Tim Berners-Lee, Jean-François Groff, Robert Cailliau
CERN, 1211 Geneva 23, Switzerland
February 1992

Universal Document Identifiers on the Network

Status of this memo

This draft document is for unlimited distribution for discussion. Comments please to the author as timbl@info.cern.ch. or to the discussion list CNI–ARCH@UCCVMA.BITNET. Following such discussion a future revision of paper may be submitted to the Internet RFC editor for consideration as an internet standard.

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 used to identify documents. This is because names of documents are passed on in so many ways, from the backs of envelopes to hypertext documents, and may have a long life.

This paper discusses the requirements on a universal naming syntax which can be used to refer to documents available using existing protocols, and may be extended with technology. It makes a recommendation for a generic syntax, and its specific application to existing internet protocols.

Terms

The objects on the network which are to be named include objects which can be retrieved, and objects which can be searched.

In this paper we refer to the first as “documents”, no matter what they contain. We imply nothing about the contents at this stage. The “document” is the unit of retrieval and need not correspond to any unit of storage. We refer to objects which can be searched as “indexes”. We emphasize that this is the abstract view of the users, 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.

We use the terms “name” and “identifier” synonymously. The term “address” is reserved for an identifier which specifies a more or less physical location.

Uses of a document name

The name allows a user, with the help of a "client" program, to search indexes and/or retrieve documents from 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.

Basic requirements on a name are that

- A given name will remain valid as long as it is needed;
- A given naming syntax will remain valid through the introduction of new protocols and directory technologies;
- A name will contain enough information to allow the document or index to which it refers to be accessed.

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 document is outside the scope of this syntax.

Current practice

Current protocols use many different standards for names. For some protocols, such as ISO-10163 Search and Retrieve protocol[16], the identifiers 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 document retrieval at a later time. Typically, however, they are not long-lasting names which are independent of the location of the document. Such names may be provided using directory servers such as x.500[1]. They will refer to the registration, however formal or informal, of a document with a particular organization or person. Both hypertext and manual references rely on long-lasting names.

Current names are basically location specifiers (addresses). They give the necessary parts of an address for a reader to access an information provider using the given protocol, and ask for the document required. Examples of names used by various protocols include

File transfer protocol [Post85]:	Host name or IP-address [IP port] [user name, password] Filename

[1] Yeong in [Yeong9110] points out the desirability of using x.500 for holding addresses of FTP documents. Within the context of that paper, the "archive location" field could be a UDI. A UDI should also, however, be able to hold an x.500 name

W.A.I.S. [Kahl90]	Host name or IP-address [IP port] database name local document id
Gopher [Albe91]	Host name or IP-address [IP port] database name selector string
HTTP [Bern91]	Host name or IP-address [IP port] local document id
NNTP [Kant86] group	Group name
NNTP article	Host name unique message identifier
x.500 distinguished name	Country Organization Organizational unit Person Local document identifier

Other systems with their own naming schemes include BITNET “LISTSERV” application, FTAM file retrieval, SQLnet™ 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 document.

An addressing syntax is required which will be able to encompass existing physical address spaces, and be extendable 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. Obviously, ISO standard lower protocol layers will have their own forms of addressing, and new applications will have their own forms addressing subsections of documents.

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 document (its size, data format, authorization 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 document, and for protocols to allow the header information to be retrieved independently of the documents themselves.

Readability by people

This requirement has ben put forward by several people (Clifford Lynch, Doug Engelbart among others), and disputed by others. The author's view is that it will be a while before technology and standardization have reached the point at which names and identifiers 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, parsable by a human being.

Structure

A physical address is required in order for

- The user's program to contact the server
- The server to search and index or retrieve a document
- The user's program to locate an individual position or element within a document.

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. The client should normally be transparent to this part of the name.
- Information to be used by the application once the document 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 organization of the system. From the naming point of view, it has the advantage that a reference in one document to another document 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 [Huit91]: 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 favorite 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.

The use of white space characters has been avoided in UDIs: spaces are not legal characters. This was done 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.

The syntax[2] is described in two parts. Firstly, the syntax rules of a completely specified name are given: then, the rules under which parts of the name may be omitted in a well-defined context.

Full form

A complete address consists of a naming scheme specifier followed by an address whose format is a function of the naming scheme. For physical addresses of information on the internet, a common syntax is used for the internet address part. A BNF description of the UDI syntax is given in an appendix. The components are as follows.

Anchor-id

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

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

Scheme

Within the UDI of a document, the first element is the name of the scheme, separated from the rest of the document by a colon. The rest of the UDI 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 document 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 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 not the default number for the protocol, in decimal notation after a colon.

The rest of the address if 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.

[2] The proposed system is based on the address syntax which has been adopted succesfully for an 18-month pilot run of the World-Wide Web (W3) global hypermedia project.

Path

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

Within a document whose UDI is well defined, the UDI of another document may be given in abbreviated form, where parts of the two UDIs are the same. This allows documents within a group to refer to each other without requiring the space for a complete reference, and it incidentally allows the group of documents to be moved without changing any references.

This relies on a property of the UDI syntax that certain characters ("/") and certain path elements (“..”, “.”) have a significance reserved for representing a hierarchical space, and must be recognized as such by both clients and servers.

The rules for the use of a partial name are:

- If the *scheme* parts are different, the whole absolute address must be given. Other wise, 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 address 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 address (anything following the rightmost slash) is removed, and the given relative address appended in its place.
- Within the result, all occurrences of "/xxx/.." or "/." are recursively removed, where xxx, “..” and “.” are complete path elements.

Mapping Local Names

When a system uses a local addressing scheme, it is useful to provide a mapping from local addresses into UDIs so that references to documents 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 UDIs. It is recommended that where hierarchical aspects to the local naming scheme exist, they be mapped onto the hierarchical UDI path syntax in order to allow the partial form to be used.

The following escaping method is used for mapping WAIS and Gopher addresses onto UDIs. Where the local naming scheme uses ASCII characters which are not allowed in the UDI, these may be represented in the UDI by a percent sign “%” followed by two hexadecimal digits (0-9, A-F) giving the

ASCII value for that character. If non-ASCII characters are used, then a similar escaping system should be used. Character codes other than those allowed by the syntax shall not be used in a UDI.

The same considerations apply to mapping local anchor identifiers onto the *anchorid* part of a UDI.

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

FTP

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 not felt that the UDI convention should be weighed down by extra mechanisms for identifying these cases.

News

The news addresses 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 UDI (in the partial form).

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 filename. The addition of the date and news group set to the article's UDI would allow this if a directory existed of archive sites by news group.

WAIS

The current WAIS implementation public domain requires that a client know the "type" and length of a document prior to retrieval. These values are returned along with the internal document identifier in the search response. They have been encoded into the path part of the UDI in order to make the UDI sufficient for the retrieval of the document. If changes to WAIS specs make the internal id something which is sufficient for later retrieval then this will not be necessary.

Within the WAIS world, identifiers do not of course not need to be prefixed by "wais:" (by the partial form rules).

Gopher

The first part of the UDI path part is a single-character type field which is that used by the Gopher protocol.

Telnet, rlogin

The use of UDIs to represent interactive sessions is a convenient extension to their uses for documents. 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.

Conclusion

A need has been demonstrated, and a number of requirements have been stated for universal document identifiers (UDIs). A scheme has been proposed which builds on existing conventions to define a syntax for UDIs. Adoption of the scheme in correspondence, standards and software will ease the use of references to online information in a flexible way as the coming information age arrives.

Acknowledgements

This paper builds on much discussion of these issues by many people on the network. The discussion was particularly stimulated by articles by Clifford Lynch [Lync91], Brewster Kahle [Kahl91] and Wengyik Yeong [Yeon2].

REFERENCES

[Albe91] Alberti et.al. "Notes on the Internet Gopher Protocol" Univeristy of Minnesota, December 1991, UDI=file://boombox.micro.umn.edu/pub/gopher/gopher_protocol. See also UDI=gopher://gopher.micro.umn.edu:70/00/Information%20About%20Gopher/About%20Gopher

[Bern91] Berners-Lee, T., "HTTP as implemented in WWW", CERN, December 1991, UDI=file://info.cern.ch./pub/www/doc/http.txt

[ISOSR] International Standards Organization, *Information and Documentation – Search and Retrieve Application Protocol Specification for open Systems Interconnection*, ISO-10163

[Huit91] Huitema, C., "Naming: strategies and techniques", *Computer Networks and ISDN Systems* **23** (1991) 107-110.

[Kahl91] Kahle, Brewster, "Document Identifiers, or International Standard Book Numbers for the Electronic Age", UDI=file://quake.think.com/pub/wais/doc/doc-ids.txt

[Kahl90] Kahle, B., et. al., "WAIS Interface Prototype Functional Specification", Thinking Machines Corporation, April 1990 UDI=file://quake.think.com/pub/wais/doc/wais-concepts.txt

[Kant86] Kantor, B., and Lapsley, P., "A proposed standard for the stream-based transmission of news", Internet RFC-977, February 1986. UDI=file://nnsc.nsf.net/rfc/rfc977.txt

[Lync91] Lynch,C. , Coallition for Networked Information: "Workshop on ID and Reference Structures for Networked Information", November 1991. See UDI=wais://quake.think.com/wais-discussion-archives?lynch

[Mock87] P. Mockapetris, "Domain names – concepts and facilities", RFC-1034, USC-ISI, November 1987, UDI=file://nnsc.nsf.net/rfc/rfc1034.txt

[Post85] Postel, J. and Reynolds, J. "File Transfer Protocol (FTP)", Internet RFC-959, October 1985. UDI=file://nnsc.nsf.net/rfc/rfc959.txt

[Yeon1] Yeong, W., "Towards Networked Information Retrieval", *Technical report 91-06-25-01*, Performance Systems International, Inc. UDI=file://uu.psi.com/wp/nir.txt

[Yeon2] Yeong, W., P.S.I., "Representing Public Archives in the Directory", *Internet Draft*, November 1991. In UDI=wais://nnsc.nsf.net/internet-drafts?yeong

Appendix: BNF syntax of Universal Document Identifiers

This is a BNF-like description of the W3 addressing syntax. We use a vertical line "|" to indicate alternatives, and [brackets] to indicate optional parts. Spaces are representational only: no spaces are actually allowed within a UDI. Single letters stand for single letters. All words of more than one letter below are entities described somewhere in this description.

anchoraddress	docaddress [**#** anchorid]
docaddress	generic | httpaddress | fileaddress | newsaddress | telnetaddress | gopheraddress | waisaddress
generic	scheme : path
scheme	ialpha
httpaddress	**h t t p : / /** hostport [**/** path] [**?** search]
fileaddress	**f i l e : / /** host / 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
telnetaddress	**t e l n e t : / /** [user @] hostport
gopheraddress	**g o p h e r : / /** hostport [**/** gtype [**/** selector]] [**?** search]
hostport	host [**:** port]
host	hostname | hostnumber
hostname	ialpha [**.** hostname]
hostnumber	digits **.** digits **.** digits **.** digits

port	digits
selector	path
path	void | xalphas [**/** path]
search	xalphas [**+** search]
user	xalphas
anchorid	xalphas
gtype	xalpha
xalpha	alpha | **$** | _ | @ | **!** | **%** | **^** | **&** | ***** | **(** | **)** | **.** | digit
xalphas	xalpha [xalphas]
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**
digits	digit [digits]
alphanum	alpha | digit
alphanums	alphanum [alphanums]
void	