: the slash from the URL syntax, followed by the initial slash from the <hsoname>. (E.g., <URL:prospero://host.dom//pros/name> designates a <hsoname> of "/pros/name".)

In addition, after the <hsoname>, optional fields and values associated with a Prospero link may be specified as part of the URL. When present, each field/value pair is separated from each other and from the rest of the URL by a ";" (semicolon). The name of the field and its value are separated by a "=" (equal sign). If present, these fields serve to identify the target of the URL. For example, the OBJECT-VERSION field can be specified to identify a specific version of an object.

4. REGISTRATION OF NEW SCHEMES

A new scheme may be introduced by defining a mapping onto a conforming URL syntax, using a new prefix. URLs for experimental schemes may be used by mutual agreement between parties. Scheme names starting with the characters "x-" are reserved for experimental purposes.

The Internet Assigned Numbers Authority (IANA) will maintain a registry of URL schemes. Any submission of a new URL scheme must include a definition of an algorithm for accessing of resources within that scheme and the syntax for representing such a scheme.

URL schemes must have demonstrable utility and operability. One way to provide such a demonstration is via a gateway which provides objects in the new scheme for clients using an existing protocol. If the new scheme does not locate resources that are data objects, the properties of names in the new space must be clearly defined.

New schemes should try to follow the same syntactic conventions of existing schemes, where appropriate. It is likewise recommended that, where a protocol allows for retrieval by URL, that the client software have provision for being configured to use specific gateway locators for indirect access through new naming schemes.

The following scheme have been proposed at various times, but this document does not define their syntax or use at this time. It is suggested that IANA reserve their scheme names for future definition:

```
afs              Andrew File System global file names.
mid              Message identifiers for electronic mail.
cid              Content identifiers for MIME body parts.
nfs              Network File System (NFS) file names.
tn3270           Interactive 3270 emulation sessions.
mailserver       Access to data available from mail servers.
z39.50           Access to ANSI Z39.50 services.
```

5. BNF for specific URL schemes

This is a BNF-like description of the Uniform Resource Locator syntax, using the conventions of RFC822, except that "|" is used to designate alternatives, and brackets [] are used around optional or repeated elements. Briefly, literals are quoted with "", optional elements are enclosed in [brackets], and elements may be preceded with <n>* to designate n or more repetitions of the following element; n defaults to 0.

```
; The generic form of a URL is:

genericurl     = scheme ":" schemepart

; Specific predefined schemes are defined here; new schemes
; may be registered with IANA

url            = httpurl | ftpurl | newsurl |
                 nntpurl | telneturl | gopherurl |
                 waisurl | mailtourl | fileurl |
                 prosperourl | otherurl

; new schemes follow the general syntax
otherurl       = genericurl

; the scheme is in lower case; interpreters should use case-ignore
scheme         = 1*[ lowalpha | digit | "+" | "-" | "." ]
```

```
schemepart     = *xchar | ip-schemepart


; URL schemeparts for ip based protocols:

ip-schemepart  = "//" login [ "/" urlpath ]

login          = [ user [ ":" password ] "@" ] hostport
hostport       = host [ ":" port ]
host           = hostname | hostnumber
hostname       = *[ domainlabel "." ] toplabel
domainlabel    = alphadigit | alphadigit *[ alphadigit | "-" ] alphadigit
toplabel       = alpha | alpha *[ alphadigit | "-" ] alphadigit
alphadigit     = alpha | digit
hostnumber     = digits "." digits "." digits "." digits
port           = digits
user           = *[ uchar | ";" | "?" | "&" | "=" ]
password       = *[ uchar | ";" | "?" | "&" | "=" ]
urlpath        = *xchar    ; depends on protocol see section 3.1

; The predefined schemes:

; FTP (see also RFC959)

ftpurl         = "ftp://" login [ "/" fpath [ ";type=" ftptype ]]
fpath          = fsegment *[ "/" fsegment ]
fsegment       = *[ uchar | "?" | ":" | "@" | "&" | "=" ]
ftptype        = "A" | "I" | "D" | "a" | "i" | "d"

; FILE

fileurl        = "file://" [ host | "localhost" ] "/" fpath

; HTTP

httpurl        = "http://" hostport [ "/" hpath [ "?" search ]]
hpath          = hsegment *[ "/" hsegment ]
hsegment       = *[ uchar | ";" | ":" | "@" | "&" | "=" ]
search         = *[ uchar | ";" | ":" | "@" | "&" | "=" ]

; GOPHER (see also RFC1436)

gopherurl      = "gopher://" hostport [ / [ gtype [ selector
                 [ "%09" search [ "%09" gopher+_string ] ] ] ] ]
gtype          = xchar
selector       = *xchar
gopher+_string = *xchar
```

```
; MAILTO (see also RFC822)

mailtourl      = "mailto:" encoded822addr
encoded822addr = 1*xchar               ; further defined in RFC822

; NEWS (see also RFC1036)

newsurl        = "news:" grouppart
grouppart      = "*" | group | article
group          = alpha *[ alpha | digit | "-" | "." | "+" | "_" ]
article        = 1*[ uchar | ";" | "/" | "?" | ":" | "&" | "=" ] "@" host

; NNTP (see also RFC977)

nntpurl        = "nntp://" hostport "/" group [ "/" digits ]

; TELNET

telneturl      = "telnet://" login [ "/" ]

; WAIS (see also RFC1625)

waisurl        = waisdatabase | waisindex | waisdoc
waisdatabase   = "wais://" hostport "/" database
waisindex      = "wais://" hostport "/" database "?" search
waisdoc        = "wais://" hostport "/" database "/" wtype "/" wpath
database       = *uchar
wtype          = *uchar
wpath          = *uchar

; PROSPERO

prosperourl    = "prospero://" hostport "/" ppath *[ fieldspec ]
ppath          = psegment *[ "/" psegment ]
psegment       = *[ uchar | "?" | ":" | "@" | "&" | "=" ]
fieldspec      = ";" fieldname "=" fieldvalue
fieldname      = *[ uchar | "?" | ":" | "@" | "&" ]
fieldvalue     = *[ uchar | "?" | ":" | "@" | "&" ]

; Miscellaneous definitions

lowalpha       = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                 "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                 "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                 "y" | "z"
hialpha        = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
                 "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
                 "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
```

```
alpha          = lowalpha | hialpha
digit          = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                 "8" | "9"
safe           = "$" | "-" | "_" | "." | "+"
extra          = "!" | "*" | "'" | "(" | ")" | ","
national       = "{" | "}" | "|" | "\" | "^" | "~" | "[" | "]" | "`"
punctuation    = "<" | ">" | "#" | "%" | <">


reserved       = ";" | "/" | "?" | ":" | "@" | "&" | "="
hex            = digit | "A" | "B" | "C" | "D" | "E" | "F" |
                 "a" | "b" | "c" | "d" | "e" | "f"
escape         = "%" hex hex

unreserved     = alpha | digit | safe | extra
uchar          = unreserved | escape
xchar          = unreserved | reserved | escape
digits         = 1*digit
```

6. Security Considerations

The URL scheme does not in itself pose a security threat. Users should beware that there is no general guarantee that a URL which at one time points to a given object continues to do so, and does not even at some later time point to a different object due to the movement of objects on servers.

A URL-related security threat is that it is sometimes possible to construct a URL such that an attempt to perform a harmless idempotent operation such as the retrieval of the object will in fact cause a possibly damaging remote operation to occur. The unsafe URL is typically constructed by specifying a port number other than that reserved for the network protocol in question. The client unwittingly contacts a server which is in fact running a different protocol. The content of the URL contains instructions which when interpreted according to this other protocol cause an unexpected operation. An example has been the use of gopher URLs to cause a rude message to be sent via a SMTP server. Caution should be used when using any URL which specifies a port number other than the default for the protocol, especially when it is a number within the reserved space.

Care should be taken when URLs contain embedded encoded delimiters for a given protocol (for example, CR and LF characters for telnet protocols) that these are not unencoded before transmission. This would violate the protocol but could be used to simulate an extra operation or parameter, again causing an unexpected and possible harmful remote operation to be performed.

The use of URLs containing passwords that should be secret is clearly unwise.

7. Acknowledgements

This paper builds on the basic WWW design (RFC 1630) and much discussion of these issues by many people on the network. The discussion was particularly stimulated by articles by Clifford Lynch, Brewster Kahle [10] and Wengyik Yeong [18]. Contributions from John Curran, Clifford Neuman, Ed Vielmetti and later the IETF URL BOF and URI working group were incorporated.

Most recently, careful readings and comments by Dan Connolly, Ned Freed, Roy Fielding, Guido van Rossum, Michael Dolan, Bert Bos, John Kunze, Olle Jarnefors, Peter Svanberg and many others have helped refine this RFC.

APPENDIX: Recommendations for URLs in Context

URIs, including URLs, are intended to be transmitted through protocols which provide a context for their interpretation.

In some cases, it will be necessary to distinguish URLs from other possible data structures in a syntactic structure. In this case, is recommended that URLs be preceeded with a prefix consisting of the characters "URL:". For example, this prefix may be used to distinguish URLs from other kinds of URIs.

In addition, there are many occasions when URLs are included in other kinds of text; examples include electronic mail, USENET news messages, or printed on paper. In such cases, it is convenient to have a separate syntactic wrapper that delimits the URL and separates it from the rest of the text, and in particular from punctuation marks that might be mistaken for part of the URL. For this purpose, is recommended that angle brackets ("<" and ">"), along with the prefix "URL:", be used to delimit the boundaries of the URL. This wrapper does not form part of the URL and should not be used in contexts in which delimiters are already specified.

In the case where a fragment/anchor identifier is associated with a URL (following a "#"), the identifier would be placed within the brackets as well.

In some cases, extra whitespace (spaces, linebreaks, tabs, etc.) may need to be added to break long URLs across lines. The whitespace should be ignored when extracting the URL.

No whitespace should be introduced after a hyphen ("-") character. Because some typesetters and printers may (erroneously) introduce a hyphen at the end of line when breaking a line, the interpreter of a URL containing a line break immediately after a hyphen should ignore all unencoded whitespace around the line break, and should be aware that the hyphen may or may not actually be part of the URL.

Examples:

```
Yes, Jim, I found it under <URL:ftp://info.cern.ch/pub/www/doc;
type=d> but you can probably pick it up from <URL:ftp://ds.in
ternic.net/rfc>.  Note the warning in <URL:http://ds.internic.
net/instructions/overview.html#WARNING>.
```

References

[1] 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, University of Minnesota, March 1993. <URL:ftp://ds.internic.net/rfc/rfc1436.txt;type=a>

[2] Anklesaria, F., Lindner, P., McCahill, M., Torrey, D., Johnson, D., and B. Alberti, "Gopher+: Upward compatible enhancements to the Internet Gopher protocol", University of Minnesota, July 1993. <URL:ftp://boombox.micro.umn.edu/pub/gopher/gopher_protocol /Gopher+/Gopher+.txt>

[3] Berners-Lee, T., "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", RFC 1630, CERN, June 1994. <URL:ftp://ds.internic.net/rfc/rfc1630.txt>

[4] Berners-Lee, T., "Hypertext Transfer Protocol (HTTP)", CERN, November 1993. <URL:ftp://info.cern.ch/pub/www/doc/http-spec.txt.Z>

[5] Braden, R., Editor, "Requirements for Internet Hosts -- Application and Support", STD 3, RFC 1123, IETF, October 1989. <URL:ftp://ds.internic.net/rfc/rfc1123.txt>

[6] Crocker, D. "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, UDEL, April 1982. <URL:ftp://ds.internic.net/rfc/rfc822.txt>

[7] Davis, F., Kahle, B., Morris, H., Salem, J., Shen, T., Wang, R., Sui, J., and M. Grinbaum, "WAIS Interface Protocol Prototype Functional Specification", (v1.5), Thinking Machines Corporation, April 1990. <URL:ftp://quake.think.com/pub/wais/doc/protspec.txt>

[8] Horton, M. and R. Adams, "Standard For Interchange of USENET Messages", RFC 1036, AT&T Bell Laboratories, Center for Seismic Studies, December 1987. <URL:ftp://ds.internic.net/rfc/rfc1036.txt>

[9] Huitema, C., "Naming: Strategies and Techniques", Computer Networks and ISDN Systems 23 (1991) 107-110.

[10] Kahle, B., "Document Identifiers, or International Standard Book Numbers for the Electronic Age", 1991. <URL:ftp://quake.think.com/pub/wais/doc/doc-ids.txt>

[11] Kantor, B. and P. Lapsley, "Network News Transfer Protocol: A Proposed Standard for the Stream-Based Transmission of News", RFC 977, UC San Diego & UC Berkeley, February 1986. <URL:ftp://ds.internic.net/rfc/rfc977.txt>

[12] Kunze, J., "Functional Requirements for Internet Resource Locators", Work in Progress, December 1994. <URL:ftp://ds.internic.net/internet-drafts /draft-ietf-uri-irl-fun-req-02.txt>

[13] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, USC/Information Sciences Institute, November 1987. <URL:ftp://ds.internic.net/rfc/rfc1034.txt>

[14] Neuman, B., and S. Augart, "The Prospero Protocol", USC/Information Sciences Institute, June 1993. <URL:ftp://prospero.isi.edu/pub/prospero/doc /prospero-protocol.PS.Z>

[15] Postel, J. and J. Reynolds, "File Transfer Protocol (FTP)", STD 9, RFC 959, USC/Information Sciences Institute, October 1985. <URL:ftp://ds.internic.net/rfc/rfc959.txt>

[16] Sollins, K. and L. Masinter, "Functional Requirements for Uniform Resource Names", RFC 1737, MIT/LCS, Xerox Corporation, December 1994. <URL:ftp://ds.internic.net/rfc/rfc1737.txt>

[17] St. Pierre, M, Fullton, J., Gamiel, K., Goldman, J., Kahle, B., Kunze, J., Morris, H., and F. Schiettecatte, "WAIS over Z39.50-1988", RFC 1625, WAIS, Inc., CNIDR, Thinking Machines Corp., UC Berkeley, FS Consulting, June 1994. <URL:ftp://ds.internic.net/rfc/rfc1625.txt>

[18] Yeong, W. "Towards Networked Information Retrieval", Technical report 91-06-25-01, Performance Systems International, Inc. <URL:ftp://uu.psi.com/wp/nir.txt>, June 1991.

[19] Yeong, W., "Representing Public Archives in the Directory", Work in Progress, November 1991.

[20] "Coded Character Set -- 7-bit American Standard Code for Information Interchange", ANSI X3.4-1986.

Editors' Addresses

Tim Berners-Lee
World-Wide Web project
CERN,
1211 Geneva 23,
Switzerland

Phone: +41 (22)767 3755
Fax: +41 (22)767 7155
EMail: timbl@info.cern.ch

Larry Masinter
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94034

Phone: (415) 812-4365
Fax: (415) 812-4333
EMail: masinter@parc.xerox.com

Mark McCahill
Computer and Information Services,
University of Minnesota
Room 152 Shepherd Labs
100 Union Street SE
Minneapolis, MN 55455

Phone: (612) 625 1300
EMail: mpm@boombox.micro.umn.edu