Internationalized Resource Identifiers (iri)
Internet-Draft
Intended status: Standards Track
Expires: February 13, 2012

L. Masinter
Adobe
M.J. Duerst
Aoyama Gakuin University

August 12, 2011

Equivalence and Canonicalization of Internationalized Resource Identifiers (IRIs)

draft-masinter-iri-comparison-00

Abstract

Internationalized Resource Identifiers (IRIs) are unicode strings used to identify resources on the Internet. Applications that use IRIs often define a means of comparing two IRIs to determine when two IRIs are equivalent for the purpose of that application. Some applications also define a method for 'canonicalizing' or 'normalizing' an IRI -- translating one IRI into another which is equivalent under the comparison method used.
This document gives guidelines and best practices for defining and using IRI comparison, equivalence, normalization and canonicalization methods.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Internationalized Resource Identifiers (IRIs) are unicode strings used to identify resources on the Internet. Applications that use IRIs often define a means of comparing two IRIs to determine when two IRIs are equivalent for the purpose of that application. Some applications also define a method for 'canonicalizing' or 'normalizing' an IRI -- translating one IRI into another which is equivalent under the comparison method used.
This document gives guidelines and best practices for defining and using IRI comparison, equivalence, normalization and canonicalization methods.

Note: The structure and much of the material for this document was originally taken from section 6 of [RFC3986].

One of the most common operations on IRIs is simple comparison: Determining whether two IRIs are equivalent, without using the IRIs to access their respective resource(s). A comparison is performed whenever a response cache is accessed, a browser checks its history to color a link, or an XML parser processes tags within a namespace. Extensive normalization prior to comparison of IRIs may be used by spiders and indexing engines to prune a search space or reduce duplication of request actions and response storage.
IRI comparison is performed for some particular purpose. Protocols or implementations that compare IRIs for different purposes will often be subject to differing design trade-offs in regards to how much effort should be spent in reducing aliased identifiers. This document describes various methods that may be used to compare IRIs, the trade-offs between them, and the types of applications that might use them.

2. Equivalence

Because IRIs exist to identify resources, presumably they should be considered equivalent when they identify the same resource. However, this definition of equivalence is not of much practical use, as there is no way for an implementation to compare two resources to determine if they are "the same" unless it has full knowledge or control of them. For this reason, determination of equivalence or difference of IRIs is based on string comparison, perhaps augmented by reference to additional rules provided by URI scheme definitions. We use the terms "different" and "equivalent" to describe the possible outcomes of such comparisons, but there are many application-dependent versions of equivalence.
Even when it is possible to determine that two IRIs are equivalent, IRI comparison is not sufficient to determine whether two IRIs identify different resources. For example, an owner of two different domain names could decide to serve the same resource from both, resulting in

two different IRIs. Therefore, comparison methods are designed to minimize false negatives while strictly avoiding false positives.
In testing for equivalence, applications should not directly compare relative references; the references should be converted to their respective target IRIs before comparison. When IRIs are compared to select (or avoid) a network action, such as retrieval of a representation, fragment components (if any) MUST be excluded from the comparison.
Applications using IRIs as identity tokens with no relationship to a protocol MUST use the Simple String Comparison (see Section 4.1). All other applications MUST select one of the comparison practices from the Comparison Ladder (see Section 4.

3. Preparation for Comparison

Any kind of IRI comparison REQUIRES that any additional contextual processing is first performed, including undoing higher-level escapings or encodings in the protocol or format that carries an IRI. This preprocessing is usually done when the protocol or format is parsed. Examples of such escapings or encodings are entities and numeric character references in [HTML4] and [XML1]. As an example, "http://example.org/rosé" (in HTML), "http://example.org/rosé" (in HTML or XML), and
"http://example.org/rosé" (in HTML or XML) are all resolved into what is denoted in this document (see 'Notation' section of [RFC3987bis]) as "http://example.org/rosé" (the "é" here standing for the actual e-acute character, to compensate for the fact that this document cannot contain non-ASCII characters).
Similar considerations apply to encodings such as Transfer Codings in HTTP (see [RFC2616]) and Content Transfer Encodings in MIME ([RFC2045]), although in these cases, the encoding is based not on characters but on octets, and additional care is required to make sure that characters, and not just arbitrary octets, are compared (see Section 4.1).

4. Comparison Ladder

In practice, a variety of methods are used to test IRI equivalence. These methods fall into a range distinguished by the amount of processing required and the degree to which the probability of false negatives is reduced. As noted above, false negatives cannot be eliminated. In practice, their probability can be reduced, but this reduction requires more processing and is not cost-effective for all applications.
If this range of comparison practices is considered as a ladder, the following discussion will climb the ladder, starting with practices that are cheap but have a relatively higher chance of producing false negatives, and proceeding to those that have higher computational cost and lower risk of false negatives.

4.1. Simple String Comparison

If two IRIs, when considered as character strings, are identical, then it is safe to conclude that they are equivalent. This type of equivalence test has very low computational cost and is in wide use in a variety of applications, particularly in the domain of parsing. It is also used when a definitive answer to the question of IRI equivalence is needed that is independent of the scheme used and that can be calculated quickly and without accessing a network. An example of such a case is XML Namespaces ([XMLNamespace]).

Testing strings for equivalence requires some basic precautions. This procedure is often referred to as "bit-for-bit" or "byte-for-byte" comparison, which is potentially misleading. Testing strings for equality is normally based on pair comparison of the characters that make up the strings, starting from the first and proceeding until both strings are exhausted and all characters are found to be equal, until a pair of characters compares unequal, or until one of the strings is exhausted before the other.

This character comparison requires that each pair of characters be put in comparable encoding form. For example, should one IRI be stored in a byte array in UTF-8 encoding form and the second in a UTF-16 encoding form, bit-for-bit comparisons applied naively will produce errors. It is better to speak of equality on a character-for-character rather than on a byte-for-byte or bit-for-bit basis. In practical terms, character-by-character comparisons should be done codepoint by codepoint after conversion to a common character encoding form. When comparing character by character, the comparison function MUST NOT map IRIs to URIs, because such a mapping would create additional spurious equivalences. It follows that an IRI SHOULD NOT be modified when being transported if there is any chance that this IRI might be used in a context that uses Simple String Comparison.

False negatives are caused by the production and use of IRI aliases. Unnecessary aliases can be reduced, regardless of the comparison method, by consistently providing IRI references in an already normalized form (i.e., a form identical to what would be produced after normalization is applied, as described below). Protocols and data formats often limit some IRI comparisons to simple string comparison, based on the theory that people and implementations will, in their own best interest, be consistent in providing IRI references, or at least be consistent enough to negate any efficiency that might be obtained from further normalization.

4.2. Syntax-Based Normalization

Implementations may use logic based on the definitions provided by this specification to reduce the probability of false negatives. This processing is moderately higher in cost than character-for-character string comparison. For example, an application using this approach could reasonably consider the following two IRIs equivalent:

example://a/b/c/%7Bfoo%7D/rosé
eXAMPLE://a/./b/../b/%63/%7bfoo%7d/ros%C3%A9

Web user agents, such as browsers, typically apply this type of IRI normalization when determining whether a cached response is available. Syntax-based normalization includes such techniques as case normalization, character normalization, percent-encoding normalization, and removal of dot-segments.

4.2.1. Case Normalization

For all IRIs, the hexadecimal digits within a percent-encoding triplet (e.g., "%3a" versus "%3A") are case-insensitive and therefore should be normalized to use uppercase letters for the digits A-F.
When an IRI uses components of the generic syntax, the component syntax equivalence rules always apply; namely, that the scheme and US-ASCII only host are case insensitive and therefore should be normalized to lowercase. For example, the URI "HTTP://www.EXAMPLE.com/" is equivalent to "http://www.example.com/". Case equivalence for non-ASCII characters in IRI components that are IDNs are discussed in Section 4.3. The other generic syntax components are assumed to be case sensitive unless specifically defined otherwise by the scheme.
Creating schemes that allow case-insensitive syntax components containing non-ASCII characters should be avoided. Case normalization of non-ASCII characters can be culturally dependent and is always a complex operation. The only exception concerns non-ASCII host names for which the character normalization includes a mapping step derived from case folding.

4.2.2. Character Normalization

The Unicode Standard [UNIV6] defines various equivalences between sequences of characters for various purposes. Unicode Standard Annex #15 [UTR15] defines various Normalization Forms for these equivalences, in particular Normalization Form C (NFC, Canonical Decomposition, followed by Canonical Composition) and Normalization Form KC (NFKC, Compatibility Decomposition, followed by Canonical Composition).
IRIs already in Unicode MUST NOT be normalized before parsing or interpreting. In many non-Unicode character encodings, some text cannot be represented directly. For example, the word "Vietnam" is natively written "Việt Nam" (containing a LATIN SMALL LETTER E WITH CIRCUMFLEX AND DOT BELOW) in NFC, but a direct transcoding from the windows-1258 character encoding leads to "Việt Nam" (containing a LATIN SMALL LETTER E WITH CIRCUMFLEX followed by a COMBINING DOT BELOW). Direct transcoding of other 8-bit encodings of Vietnamese may lead to other representations.
Equivalence of IRIs MUST rely on the assumption that IRIs are appropriately pre-character-normalized rather than apply character normalization when comparing two IRIs. The exceptions are conversion

from a non-digital form, and conversion from a non-UCS-based character encoding to a UCS-based character encoding. In these cases, NFC or a normalizing transcoder using NFC MUST be used for interoperability. To avoid false negatives and problems with transcoding, IRIs SHOULD be created by using NFC. Using NFKC may avoid even more problems; for example, by choosing half-width Latin letters instead of full-width ones, and full-width instead of half-width Katakana.
As an example, "http://www.example.org/résumé.html" (in XML Notation) is in NFC. On the other hand, "http://www.example.org/résumé.html" is not in NFC.
The former uses precombined e-acute characters, and the latter uses "e" characters followed by combining acute accents. Both usages are defined as canonically equivalent in [UNIV6].

Note: Because it is unknown how a particular sequence of characters is being treated with respect to character normalization, it would be inappropriate to allow third parties to normalize an IRI arbitrarily. This does not contradict the recommendation that when a resource is created, its IRI should be as character normalized as possible (i.e., NFC or even NFKC). This is similar to the uppercase/lowercase problems. Some parts of a URI are case insensitive (for example, the domain name). For others, it is unclear whether they are case sensitive, case insensitive, or something in between (e.g., case sensitive, but with a multiple choice selection if the wrong case is used, instead of a direct negative result). The best recipe is that the creator use a reasonable capitalization and, when transferring the URI, capitalization never be changed.

Various IRI schemes may allow the usage of Internationalized Domain Names (IDN) [RFC5890] either in the ireg-name part or elsewhere. Character Normalization also applies to IDNs, as discussed in Section 4.3.

4.2.3. Percent-Encoding Normalization

The percent-encoding mechanism (Section 2.1 of [RFC3986]) is a frequent source of variance among otherwise identical IRIs. In addition to the case normalization issue noted above, some IRI producers percent-encode octets that do not require percent-encoding, resulting in IRIs that are equivalent to their nonencoded counterparts. These IRIs should be normalized by decoding any percent-encoded octet sequence that corresponds to an unreserved character, as described in section 2.3 of [RFC3986].
For actual resolution, differences in percent-encoding (except for the percent-encoding of reserved characters) MUST always result in the same resource. For example, "http://example.org/~user", "http://example.org/%7euser", and "http://example.org/%7Euser", must resolve to the same resource.

If this kind of equivalence is to be tested, the percent-encoding of both IRIs to be compared has to be aligned; for example, by converting both IRIs to URIs (see Section 3.1), eliminating escape differences in the resulting URIs, and making sure that the case of the hexadecimal characters in the percent-encoding is always the same (preferably upper case). If the IRI is to be passed to another application or used further in some other way, its original form MUST be preserved. The conversion described here should be performed only for local comparison.

4.2.4. Path Segment Normalization

The complete path segments "." and ".." are intended only for use within relative references (Section 4.1 of [RFC3986]) and are removed as part of the reference resolution process (Section 5.2 of [RFC3986]). However, some implementations may incorrectly assume that reference resolution is not necessary when the reference is already an IRI, and thus fail to remove dot-segments when they occur in non-relative paths. IRI normalizers should remove dot-segments by applying the remove_dot_segments algorithm to the path, as described in Section 5.2.4 of [RFC3986].

4.3. Scheme-Based Normalization

The syntax and semantics of IRIs vary from scheme to scheme, as described by the defining specification for each scheme. Implementations may use scheme-specific rules, at further processing cost, to reduce the probability of false negatives. For example, because the "http" scheme makes use of an authority component, has a default port of "80", and defines an empty path to be equivalent to "/", the following four IRIs are equivalent:

```
http://example.com
http://example.com/
http://example.com:/
http://example.com:80/
```

In general, an IRI that uses the generic syntax for authority with an empty path should be normalized to a path of "/". Likewise, an explicit ":port", for which the port is empty or the default for the scheme, is equivalent to one where the port and its ":" delimiter are elided and thus should be removed by scheme-based normalization. For example, the second IRI above is the normal form for the "http" scheme.
Another case where normalization varies by scheme is in the handling of an empty authority component or empty host subcomponent. For many scheme specifications, an empty authority or host is considered an error; for others, it is considered equivalent to "localhost" or the end-user's host. When a scheme defines a default for authority and an IRI reference to that default is desired, the reference should be

normalized to an empty authority for the sake of uniformity, brevity, and internationalization. If, however, either the userinfo or port subcomponents are non-empty, then the host should be given explicitly even if it matches the default.
Normalization should not remove delimiters when their associated component is empty unless it is licensed to do so by the scheme specification. For example, the IRI "http://example.com/?" cannot be assumed to be equivalent to any of the examples above. Likewise, the presence or absence of delimiters within a userinfo subcomponent is usually significant to its interpretation. The fragment component is not subject to any scheme-based normalization; thus, two IRIs that differ only by the suffix "#" are considered different regardless of the scheme.
Some IRI schemes allow the usage of Internationalized Domain Names (IDN) [RFC5890] either in their ireg-name part or elswhere. When in use in IRIs, those names SHOULD conform to the definition of U-Label in [RFC5890]. An IRI containing an invalid IDN cannot successfully be resolved. For legibility purposes, they SHOULD NOT be converted into ASCII Compatible Encoding (ACE).
Scheme-based normalization may also consider IDN components and their conversions to punycode as equivalent. As an example, "http://résumé.example.org" may be considered equivalent to "http://xn--rsum-bpad.example.org".
Other scheme-specific normalizations are possible.

4.4. Protocol-Based Normalization

Substantial effort to reduce the incidence of false negatives is often cost-effective for web spiders. Consequently, they implement even more aggressive techniques in IRI comparison. For example, if they observe that an IRI such as

   http://example.com/data

redirects to an IRI differing only in the trailing slash

   http://example.com/data/

they will likely regard the two as equivalent in the future. This kind of technique is only appropriate when equivalence is clearly indicated by both the result of accessing the resources and the common conventions of their scheme's dereference algorithm (in this case, use of redirection by HTTP origin servers to avoid problems with relative references).

5. Security Considerations

The primary security difficulty comes from applications choosing the wrong equivalence relationship, or two different parties disagreeing on

equivalence. This is especially a problem when IRIs are used in security protocols.
Besides the large character repertoire of Unicode, reasons for confusion include different forms of normalization and different normalization expectations, use of percent-encoding with various legacy encodings, and bidirectionality issues. See also [UTR36].

6. References

6.1. Normative References

[RFC3987bis] Duerst, M., Masinter, L. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3490] Faltstrom, P., Hoffman, P. and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[UNIV6] The Unicode Consortium, "The Unicode Standard, Version 6.0.0 (Mountain View, CA, The Unicode Consortium, 2011, ISBN 978-1-936213-01-6)", October 2010.

[UTR15] Davis, M. and M.J. Duerst, "Unicode Normalization Forms", Unicode Standard Annex #15, March 2008.

6.2. Informative References

[HTML4] Raggett, D., Le Hors, A. and I. Jacobs, "HTML 4.01 Specification", World Wide Web Consortium Recommendation, December 1999.

[RFC2045] Freed, N. and N.S. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
[UTR36]	Davis, M. and M. Suignard, "Unicode Security Considerations", Unicode Technical Report #36, August 2010.
[XML1]	Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E. and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Forth Edition)", World Wide Web Consortium Recommendation, August 2006.
[XMLNamespace]	Bray, T., Hollander, D., Layman, A. and R. Tobin, "Namespaces in XML (Second Edition)", World Wide Web Consortium Recommendation, August 2006.

Authors' Addresses

Larry Masinter Masinter Adobe
345 Park Ave San Jose, CA 95110 U.S.A. Phone: +1-408-536-3024 EMail: masinter@adobe.com URI: http://larry.masinter.net

Martin Duerst Duerst Aoyama Gakuin University 5-10-1 Fuchinobe Sagamihara, Kanagawa 229-8558 Japan Phone: +81 42 759 6329 EMail: duerst@it.aoyama.ac.jp URI: http://www.sw.it.aoyama.ac.jp/D%C3%BCrst/