Internationalization of the Handle System – A Persistent Global Name Service

Sam X. Sun
Corporation for National Research Initiatives
Reston, VA USA
ssun@cnri.reston.va.us
http://cnri.reston.va.us

Abstract

The Handle System is a distributed computer system developed at the Corporation for National Research Initiatives (CNRI). The Handle System provides globally unique, persistent naming and name resolution service over the Internet. This paper gives a brief description of the Handle System, and further discusses internationalization issues involved in designing such a system, specifically the encoding issues of the global namespace to ensure global uniqueness, global readability, and global transcribability. The paper separates the current major practices of character set encoding and their practical use in the Handle System into three categories, namely, the limited character set approach, the use of encoding tags, and the Unicode based encoding. It concludes that, for any global name service to achieve global acceptance, every native alphabet has to be supported. Unicode based UTF-8 encoding is the recommended approach, mainly because it allows recognizable names to be defined in terms of any native language alphabet, including, but not limited to English or western European languages. Further, the paper provides details on how handles are defined within the Handle System using the UTF-8 encoding, as well as how handles can be used in the World Wide Web environment using current browser technology.

1. The Handle System

The Handle System [1] is a comprehensive service for assigning, managing, and resolving persistent names or identifiers, known as "Handles", for network resources. Each handle defined in the Handle System is globally unique, and can be used as a persistent reference to any resource over the Internet. (Note that in this document we do not attempt to distinguish between the term 'name' and 'identifier' and will use them interchangeably.)

The Handle System provides (a) an open set of protocols for a distributed computer system [2,3], so that handles can be stored and resolved into their corresponding digital resources in a unambiguous manner; (b) a global name space, which allows network resources to be named in terms of any native alphabet; and (c), an implementation of the distributed global name service based on the handle protocols. The service provides a persistent global naming service, based on a centrally administered naming authority registration service, and a global name resolution service integrated with popular web browsers.

Handles defined in the Handle System are readily accessible by any application over the Internet via the Handle Resolution protocol [2]. In particular, handles can be resolved from popular web browsers using the Handle System Resolver [4], Handle System Proxy Service, and in the future, the URN (Uniform Resource Names) services defined by the IETF URN working group [5,6]. The Handle System is designed to conform to all the fundamental requirements as outlined in the URN/URI specifications [6,7].

Specifically, the Handle System was designed to address the following issues in network resource identification:

- Persistence

A handle name provides an identity, ie. a name, for a resource. The handle name can remain unchanged even when characteristics of the named resource change over time. For example, the Handle System could provide persistent names for resources that outlast any specific computer system or organization. In contrast, any resource name that is inextricably linked with a specific system or the name of an organization will not be able to survive the demise or radical change of that computer system or organization. By separating the object name from location, ownership, and other state information, the Handle System allows that name to persist over time.

- Location independence

With handles, references to network resources can be made independent of their network location. This allows easy reorganization of information. Handles make it possible to transfer resources from one organization to another without affecting or breaking the existing user references (i.e., handles) to those collections. This is not possible using location based references, such as the URLs used in the World Wide Web environment.

- Multiple instances of the underlying object

A handle can refer to more than one instance of the underlying object. As an example, CNRI is the publisher of D-Lib Magazine [8]. The magazine is mirrored in Britain. By giving a handle to each article, the Handle System can store data that lists each instance of the article and how to access it. Using a single name, a handle, as a reference to multiple instances of the underlying object, provides clients with alternative ways to access the object, even when one of its storage locations is not readily accessible. It also provides information that can be used to access the object more efficiently over a distributed network environment.

2. Handle System – A Unicode Based Global Name Service

2.1. Character Set Encoding Issues in Global Naming

In summary, the three fundamental character set encoding issues faced by any global naming scheme are global uniqueness, global readability, and global transcribability.

2.1.1. Global Uniqueness

Global uniqueness requires that the name given to an object in its encoded form not be repeated by another name within the namespace. Overlappping character set encodings could potentially result in different names from different character set encodings yielding the same octet value. Any global naming scheme should be designed to prevent this from happening.

For historic reasons, computers (hardware/software) and network protocols were designed to use 8 bits as a basic unit for information storage and transformation. And the same octet encoding space was selected to encode the character set for most of the world’s native language character sets [9]. While this is sufficient to represent characters used in many native language character sets independently, not enough encoding space is available to identify every character of, say Greek, Hebrew, Arabic, and English combined. Different languages or native character sets are re-using the same 8-bit encoding space to encode their own character set independently, causing overlapping of the encoding space in the global computing environment. The situation only became more confused when multi-byte encodings was introduced.

As an example of overlapped encoding among different native alphabets, the octet sequence “**0xE1, 0xE2, 0xE3**” maps to “Alpha, Beta, Gamma” under ISO-8859-7 encoding, but also maps to “Small A with Acute, Small A with a Circumflex, Small A with a Tilde” under ISO-8859-1 encoding. Without specifying the encoding used, the same byte sequence “**0xE1, 0xE2, 0xE3**” can be a name “αβγ” in Greek, while it can also be a name “áâã” in Latin.

Any global name service has to ensure that every octet sequence (eg. "0xE1, 0xE2, 0xE3") can map to no more than one named object. Thus the global name service has to define the encoding used so that no two names share the same binary encoding, and any name defined within the system can be resolved unambiguously to its corresponding object.

2.1.2. Global Readability

Global readability requires the ability to create recognizable names in any native alphabet, and be displayed in its native computing environment. By limiting the character set to ASCII, it is impossible for non-English speaking people to create recognizable names under their native computing environment. Over 90% of the world's population are not native English speakers. Just as there is a need to allow HTML documents to be encoded using any native character set, so should names be permitted in any native alphabet. Many systems designed with ASCII-only in mind will have to accommodate other language encoding eventually when used in a global context.

No single native alphabet encoding is recognizable for everyone in the world. While names defined in printable ASCII characters appear to be most recognizable in the traditional computing environment, ASCII names can be viewed as equally unrecognizable to many non-English speaking people as Kanji names are to English speaking people.

If the native character set encodings overlap, when documents encoded in one native character set get used or displayed in another encoding environment, the result is frequently totally unreadable. A name encoded in one encoding could look totally different or unrecognizable under a different encoding environment. For example, the name "日本" (Japan) defined using CJK encoding would look like "eäg," under the ISO-8859-1 encoding environment. Thus it is also necessary for any global name service to define its encoding so that any name defined within the service can have a deterministic presentation, and be recognizable in terms of the native alphabet upon which the name is defined.

2.1.3. Global Transcribability

Global transcribability means that names should be transcribable from one medium to another (such as voice to paper, or paper to keyboard), in terms of any native alphabet. While people may decide to define a name in ASCII so that it can be transcribed by most current computing environments around the world, they may equally desire to create names in their native alphabet so that they and their colleagues can more easily recognize and use them later. Although the current lack of a universal input method presents some difficulty for names defined in many native alphabets to be transcribed easily by anyone anywhere, the advancement of other technologies, such as OCR or hand written recognition are making it easier and easier.

A name can be defined so that it can be transcribed under most of the native computing environments, using ASCII printable characters or digits only. It can also be defined so that it can be easily recognized and transcribed by people sharing the same native alphabet and computing environment. The decision to use any particular alphabet for any given name should be made by the person who assigns the name, and should not be restricted by the underlying technology. It is not practical to enforce every document around the world to be encoded in ASCII only; neither will it be acceptable to enforce the world to name every object in terms of any particular native alphabet. The global name service should allow names to be defined so that they can be transcribed in terms of any native alphabet.

No single native alphabet is readily transcribable into every computing environment in the world. While ASCII names can be transcribed easily into traditional computing environment, these names can be hard to enter for non-English speaking people, and there are keyboards designed for non-English speaking users that do not have the ASCII characters on them. For example, keyboard in Japanese (normal Kana state), Greek (normal state), Hebrew (normal state), or Arabic (normal state) will have their native characters located in place of the ASCII characters on the standard English keyboard.

2.2. Three Approaches to Character Set Encoding for Global Naming

A close examination of current encoding methods used to identify or interpret digital objects in practice reveals that they can be summarized into three categories: the Limited Character Set Approach, the Use of Encoding Tags, and the Unicode Based Encoding.

2.2.1. The Limited Character Set Approach

An example of this approach is the current URL specification defined in RFC1738[10] and RFC1808[11]. The URL specification defines a subset of the US ASCII character set as the only "legal" characters for use in the URL context, and specifies that all other characters, including those of any non-English languages, must be hex encoded (eg. %FF) when used in the URL.

Restricting URLs to printable ASCII characters has its own advantages. Because the ASCII character set encoding is an invariant common subset of many native character set encodings registered under ISO, names using ASCII characters usually maintain their unique encoding in the global scope. Printable characters are those ASCII characters defined on the standard keyboard. Names using printable ASCII characters can be easily transcribed by computing environments equipped with standard keyboards. But this approach rules out the use of many native alphabets to form recognizable names in non-English speaking environments, which is an extreme constraint for any global name service, as we discussed earlier.

An extreme case of the Limited Character Set Approach is to limit the character set to digits encoded in ASCII. Such an approach can be called the Digital ID approach. An example of this approach is the use of Global Unique ID (GUID), used by the Microsoft Corporation to uniquely identify COM objects.

Digital IDs ensure their uniqueness regardless of the native language or character set. Digital IDs are easily transcribable across any media, as long as they can be remembered and typed in correctly. On the other hand, Digital IDs have a severe limitation in human practice due to their non-descriptive nature. They may eventually require a corresponding descriptive name system to describe the Digital IDs, which will bring us back to the original problem. For example, even though every COM object has its global unique ID to identify itself, most COM developers assign their COM object a descriptive name also. The descriptive name helps developers, as well as users of the COM objects, to identify or use those objects in practice, but there is no guarantee that the descriptive names will not collide. Thus even though GUID, using ASCII digits only, can uniquely identify any COM object, the practical use of descriptive names makes the named objects still vulnerable to name conflicts.

The URL approach is designed for the English speaking domain, and can be characterized as ASCII ID approach. It shares the same transcribability features of the Digital ID approach. When used in a non-English language context however, it has the same limitations as the Digital ID approach. The ASCII ID approach may force non-English speaking people to encode their objects into some encoding totally unrecognizable to them, which is difficult to transcribe and unacceptable in practice. In reality, users of non-English environments can and are ignoring the character set limits specified by the URL specification, and creating URLs using their own native character sets.

An example of a working URL containing non-ASCII characters is as follows:

> http://www.handle.net/resolver/北京.htm under GB2312 (Simplified Chinese) encoding,
>
> or, the same document's URL reference,
>
> http://www.handle.net/resolver/ÖÐÎÄ.htm under ISO-8859-1 (western European) encoding.

It is interesting to note that, because the URL defines the specific network "location" of a network resource, it generally will not cause ambiguity when non-ASCII encoding is used, especially when it is used to specify a file path where octet value, instead of the exact alphabet, is used to determine the corresponding document. The same is not true for the global naming service. As in the example discussed earlier, the octet string "0xE1, 0xE2, 0xE3" without encoding defined will resolve to an object named "αβγ" where Greek encoding (ISO-8859-7) is used. It will also resolve to another object "áâã" where LATIN 1 encoding (ISO-8859-1) is used.

2.2.2. The Use of Encoding Tags

This is the approach used by MIME[12], LDAP[13,14], and many other application level protocols. It generally attaches a tag (an ASCII string) before or after the original context. The tag uniquely identifies the native character set and the encoding used by the context. The assignment of the tags for various native character sets and their encoding are defined in various ISO standards [15,16,17], and the proper use of these tags for various native language contexts is defined in RFC1766[18].

An example of the Use of Encoding Tags provided in RFC2045 [12] is as follows:

> Content-type: text/plain; charset=iso-8859-10
> Content-Language: i-sami-no (North Sami)
> { context using ISO-8859-10 character set encoding. }

Use of Encoding Tags allows the interchange of documents of various native encodings with minimal or no change to the original content. A name using any native character set encoding with the proper tag can uniquely identify a single object, and it can be displayed and recognized in terms of its native alphabet. Names defined in a non-ASCII alphabet can be transcribed and recognized easily under the computing environment where that particular encoding is supported.

Allowing every kind of native encoding, with proper tag attached, to be used to name network resources allow names to be defined in terms of any native alphabet. It also allows existing native software to be used to transcribe names from various media.

On the other hand, this approach requires the additional character set prefix to be associated with the name. Applications will have to recognize all of the character set encodings, and possibly be able to process all those encodings they intend to support. There are over one hundred encoding schemes defined by the ISO standards, and the many equivalent names from various encodings makes it unnecessarily complex for any global name service to maintain and manage. Thus a default encoding is needed for use by the name service as the canonical form for the name of any digital object. The default encoding should allow every native alphabet to be used.

2.2.3. The Unicode Based Encoding

ISO 10646 defines Unicode [17], a single character set encoding for all characters of all the native languages used around the world. In Unicode, each native character of any native alphabet has its unique code point. Using Unicode encoding, names can be defined in terms of any native alphabet, with confidence in its encoding uniqueness from any other names using any native alphabet.

The Unicode encoding removes the limitations of 8-bit character set encoding, but has the problem of inter-operability. In general, it requires substantial change to existing systems, and presents some compatibility issues on certain platforms. A variation of Unicode encoding, UTF-8 [19] provides a better practical solution. A good description of UTF-8 encoding is given in the "The Properties and Promises of UTF-8" [20].

UTF-8 encoding is based on the Unicode character set, hence it preserves the native language transcribability by allowing any native alphabet to be used. Using UTF-8 encoding, names defined using ASCII characters remain unchanged in their binary format. The selection of UTF-8 as the canonical form for names defined and transmitted over the Internet simplifies the task for the global name service. It has only one encoding to deal with, and every name can be stored and managed in one and only one format.

Because any character in any native character set has its own unique UTF-8 encoding, no overlapping of encoding will occur among different native alphabets, hence ensuring global uniqueness.

In practice, because of the lack of software that supports UTF-8 and the existence of various native computing environments, it is still not practical to force every name to be entered in UTF-8 encoding. A combination of UTF-8 encoding and the Use of Encoding Tags on the client side of the name service is used by the Handle System to resolve the situation.

UTF-8 encoding is designed to allow international interchange of any documents in any native language, and is becoming widely accepted as the future encoding standard for documents intended for international exchange. As international awareness rises, it is expected that more and more applications will be able to generate and display UTF-8 encoded names or documents directly, which could eventually make the encoding tags unnecessary.

2.3. Handles defined using UTF-8.

The Handle System defines handles using UTF-8 encoding. This allows handles to be specified in terms of any native alphabet directly, thus preserving readability and transcribability for names defined in any native language. Because each different character in any native language has a different encoding under UTF-8, global uniqueness of each handle name is ensured.

Every handle in the Handle System is defined in two parts: its naming authority, and a unique handle string created under that naming authority. Naming authorities designate the administrative unit under which the handle string is created. Once created, the handle is treated as an opaque string. That is, the handle persists even if the administrative responsibility for that handle is shifted from the original creator. Naming authorities are currently defined using a subset of ASCII characters, including alphanumeric characters, and some special characters listed below. Characters in the naming authority segment are case insensitive.

The handle string uses UTF-8 character set encoding. The handle string can consist of any characters of any native language alphabet. ACSII characters within the handle string are case insensitive.

The following is the handle syntax described in Extended Backus-Naur Form (EBNF) notation:

```
<Handle>              ::=  <Naming Authority> "/" <HandleString>

<Naming Authority>    ::=  ( <AlphaNumeric> | "." | "-"  | "_" ) +

<HandleString>        ::=  ( < octet from UTF-8 encoding > ) +

<AlphaNumeric>        ::=  “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” | “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”
```

And here are some examples of valid handles:

cnri.dlib/july95-arms

10.1000/1

```
any-printable-characters/a-zA-Z0-9!@#$%^&*()_"<>,.?/`~|\
```

cnri.test/日本

handles-in-germany/Universität-Karlsruhe

Lack of a universal input method for all native language characters does affect the easy transcribability of handles defined in non-ASCII languages. For example, one can imagine the difficulty of English speaking people trying to enter a handle found printed in a Kanji publication. Until this problem is solved, any handle targeted for international reference is best defined in ASCII, or provided an ASCII "Alias Handle" (a handle which refers to another handle) to the natively named handles for easy international reference.

3. Handles used in the WWW

Handles in the Handle System are defined using UTF-8 encoding, but not every software or computing environment can accept or display the UTF-8 encoded names. Thus it is necessary to define a method so that handles can be entered and/or displayed in any native encoding from any native computing environment.

When handles are referenced in the World Wide Web context, the handle reference is defined by:

<HandleRef> ::= [<Modifier> "@"] <Handle Name>

Here the <Handle Name> identifies the character string (not the binary encoding) of the Handle stored and managed by the Handle System. The <Handle Name> can consist of any native language alphabet. One possible use of the <Modifier> is to specify the encoding information used by the <Handle Name>. Combined with the <Modifier>, the <HandleRef> maps to a unique UTF-8 encoded <Handle> stored and managed by the Handle System.

Note that the <Modifier> is positioned in front of the naming authority of the underlying <Handle>. This is to avoid defining any reserved characters in the name space of <HandleString> as defined in section 2.3, by taking advantage of the constrained character set of <Naming Authority> segment.

It is important to distinguish <Handle> from <HandleRef>. A <Handle> is defined in UTF-8 encoding as specified in section 2.3, and a <HandleRef> can use any native character set encoding supported by any native computing environment. A <HandleRef> identifies a <Handle> after conversion into the corresponding UTF-8 encoding based on information provided by its <Modifier>. The <Modifier> is a mechanism allowing specification of customized usage of the underlying handles. It is not part of the name of the underlying resource, nor is it part of the handle resolution process on the server side of the Handle System. The implementation and interpretation of <Modifier> is left to the client software using the Handle System.

The <Modifier> of the <HandleRef> is optional. When it is missing, it is assumed that UTF-8 is the encoding used by the <HandleRef>. Because ASCII names are preserved under UTF-8 encoding, handles defined using ASCII characters can be used directly without any <Modifier> attached.

3.1. Handle URI format

Handles can be used in place of URIs in Web browsers or in HTML documents to refer to persistent Internet resources. The direct approach is to use the handle preceded with the URI scheme "hdl:". This is called Handle URI Syntax:

<Handle URI Syntax> ::= "hdl:" <HandleRef>

The Handle URI Syntax has a reserved characters % and an excluded character " from the native handle name space. The character % is used for hex encoding, which is necessary to allow any characters, printable or not under any native environment, to be entered from the standard keyboard. And the character " is excluded so that handles can be separated from the surrounding context. Reserved or excluded characters must be hex encoded when used in the Handle URI context. The choice of % and hex encoding makes it compatible with the current URI practice. Because some browser implementations drop the # character when processing the URI, regardless of the URI scheme, hex encoding of character # is also strongly recommended. Examples of handles using Handle URI Syntax are:

hdl:jis@cnri.test/曰本

hdl:cnri.test/handle%25abc

Handles can also be resolved using a URL that references a Handle System Proxy Service. Users who choose to use the proxy service can have their handles specified via Handle Proxy Syntax:

<HandleProxySyntax> ::= "http://" <ProxyServer> "/" <HandleRef>

In this case, the <Proxy Server> is the domain name of a proxy server that will be responsible for resolving the <Handle> and directing the resolution results back to the client. Since this is a URL, any reserved/excluded characters in the URL specifications [10, 11] will have to be hex encoded when used in the <HandleRef>.

Examples of handles using Handle Proxy Syntax are:

http://hdl.handle.net/cnri.dlib/july95-arms

http://hdl.handle.net/jis@cnri.test/曰本

http://hdl.handle.net/cnri.test/handle%25abc

Here "hdl.handle.net" is a proxy service provided by CNRI to allow handles to be resolved using the "http:" protocol.

The handle name space by itself does not impose any hex or escape encoding, nor does the underlying Handle System. The reserved characters and hex encoding are introduced only when handles are used in the URI context. It is the client software's responsibility to decode any hex encoding in the Handle URI before sending the handles out for resolution. On systems where a character set encoding other than UTF-8 is used, it is also the client software's responsibility to convert the natively encoded handle to its UTF-8 encoding before sending it out for resolution.

3.2. Resolving Handles using the Handle System Resolver

Handles specified using Handle URI Syntax (ie, hdl:<handle>) can be resolved from a Web browser directly using the Handle System Resolver [4]. The Handle System Resolver is freely available software developed by CNRI to be used with popular Web browsers. It resolves handles into corresponding URLs, which are then retrieved by the browser in the normal fashion. This is the recommended way to resolve handles, since it provides good performance, is scalable, and is locally configurable.

Handles in URI format can be resolved directly if specified using UTF-8 encoding. Because of the good heuristic characteristics of UTF-8 encoding [20], the Resolver can always check if this is the case, and perform resolution directly if UTF-8 is detected.

In practice, handles referenced in a document are likely to be encoded in whatever encoding is used in the rest of the document. The encoding may not be the one configured or supported by the workstation or the browser. For handles not using UTF-8 encoding, encoding information will have to be provided by either the host document or the <Modifier> of the Handle URI. The encoding information will be used by the Resolver to correctly convert the handle into UTF-8 and perform the resolution.

The situation is more complicated for multi-lingual documents where multiple character set encodings are used. Either the encoding of the handle must be specified in the Handle URI through the <Modifier>, or the Resolver will have to parse the document to determine the encoding used by the handle reference. Because of the high cost of parsing the host document, Handle URIs using a different character set encoding from the host document should always have encoding specified in the <Modifier>.

For example, a Handle URI using JIS encoding should be entered as:

hdl:jis@cnri.test/曰本

The process for the Resolver to submit a Handle URI for resolution can be summarized as follows (please note that this is not yet fully implemented in the Resolver at the time of this writing):

- If the character set encoding is specified in the Handle URI through the <Modifier> followed by character @, use that as the encoding to convert the Handle URI into the corresponding UTF-8 encoding, and then send the result for resolution by the Handle System.

- Otherwise, the Resolver would assume that the Handle URI is entered using the default UTF-8 encoding. Because of the high heuristic characteristic of UTF-8 encoding [20], the Resolver could further detect if the URI is valid. If so, the Resolver will send the URI out for resolution.

- Otherwise, the Handle URI is not UTF-8 encoded, and does not have the encoding information provided for proper conversion into UTF-8. As a last resort, the Resolver could check if the host document of the Handle URI has its character set encoding defined. If so, it could use that as the encoding used by the Handle URI.

- Otherwise, the Handle URI is not UTF-8 encoded, and the Resolver should report an encoding error of the Handle URI.

3.3. Resolving Handles using a Handle Proxy Server

Handles can also be resolved by embedding them as http URLs in which the server name references a Handle Proxy Server. For example, a Handle Proxy URL

http://hdl.handle.net/ jis@cnri.test/曰本

will submit the <HandleRef> "jis@cnri.test/曰本" to the proxy service "hdl.handle.net". (Note that the URL contains "illegal" characters according to RFC1738 [21], but never the less works.)

Handles specified using Handle Proxy URL syntax are subject to the syntax restrictions on http URLs, specifically the set of reserved/excluded characters as defined in RFC1738 [21]. The proxy server performs the handle resolution task, and sends the resulting URL to the client browser for processing. Currently, CNRI provides global handle proxy service through "hdl.handle.net".

Even though using the proxy server approach is straight forward and does not require any customer software customization, it has the effect of tying the handles to the proxy server's location. Hence the reference to a proxy server should be made with care. This is a good reason for using the Handle System Resolver instead of the proxy server.

The current http protocol does not inform the server of the character set encoding of the host document of any URL submitted. Although the server may detect the language preference of the client, that may not be the exact encoding used by any Handle URL reference. Thus non UTF-8 encoded handles using the Handle Proxy Syntax should always have their character set encoding explicitly specified through a modifier. This will allow the proxy service to convert the handle into its correct UTF-8 encoding for resolution.

4. Handle System Applications

The Handle System is currently in use in a number of advanced prototype projects, including efforts with the National Digital Library Program (NDLP) at the US Library of Congress, the networked Computer Science Technical Reports Library (NCSTRL), the Copyright Office Electronic Registration, Recordation and Deposit System (CORDS), the Defense Technical Information Center (DTIC), the United States Information Agency (USIA), and the International Digital Object Identifier (DOI) Foundation. Details of these activities can be found in the Handle System home page at http://www.handle.net.

5. References

[1] The Handle System home page. http://www.handle.net

[2] Handle Resolution Protocol. http://www.handle.net/client_spec.html

[3] Handle Administration Protocol. http://www.handle.net/handle_admin.html

[4] Handle System Resolver. http://www.handle.net/resolver/index.html

[5] Moats, Ryan, "URN Syntax", RFC-2141, May 1997.
http://ds.internic.net/rfc/rfc2141.txt

[6] Berners-Lee, T., "Universal Resource Identifiers in WWW" RFC 1630, June 1994.
http://ds.internic.net/rfc/rfc1630.txt

[7] Sollins, K. and L. Masinter, "Functional Requirements for Uniform Resource Names"
RFC 1737, December 1994.
http://ds.internic.net/rfc/rfc1737.txt

[8] D-Lib Magazine. http://www.dlib.org

[9] Guide to Open System Specification, "Character Sets - Historical background".
http://www.ewos.bye/tg-cs/ghist.htm

[10] Berners-Lee, T., Masinter, L., McCahill, M., et al.,
"Uniform Resource Locators (URL)" RFC1738, December 1994.
http://ds.internic.net/rfc/rfc1738.txt

[11] Fielding, R., "Relative Uniform Resource Locators", RFC 1808, June 1995.
http://ds.internic.net/rfc/rfc1808.txt

[12] N. Freed & N. Borenstein, "Multipurpose Internet Mail Extensions (MIME)
Part One: Format of Internet Message Bodies," RFC 2045, November 1996.
http://ds.internic.net/rfc/rfc2045.txt

[13] M. Wahl, T. Howes, " Lightweight Directory Access Protocol" RFC1777, March 1995.
http://ds.internic.net/rfc/rfc1777.txt

[14] M. Wahl, T. Howes, "Use of Language Codes in LDAPv3", September 26, 1997 Work in Progress.
ftp://ftp.ietf.org/internet-drafts/draft-ietf-ldapext-lang-00.txt

[15] ISO 639:1988 (E/F) - Code for the representation of names of languages - The International Organization for Standardization, 1st edition, 1988.
17 pages Prepared by ISO/TC 37 - Terminology (principles and coordination).

[16] ISO 3166:1988 (E/F) - Codes for the representation of names of countries - The International Organization for Standardization, 3rd edition, 1988-08-15.

[17] The Unicode Consortium, "The Unicode Standard, Version 2.0", Addison-Wesley Developers Press, 1996, ISBN 0-201-48345-9.

[18] H. Alvestrand, "Tags for the Identification of Languages" RFC1766, March 1995.
http://ds.internic.net/rfc/rfc1766.txt

[19] Yergeau, Francois, "UTF-8, A Transform Format for Unicode and ISO10646", RFC2044, October 1996.
http://ds.internic.net/rfc/rfc2044.txt

[20] Martin J. Dürst: "The Properties and Promises of UTF-8". Presented at the 11th International Unicode Conference, San Jose, CA, Sept. 1997.
http://www.ifi.unizh.ch/groups/mml/people/mduerst/papers.html

[21] Berners-Lee, T., Masinter, L., McCahill, M., et al., "Uniform Resource Locators (URL)", RFC1738, December 1994.
http://ds.internic.net/rfc/rfc1738.txt