Uniform Resource Locators | Tim Berners-Lee
Internet Draft | CERN
Expires 14 May 1994 | 14 October 1993

Uniform Resource Locators

A unifying syntax for the expression of names and addresses of objects on the network

Status of this memo

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

Internet Drafts are working documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress".

Distribution of this document is unlimited. Please send comments to the author as timbl@info.cern.ch. or to the discussion list ietf-url@merit.edu.

Abstract

Many protocols and systems for document search and retrieval are currently in use, and many more protocols or refinements of existing protocols are to be expected in a field whose expansion is explosive.

These systems are aiming to achieve global search and readership of documents across differing computing platforms, and despite a plethora of protocols and data formats. As protocols evolve, gateways can allow global access to remain possible. As data formats evolve, format conversion programs can preserve global access. There is one area, however, in which it is impractical to make conversions, and that is in the names and addresses used to identify objects. This is because names and addresses of objects are passed on in so many ways, from the backs of envelopes to hypertext objects, and may have a long life.

This paper discusses the requirements on a universal syntax which can be used to refer to objects available using existing protocols, and may be extended with technology. It makes a recommendation for a generic syntax, and for specific forms for "Uniform Resource Locators" (URLs)of objects accessible using existing Internet protocols.

The syntax has been in widespread use by World-Wide Web software since 1990.

Terms

The objects on the network which are to be named and addressed include typically objects which can be retrieved, and objects which can be searched. There is a great variety of other objects which may support other operations. We imply nothing about the contents of objects in this document. Whereas human-readable documents are currently the center of interest of the field, we envisage all aspects discussed in this paper applying to generalised objects when systems to handle them become available. The "object" is the unit of reference and need not correspond to any unit of storage. We refer to objects which can be searched as "indexes". We emphasise that this is the abstract view of the client, and these objects need not correspond to physical files on computers. We refer to the person who does the retrieval or searching as the user.

Within this document, we use the terms "name" very generally for a string of characters describing an object, whatever its combination of properties mentioned below. (The term usually has a narrower meaning but we needed some term for the universal set.). This uniform syntax applied to a generic name is known as a Uiform Resource Identifier (URI). The term "address" is reserved for an string which specifies a more or less physical location. The term "locator" refers to a URL as here defined. URIs which have a greater persistence than URLs are referred to as URNs.

Requirements

This section discusses requirements for URLs, as an introduction of and background for the Recommendations section.

Uses of names and addresses

A name allows a user, with the help of a "client" program, to retrieve or operate on objects via a "server" program. A name may be passed for example:

- In communication of any form between two people, to refer to a document, or part of a document;
- As part of the description of a link associated with a hypertext document;
- As part of the result of searching an index.

Some typical requirements on a name which are met to a varying degree by various schemes are for example that the name is

Persistent A given name will remain valid as long as it is needed;

Extensible A given naming syntax will remain valid through the introduction of new protocols and directory technologies;

Resolvable A name will contain enough information to allow the document or index to which it refers to be accessed, perhaps via resolution into an intermediate, more physical, name.

Unique Each object can only have one such name. The fact that two such names are different implies that the objects to which they refer are different (in some way).

Unambiguous The fact that two names are identical implies that the objects named are the same (in some way).

The syntax discussed is the syntax of one name, be it a lasting name or a physical address. When a directory server or hypertext link contains a set of alternative names, then that is beyond the scope of this syntax. Similarly, a syntax for describing a compound object is outside the scope of this syntax. The specific locator name spaces (defined under the umbrella of the general syntax) each meet the requirements above to a greater or lesser extent.

Current practice

Current protocols use many different standards for names. For some protocols, such as ISO-10163 Search and Retrieve protocol[16], the names returned in a search are only valid during the session. For others, such as FTP[9], they are lasting names which may be used for object retrieval at a later time. Typically, however, they are not long-lasting names which are independent of the location of the object. Such names may be provided using directory servers such as x.500. They will refer to the registration, however formal or informal, of a object with a particular organisation or person. Both hypertext and manual references rely on long- lasting names. Current names are basically location specifiers (addresses). These may be known as Uniform Resource Locators (URLs). They give the necessary parts of an address for a reader to access an information provider using the given protocol, and ask for the object required. Examples of names used by various protocols include

File Transfer Protocol (Postel 1985):

- Host name or IP-address
- [TCP port]
- [user name, password]
- Filename

W.A.I.S. (Kahle 1990)

- Host name or IP-address
- [TCP port]
- local document id

Gopher (Alberti 1991)

- Host name or IP-address
- [TCP port]
- database name
- selector string

HTTP (Berners-Lee 1991)

Host name or IP-address

[TCP port]

local object id

NNTP (Kantor 1986) group Group name

NNTP article

- Host name
- unique message identifier

Prospero links (Neuman 1992)

Host name or IP address

- [UDP port]
- Host specific object name
- [version]
- [identifier]*

x.500 distinguished name

- Country
- Organisation
- Organisational unit
- Person
- Local object identifier

Other systems with their own naming schemes include BITNET "LISTSERV" application, FTAM file retrieval, SQLnetTM remote database search, proprietary distributed file systems, etc. Conventional syntax for writing these addresses involve various forms of punctuation to separate these parts. This sometimes, but not always, allows the naming scheme to be deduced from the punctuation. For example, a name of the form xxx.yyy.zz.edu:/pub.aa.bb.cc often implies anonymous FTP access. However, there is no well-defined algorithm for parsing an arbitrary name, as there is no common syntax.

Expandability

There will necessarily be a phase during which lasting names will become more common, as the deployment of directory services increases to the point where every user has direct or indirect access to one. Even then, however, one can envisage more than one competing directory system, and cases in which physical names are still required. A directory service takes a lasting name and reduces it to a physical address (or set of addresses) which, though less useful for lasting reference, is the only way to actually retrieve the object. An addressing syntax is required which will be able to encompass existing physical address spaces, and be extendible to any future protocols. This requires that it contain an identifier for the protocol in use. The format of the rest of the address will necessarily depend to a certain extent on the protocol.

Relevance

The life of a name is limited by any information contained within it which may become prematurely invalid. It is therefore necessary to limit the contents of a name to the information required for the operations above. Other extraneous information about the object (its size, data format, authorisation details, etc.) may in general change with time and should not be part of the name. One might expect such information to be part of the "header" of a object, and for protocols to allow the header information to be retrieved independently of the objects themselves. Any physical address may be subject to change with time: hence we encourage the move to lasting names and directory services.

Uniqueness

Clearly one requires unambiguous names in the sense that one name should refer to only one logical object. This is the case with all the addressing schemes in use, whether they are directory systems or physical addresses. (The internet addresses all rely on the domain name (Mockapetris 1987) of the host to achieve this). However, given that names can be translated, many apparently different names may lead to the same object. Any object may therefore be referred to by many names. One needs to be able to know whether two objects, retrieved through different paths, are in fact the same object. It is suggested that each object have a unique "official" name. This name could be stored in the object in some representations, or stored in a database accessible to the server, for example. Any references within that object should be parsed in the context of the official name. In the presence of a directory service, the official name will normally be the registered name of the object. However, a name in any scheme will do, so long as it is completely specified. On systems which do not allow the name to be stored (such as anonymous FTP archive sites), a possible ambiguity will always exist as to whether two similarly named objects are in fact the same. Note that Internet newsgroup names are unique world-wide, and news articles carry a unique message id. In most other cases, however, there is no guarantee that dereferencing a URL will work, or that if it does the object it refers to will in fact be the object intended. URLs such as FTP addresses are transient in that files may be moved and even replaced by different files of the same name. This disorganisation may be limited by good server management, but a naming scheme which is independent also of internet host name is obviously preferable.

Readability by people

This requirement has been put forward by several people (Clifford Lynch, Douglas Engelbart among others), and disputed by others. The author's view is that it will be a while before technology and standardisation have reached the point at which names and addresses will be hidden from human beings. As long as they must be written on the backs of envelopes and "cut and pasted" between workstation windows, there is a strong need for names to be

- Short
- Composed of printable (preferably non-white) characters
- To a certain extent, understadable by a human being.

Structure of names and addresses

A physical address is required in order for:

- The user's program to contact the server;
- The server to perform the operation (e.g. search and index, retrieve a object, or look up the name) and return a result;
- The user's program to locate an individual position or element within a returned object.

This suggests that a name be structured, such that the parts necessary for these three operations be separate and only used by those system elements which need those parts. This corresponds to the basic principle of information hiding. In fact, four parts are necessary, including the indicator of the naming scheme to be used:

- The naming scheme: a registered identifier for the protocol.
- The name of a suitable server. The format of this part must be well defined. It will depend on the lower-layer protocols in use. Systems which use widely distributed information, such as x.500 and NNTP, do not need this part as each client generally contacts his nearest server (or a particular server).
- Information to be passed to the server. This may be private to the server, as all names may be generated and used by the same server. This part of the name should be opaque to the client.
- Information to be used by the application once the object has been retrieved. This part is private to the application (or, more strictly, the data format) and so cannot be defined here.

Both lasting names and physical addresses often share a hierarchical structure. This follows often from the organisation of the system. From the naming point of view, it has the advantage that a reference in one object to another object need not include that part of the structure which is common to both names.

Choices

The requirements above leave little room for choice save for the order and punctuation of the elements of an address. It is only reasonable for the order of writing of the parts to be consistently from left to right (or right to left) with increasing specificity. Punctuation schemes fall into two categories (Huitema 1991): tagged schemes in which field are given names, and fields which use special characters and field order. The latter tend to be more compact schemes.

```
protocol: aftp host: xxx.yyy.edu path:

        /pub/doc/README

        PR=aftp; H=xx.yy.edu; PA=/pub/doc/README;

        PR:aftp/xx.yy.edu/pub/doc/README

        /aftp/xx.yy.edu/pub/doc/README
```

Fig 1. Some alternative tagged and untagged representations

The choice of special symbols for punctuation tends to be a matter of taste. It is easier to read addresses whose symbols correspond to those of one's favourite operating system. A variety of symbols is needed so that when a name is abbreviated it is possible to tell which parts have been omitted.

The recommendation below uses special characters in order to achieve a compact name, and uses where possible punctuation symbols established in the internet or unix community.

The choice of escape character for introducing representations of non-allowed characters also tends to be a matter of taste. An ANSI standard exists in the C language, using the back-slash character "\". The use of this character on unix command lines, however, can be a problem as it is interpreted by many shell programs, and would have itself to be escaped.

There is a conflict between the need to be able to represent many characters including spaces within a URL directly, and the need to be able to use a URL in environments which have limited character sets or in which certain characters are prone to corruption. This conflict has been resolved by use of an hexadecimal escaping method which may be applied to any characters forbidden in a given context. When URLs are moved between contexts, the set of characters escaped may be enlarged or reduced unambiguously.

The use of multiple white space characters is discouraged in URLs to be printed or sent by electronic mail. This is because of the frequent introduction of extraneous white space when lines are wrapped by systems such as mail, or sheer necessity of narrow column width, and because of the inter-conversion of various forms of white space which occurs during character code conversion and the transfer of text between applications.

Recommendations

This section describes the syntax for "Uniform Resource Locators" (URLs): that is, basically physical addresses of objects which are retrievable using protocols already deployed on the net. The generic syntax provides a framework for new schemes for names to be resolved using as yet undefined protocols.

The syntax is described in two parts. Firstly, we give the syntax rules of a completely specified name; secondly, we give the rules under which parts of the name may be omitted in a well-defined context.

Full form

A complete URL consists of a naming scheme specifier followed by a string whose format is a function of the naming scheme. For locators of information on the internet, a common syntax is used for the IP address part. A BNF description of the URL syntax is given in an a later section. The components are as follows.

Fragment-id

This represents a part of, fragment of, or a sub-function within, an object or object. Its syntax and semantics are defined by the application responsible for the object, or the specification of the content type of the object. The only definition here is of the allowed characters by which it may be represented in a URL.

The fragment-id follows the URL of the whole object from which it is separated by a hash sign (#). If the fragment-id is void, the hash sign may be omitted: A void fragment-id with or without the hash sign means that the URL refers to the whole object.

While this hook is allowed for identification of fragments, the question of addressing of parts of objects, or of the grouping of objects and relationship between contined and containing objects, is not addressed by this object.

This object does not address the question of objects which are different versions of a "living" object, nor of expressing the relationships between different versions and the living object.

Scheme

Within the URL of a object, the first element is the name of the scheme, separated from the rest of the object by a colon. The rest of the URL follows the colon in a format depending on the scheme.

Internet protocol parts

Those schemes which refer to internet protocols have a common syntax for the rest of the object name. This starts with a double slash "//" to indicate its presence, and continues until the following slash "/". Within that section are

An optional user name, if this must be quoted to the server, followed by a commercial at sign "@". (Use of this field is discouraged. Provision of encoding a password after the user name, delimited by a colon, could be made but obviously is only useful when the password is public, in which case it should not be necessary, so that is also discouraged.)

The internet domain name of the host in RFC1037 format (or, optionally and less advisably, the IP address as a set of four decimal digits)

The port number, if it is not the default number for the protocol, is given in decimal notation after a colon.

Path The rest of the locator is known as the "path". It may define details of how the client should communicate with the server, including information to be passed transparently to the server without any processing by the client.

The path is interpreted in a manner dependent on the protocol being used. However, when it contains slashes, these must imply a hierarchical structure.

Partial form

In a certain limited set of cases, generally within a certain application, it may be useful to pass only a section of the URL. Within a object whose URL is well defined, the URL of another object may be given in abbreviated form, where parts of the two URLs are the same. This allows objects within a group to refer to each other without requiring the space for a complete reference, and it incidentally allows the group of objects to be moved without changing any references. This is not discussed in detail here, it is only mentioned so that the characters required by the technique be reserved for that purpose. It must be emphasised that when a reference is passed in anything other than a well controlled context, the full form must always be used.

The partial form relies on a property of the URL syntax that certain characters ("/") and certain path elements ("..", ".") have a significance reserved for representing a hierarchical space, and must be recognised as such by both clients and servers.

A partial form can be distinguished from a full form in that a full form must have a colon and that colon must occur before any slash characters.

The rules for the use of a partial name are:

- If the scheme parts are different, the whole absolute locator must be given. Otherwise, the scheme is omitted, and:
- If the host and/or port parts are the different, the host, port name and all the rest of the locator must be given.
- If the access and host parts are the same, then the path may be given in absolute (fully qualified) or relative form. Within the path:
- If a leading slash is present, the path is absolute. Otherwise, a relative path is interpreted as follows:
- The last part of the path of the context locator (anything following the rightmost slash) is removed, and the given partial URL appended in its place.
- Within the result, all occurrences of "/xxx/.." or "/." are recursively removed, where xxx, ".." and "." are complete path elements.

Note: If a path of the context locator end in slash, partial URLs will be treated differently to their treatment with respect to the same path without a slash. Using a trailing slash on a directory name is not therefore recommended. The signifcance of a trailing slash may be considered as that of the locator of a file with void name within that directory.

Encoding prohibited characters

When a system uses a local addressing scheme, it is useful to provide a mapping from local addresses into URLs so that references to objects within the addressing scheme may be referred to globally, and possibly accessed through gateway servers.

Any mapping scheme may be defined provided it is unambiguous, reversible, and provides valid URLs. It is recommended that where hierarchical aspects to the local naming scheme exist, they be mapped onto the hierarchical URL path syntax in order to allow the partial form to be used.

The following encoding method shall be used for mapping WAIS, FTP, Prospero and Gopher addresses onto URLs. Where the local naming scheme uses ASCII characters which are not allowed in the URL, these may be represented in the URL by a percent sign "%" followed by two hexadecimal digits (0-9, A-F) giving the ISO Latin 1 code for that character. Character codes other than those allowed by the syntax shall not be used in a URL.

The same encoding method may be used for encoding characters whose use, although technically allowed in a URL, would be unwise due to problems of corruption by imperfect gateways or misrepresentation due to the use of variant character sets, or which would simply be awkward in a given environment. As a % sign always indicates an encoded character, a URL may be made safer simply by encoding any characters considered unsafe, while leaving already encoded characters still encoded.

(Note: If a new naming scheme is introduced which encodes binary data as opposed to text, then a more compact encoding such as pure hex or base 64 would be more appropriate.)

The same considerations apply to mapping local fragment identifiers onto the fragmentid part of a URL.

Specific Schemes

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

HTTP

The HTTP protocol specifies that the path is handled transparently by those who handle URLs, except for the servers which dereference them. The path is passed by the client to the server with any request, but is not otherwise understood by the client. The fragmentid part is not sent with the request. The search part, if present, is sent. Spaces in URLs should be escaped for transmission in HTTP.

FTP

The ftp: prefix indicates a file which is to be picked up from the file system of the given host. The FTP protocol is used. The port number if given gives the port of the FTP server if not the FTP default. (A client may in practice use local file access to retrieve objects which are available though more efficient means such as local file open or NFS mounting, where this is available and equivalent).

The syntax allows for the inclusion of a user name and even a password for those systems which do not use the anonymous FTP convention. The default, however, if no user or password is supplied, will be to use that convention, viz. that the user name is "anonymous" and the password the user's mail address.

The adoption of a unix-style syntax involves the conversion into non-unix local forms by either the client or server. Some non-unix servers do this, but clients wishing to access sites which do not have unix-style naming will need certain algorithms to enable other file systems to be identified and treated. Client software may also have to be flexible in terms of the sequence of FTP commands used with different varieties of server. In view of a tendency for file systems to look increasingly similar, it was felt that the URL convention should not be weighed down by extra mechanisms for identifying these cases.

The data format of a file can only, in the general FTP case, be deduced from the name, normally the suffix of the name. This is not standardised. An alternative is for it to be transferred in information outside the URL. The transfer mode (binary or text) must in turn be deduced from the data format. It is recommended that conventions for suffixes of public archives be established, but it outside the scope of this paper.

News

The news locators refer to either news group names or article message identifiers which must conform to the rules of RFC 850. A message identifier may be distinguished from a news group name by the presence of the commercial at "@" character. These rules imply that within an article, a reference to a news group or to another article will be a valid URL (in the partial form).

Note1: Among URLs the news: URLs are anomalous in that they are location-independent. They are unsuitable as URN candidates because the NNTP architecture relies on the expiry of articles and therefore a small number of articles being available at any time. When a news: URL is quoted, the assumption is that the reader will fetch the arcticle or group from his or her local news host. News host names are NOT part of news URLs.

Note 2: An outstanding problem is that the message identifier is insufficient to allow the retrieval of an expired article, as no algorithm exists for deriving an archive site and file name. The addition of the date and news group set to the article's URL would allow this if a directory existed of archive sites by news group. Suggested subject of study in conjunction with NNTP WG. Further extension possible may be to allow the naming of subject threads as addressable objects.

WAIS

The current WAIS implementation public domain requires that a client know the "type" and length of a object prior to retrieval. These values are returned along with the internal object identifier in the search response. They have been encoded into the path part of the URL in order to make the URL sufficient for the retrieval of the object. If changes to WAIS specifications make the internal id something which is sufficient for later retrieval then this will not be necessary. Within the WAIS world, names do not of course not need to be prefixed by "wais:" (by the partial form rules).

The length not now being strictly necessary is kept for historical reasons.

Prospero

The Prospero (Neuman, 1991) directory service is used to resolve the URL yielding an access method for the object (which can then itself be represented as a URL if translated). The host part contains a host name or internet address. The port part is optional. The path part contains a host specific object name, an optional version number, and an optional list of attributes. If these latter fields are present thy are separated from the host specific object name and from each other by the characters "%00" (percent, zero, zero), this being and escaped string terminator (null). If the optional list of attributes is provided, the version number must be present, but may be the empty string (i.e. the first attribute would be seperaed from the host specific name by "%00%00"). External Prospero links are represented directly as URLs of the underlying access method and are not represented as Prospero URLs.

Gopher

The first character of the URL path part (after the initial single slash) is a single-character "type" field which is that used by the Gopher protocol. The rest of the path is the "selector string", with disallowed characters encoded. Note that some selector strings begin with a copy of the gopher type character, in which case that character will occur twice consecutively in the URL. If the type character and selector are omitted, the type defaults to "1". Gopher links which refer to non-Gopher protocols are represented directly as URLs of the underlying access method and are not represented as Gopher URLs.

Telnet, rlogin, tn3270

The use of URLs to represent interactive sessions is a convenient extension to their uses for objects. This allows access to information systems which only provide an interactive service, and no information server. As information within the service cannot be addressed individually or, in general, automatically retrieved, this is a less desirable, though currently common, solution.

x500

The mapping of x500 names onto URLs is not defined here. A decision is required as to whether "distinguished names" or "user friendly names" (ufn), or both, should be allowed. If any punctuation conversions are needed from the adopted x500 representation (such as the use of slashes between parts of a ufn) they must be defined. This is a subject for study.

WHOIS

This prefix describes the access using the "whois++" scheme in the process of definition. The hostname part is the same as for other IP based schemes. The path part can be either a whois handle for a whosi object, or it can be a valid whois query string. This is a subject for further study.

Network Management Database

This is a subject for study.

Registration of naming schemes

A new naming scheme may be introduced by defining a mapping onto a conforming URL syntax, using a new scheme identifier. Experimental scheme identifiers may be used by mutual agreement between parties, and must start with the characters "x-". The scheme name "urn:" is reserved for the work in progress on a scheme for more persistent names. Therefore URNs (Names) and URLs (Locators) be distinguishable. An object which is either a URL or a URN is known as a URI (Identifier).

It is proposed that the Internet Assigned Numbers Authority (IANA) perform the function of registration of new schemes. Any submission of a new URI scheme must include a definition of an algorithm for the retrieval of any object within that scheme. The algorithm must take the URI and produce either a set of URL(s) which will lead to the desired object, or the object itself, in a well-defined or determinable format.

It is recommended that those proposing a new scheme demonstrate its utility and operability by the provision of a gateway which will provide images of objects in the new scheme for clients using an existing protocol. If the new scheme is not a locator scheme, then the properties of names in the new space should be clearly defined. It is likewise recommended that, where a protocol allows for retrieval by URI, that the client software have provision for being configured to use specific gateway locators for indirect access through new naming schemes.

BNF syntax

This is a BNF-like description of the Uniform Resource Locator syntax. A vertical line "|" indicates alternatives, and [brackets]indicate optional parts. Spaces are representated by the word "space", and the vertical line character by "vline". Single letters stand for single letters. All words of more than one letter below are entities described somewhere in this description.

The "generic" production gives a higher level parsing of the same URLs as the other productions. The "national" and "punctuation" characters fo not appear in any productions and therefore may not appear in URLs.

The "afsaddress" is left in as historical note, but is not a url production

fragmentaddress	uri [# fragmentid]
uri	url
url	generic | httpaddress | ftpaddress | newsaddress | prosperoaddress | telnetaddress | gopheraddress | waisaddress
generic	scheme : path [? search]
scheme	ialpha
httpaddress	h t t p : / / hostport [/ path][? search]
ftpaddress	f t p : / / login / path
afsaddress	a f s : / / cellname / path
newsaddress	n e w s : groupart
waisaddress	waisindex | waisdoc
waisindex	w a i s : / / hostport / database [? search]
waisdoc	w a i s : / / hostport / database / wtype / digits / path
groupart	* | group | article
group	ialpha [. group]
article	xalphas @ host
database	xalphas
wtype	xalphas
prosperoaddress	prosperolink
prosperolink	p r o s p e r o : / / hostport / hsoname [% 0 0 version [attributes]]
hsoname	path
version	digits
attributes	attribute [attributes]
attribute	alphanums
telnetaddress	t e l n e t : / / login
gopheraddress	g o p h e r : / / hostport [/ gtype [selector]][? search]
login	[user [: password]@]hostport
hostport	host [: port]
host	hostname | hostnumber
cellname	hostname
hostname	ialpha [. hostname]
hostnumber	digits . digits . digits . digits
port	digits
selector	path
path	void | xpalphas [/ path]
search	xalphas [+ search]
user	xalphas
password	xalphas
fragmentid	xalphas

gtype xalpha
xalpha alpha | digit | safe | extra | escape
xalphas xalpha [xalphas]
xpalpha xalpha | +
xpalphas xpalpha [xpalpha]
ialpha alpha [xalphas]
alpha 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 | 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
digit 0 |1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
safe $ | - | _ | @ | . | &
extra ! | * | " | ' | (|) | : | ; | , | space
escape % hex hex
hex digit | a | b | c | d | e | f | A | B | C | D | E | F
national { | } | vline | [|]| \| ^ | ~
[punctuation] < | >
digits digit [digits]
alphanum alpha | digit
alphanums alphanum [alphanums]
void

Wrappers for URIs in plain text

This section does not formally form part of the URL specification.

URIs, including URLs, will ideally be transmitted though protocols which accept them and data formats which define a context for them. However, in practice nowadays there are many occasions when URLs are included in plain ASCII non-marked-up text such as electronic mail and usenet news messages.

In this case, it is convenient to have a separate wrapper syntax to define delimiters which will enable the human or automated reader to recognize that the URI is a URI.

The recommendation is that the angle brackets (less than and greater than signs) of the ASCII set be used for this purpose.

These wrappers do not form part of the URL, are not mandatory, and should not be used in contexts (such as SGML parameters, HTTP requests, etc) in which delimiters are already specified.

Example

```
Yes, Jim, I found it under <ftp://info.cern.ch/pub> but
you can probably pick it up from <ftp://ds.internic.net/rfc>.
```

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.

The use of URLs containing passwords is clearly unwise.

Conclusion

A need has been demonstrated, and a number of requirements have been stated for uniform resource locators (URLs). A scheme has been proposed which builds on existing conventions to define a syntax for URLs. This scheme has been in serious use by World-Wide Web (W3) initiative since 1991. Adoption of the scheme in correspondence, standards and software will ease the use of references to on-line information in a flexible way as the coming information age arrives.

Acknowledgements

This paper builds on the basic W3 design and much discussion of these issues by many people on the network. The discussion was particularly stimulated by articles by Clifford Lynch (1991), Brewster Kahle (1991) and Wengyik Yeong (1991b). Contributions from John Curran (NEARnet), Clifford Neuman (ISI) Ed Vielmetti (MSEN) and later the IETF URL BOF and URI working group have been incorporated into this issue of this paper.

The draft url4 (Internet Draft 00) was generated from url3 following discussion and overall approval of the URL working group on 29 March 1993. The paper url3 had been generated from udi2 in the light of discussion at the UDI BOF meeting at the Boston IETF in July 1992. Draft url4 was Internet Draft 00. Draft url5 incorporated changes suggested by Clifford Neuman, and draft url6 (ID 01) incorporated character group changes and a few other fixes defined by the IETF URI WG in submitting it as a proposed standard. URL7 (Internet Draft 02) incorporated changes introduced at the Amsterdam IETF and refined in net discussion.

References

Alberti, R., et.al. (1991) "Notes on the Internet Gopher Protocol" University of Minnesota, December 1991, <ftp://boombox.micro.umn.edu/pub/gopher/gopher_protocol>. See also <gopher://gopher.micro.umn.edu/00/Information About Gopher/About Gopher>

Berners-Lee, T ., (1991) "Hypertext Transfer Protocol (HTTP)", CERN, December 1991, <ftp://info.cern.ch/pub/www/doc/http-spec.txt>

Davis, F, et al., (1990) "WAIS Interface Protocol: Prototype Functional Specification", Thinking Machines Corporation, April 23, 1990 <ftp://quake.think.com/pub/wais/doc/protspec.txt>

International Standards Organization, (1991) Information and Documentation - Search and Retrieve Application Protocol Specification for open Systems Interconnection, ISO-10163

Huitema, C., (1991) "Naming: strategies and techniques", Computer Networks and ISDN Systems 23 (1991) 107-110.

Kahle, Brewster, (1991) "Document Identifiers, or International Standard Book Numbers for the Electronic Age", <ftp://quake.think.com/pub/wais/doc/doc-ids.txt>

Kantor, B., and Lapsley, P., (1986) "A proposed standard for the stream-based transmission of news", Internet RFC-977, February 1986. <ftp://ds.internic.net/rfc/rfc977.txt>

Lynch, C., Coallition for Networked Information: (1991) "Workshop on ID and Reference Structures for Networked Information", November 1991. See <wais://quake.think.com/wais-discussion-archives?lynch>

Mockapetris, P., (1987) "Domain names + concepts and facilities", RFC-1034, USC-ISI, November 1987, <ftp://ds.internic.net/rfc/rfc1034.txt>

Neuman, B. Clifford, (1992) "Prospero: A Tool for Organizing Internet Resources", Electronic Networking: Research, Applications and Policy, Vol 1 No 2, Meckler Westport CT USA. See also <ftp://prospero.isi.edu/pub/prospero/oir.ps>

Postel, J. and Reynolds, J. (1985) "File Transfer Protocol (FTP)", Internet RFC-959, October 1985. <ftp://ds.internic.net/rfc/rfc959.txt>

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

Yeong, W., (1991b), "Representing Public Archives in the Directory", Internet Draft, November 1991, now expired.

Author's address


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

     Telephone: +41 (22)767 3755
Fax:       +41 (22)767 7155
Email:     timbl@info.cern.ch
```