Network Working Group
Request for Comments: 2141
Category: Standards Track

R. Moats
AT&T

May 1997

URN Syntax

Status of This Memo

Abstract

Uniform Resource Names (URNs) are intended to serve as persistent, location-independent, resource identifiers. This document sets forward the canonical syntax for URNs. A discussion of both existing legacy and new namespaces and requirements for URN presentation and transmission are presented. Finally, there is a discussion of URN equivalence and how to determine it.

1. Introduction

Uniform Resource Names (URNs) are intended to serve as persistent, location-independent, resource identifiers and are designed to make it easy to map other namespaces (which share the properties of URNs) into URN-space. Therefore, the URN syntax provides a means to encode character data in a form that can be sent in existing protocols, transcribed on most keyboards, etc.

2. Syntax

All URNs have the following syntax (phrases enclosed in quotes are REQUIRED):

<URN> ::= "urn:" <NID> ":" <NSS>

where <NID> is the Namespace Identifier, and <NSS> is the Namespace Specific String. The leading "urn:" sequence is case-insensitive. The Namespace ID determines the _syntactic_ interpretation of the Namespace Specific String (as discussed in [1]).

RFC 1630 [2] and RFC 1737 [3] each presents additional considerations for URN encoding, which have implications as far as limiting syntax. On the other hand, the requirement to support existing legacy naming systems has the effect of broadening syntax. Thus, we discuss the acceptable syntax for both the Namespace Identifier and the Namespace Specific String separately.

2.1 Namespace Identifier Syntax

The following is the syntax for the Namespace Identifier. To (a) be consistent with all potential resolution schemes and (b) not put any undue constraints on any potential resolution scheme, the syntax for the Namespace Identifier is:

```
<NID>         ::= <let-num> [ 1,31<let-num-hyp> ]

<let-num-hyp> ::= <upper> | <lower> | <number> | "-"

<let-num>     ::= <upper> | <lower> | <number>

<upper>       ::= "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"

<lower>       ::= "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"

<number>      ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                  "8" | "9"
```

This is slightly more restrictive that what is stated in [4] (which allows the characters "." and "+"). Further, the Namespace Identifier is case insensitive, so that "ISBN" and "isbn" refer to the same namespace.

To avoid confusion with the "urn:" identifier, the NID "urn" is reserved and MUST NOT be used.

2.2 Namespace Specific String Syntax

As required by RFC 1737, there is a single canonical representation of the NSS portion of an URN. The format of this single canonical form follows:

```
<NSS>         ::= 1*<URN chars>

<URN chars>   ::= <trans> | "%" <hex> <hex>

<trans>       ::= <upper> | <lower> | <number> | <other> | <reserved>

<hex>         ::= <number> | "A" | "B" | "C" | "D" | "E" | "F" |
                  "a" | "b" | "c" | "d" | "e" | "f"

<other>       ::= "(" | ")" | "+" | "," | "-" | "." |
                  ":" | "=" | "@" | ";" | "$" |
                  "_" | "!" | "*" | "'"
```

Depending on the rules governing a namespace, valid identifiers in a namespace might contain characters that are not members of the URN character set above (<URN chars>). Such strings MUST be translated into canonical NSS format before using them as protocol elements or otherwise passing them on to other applications. Translation is done by encoding each character outside the URN character set as a sequence of one to six octets using UTF-8 encoding [5], and the encoding of each of those octets as "%" followed by two characters from the <hex> character set above. The two characters give the hexadecimal representation of that octet.

2.3 Reserved characters

The remaining character set left to be discussed above is the reserved character set, which contains various characters reserved from normal use. The reserved character set follows, with a discussion on the specifics of why each character is reserved.

The reserved character set is:

```
<reserved>    ::= '%" | "/" | "?" | "#"
```

2.3.1 The "%" character

The "%" character is reserved in the URN syntax for introducing the escape sequence for an octet. Literal use of the "%" character in a namespace must be encoded using "%25" in URNs for that namespace. The presence of an "%" character in an URN MUST be followed by two characters from the <hex> character set.

Namespaces MAY designate one or more characters from the URN character set as having special meaning for that namespace. If the namespace also uses that character in a literal sense as well, the character used in a literal sense MUST be encoded with "%" followed by the hexadecimal representation of that octet. Further, a character MUST NOT be "%"-encoded if the character is not a reserved character. Therefore, the process of registering a namespace identifier shall include publication of a definition of which characters have a special meaning to that namespace.

2.3.2 The other reserved characters

RFC 1630 [2] reserves the characters "/", "?", and "#" for particular purposes. The URN-WG has not yet debated the applicability and precise semantics of those purposes as applied to URNs. Therefore, these characters are RESERVED for future developments. Namespace developers SHOULD NOT use these characters in unencoded form, but rather use the appropriate %-encoding for each character.

2.4 Excluded characters

The following list is included only for the sake of completeness. Any octets/characters on this list are explicitly NOT part of the URN character set, and if used in an URN, MUST be %encoded:

```
<excluded> ::= octets 1-32 (1-20 hex) | "\" | """ | "&" | "<"
               | ">" | "[" | "]" | "^" | "`" | "{" | "|" | "}" | "~"
               | octets 127-255 (7F-FF hex)
```

In addition, octet 0 (0 hex) should NEVER be used, in either unencoded or %-encoded form.

An URN ends when an octet/character from the excluded character set (<excluded>) is encountered. The character from the excluded character set is NOT part of the URN.

3. Support of existing legacy naming systems and new naming systems

Any namespace (existing or newly-devised) that is proposed as an URN-namespace and fulfills the criteria of URN-namespaces MUST be expressed in this syntax. If names in these namespaces contain characters other than those defined for the URN character set, they MUST be translated into canonical form as discussed in section 2.2.

4. URN presentation and transport

The URN syntax defines the canonical format for URNs and all URN transport and interchanges MUST take place in this format. Further, all URN-aware applications MUST offer the option of displaying URNs in this canonical form to allow for direct transcription (for example by cut and paste techniques). Such applications MAY support display of URNs in a more human-friendly form and may use a character set that includes characters that aren't permitted in URN syntax as defined in this RFC (that is, they may replace %-notation by characters in some extended character set in display to humans).

5. Lexical Equivalence in URNs

For various purposes such as caching, it's often desirable to determine if two URNs are the same without resolving them. The general purpose means of doing so is by testing for "lexical equivalence" as defined below.

Two URNs are lexically equivalent if they are octet-by-octet equal after the following preprocessing:

1. normalize the case of the leading "urn:" token
2. normalize the case of the NID
3. normalizing the case of any %-escaping

Note that %-escaping MUST NOT be removed.

Some namespaces may define additional lexical equivalences, such as case-insensitivity of the NSS (or parts thereof). Additional lexical equivalences MUST be documented as part of namespace registration, MUST always have the effect of eliminating some of the false negatives obtained by the procedure above, and MUST NEVER say that two URNs are not equivalent if the procedure above says they are equivalent.

6. Examples of lexical equivalence

The following URN comparisons highlight the lexical equivalence definitions:

1- URN:foo:a123,456
2- urn:foo:a123,456
3- urn:FOO:a123,456
4- urn:foo:A123,456
5- urn:foo:a123%2C456
6- URN:FOO:a123%2c456

URNs 1, 2, and 3 are all lexically equivalent. URN 4 is not lexically equivalent any of the other URNs of the above set. URNs 5 and 6 are only lexically equivalent to each other.

7. Functional Equivalence in URNs

Functional equivalence is determined by practice within a given namespace and managed by resolvers for that namespeace. Thus, it is beyond the scope of this document. Namespace registration must include guidance on how to determine functional equivalence for that namespace, i.e. when two URNs are the identical within a namespace.

8. Security considerations

This document specifies the syntax for URNs. While some namespaces resolvers may assign special meaning to certain of the characters of the Namespace Specific String, any security consideration resulting from such assignment are outside the scope of this document. It is strongly recommended that the process of registering a namespace identifier include any such considerations.

9. Acknowledgments

Thanks to various members of the URN working group for comments on earlier drafts of this document. This document is partially supported by the National Science Foundation, Cooperative Agreement NCR-9218179.

10. References

Request For Comments (RFC) and Internet Draft documents are available from <URL:ftp://ftp.internic.net> and numerous mirror sites.

[1] Sollins, K. R., "Requirements and a Framework for URN Resolution Systems," Work in Progress.

[2] Berners-Lee, T., "Universal Resource Identifiers in WWW," RFC 1630, June 1994.

[3] Sollins, K. and L. Masinter, "Functional Requirements for Uniform Resource Names," RFC 1737. December 1994.

[4] Berners-Lee, T., R. Fielding, L. Masinter, "Uniform Resource Locators (URL)," Work in Progress.

[5] Appendix A.2 of The Unicode Consortium, "The Unicode Standard, Version 2.0", Addison-Wesley Developers Press, 1996. ISBN 0-201-48345-9.

11. Editor's address

Ryan Moats
AT&T
15621 Drexel Circle
Omaha, NE 68135-2358
USA

Phone: +1 402 894-9456
EMail: jayhawk@ds.internic.net

Appendix A. Handling of URNs by URL resolvers/browsers.

The URN syntax has been defined so that URNs can be used in places where URLs are expected. A resolver that conforms to the current URL syntax specification [3] will extract a scheme value of "urn:" rather than a scheme value of "urn:<nid>".

An URN MUST be considered an opaque URL by URL resolvers and passed (with the "urn:" tag) to an URN resolver for resolution. The URN resolver can either be an external resolver that the URL resolver knows of, or it can be functionality built-in to the URL resolver.

To avoid confusion of users, an URL browser SHOULD display the complete URN (including the "urn:" tag) to ensure that there is no confusion between URN namespace identifiers and URL scheme identifiers.