INTERNET-DRAFT

Larry Masinter
Adobe Systems Incorporated
Martin Duerst
W3C/Keio University

draft-masinter-url-i18n-08.txt
Expires May 2002

November 20, 2001

Internationalized Resource Identifiers (IRI)

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This document is not a product of any working group, but may be discussed on the mailing list <uri@w3.org>. For more information on the topic of this internet-draft, please also see [W3C IRI].

Abstract

This document defines a new protocol element, an Internationalized Resource Identifier (IRI). An IRI is a sequence of characters from the Universal Character Set [10646]. A mapping from IRIs to URIs [RFC 2396] is defined, which means that IRIs can be used instead of URIs where appropriate to identify resources.

Defining a new protocol element was preferred to extending or changing the definition of URIs to allow a clear distinction and to avoid incompatibilities with existing software. Guidelines for the use and deployment of IRIs in various protocols, formats, and software components that now deal with URIs are provided.

0. Change log

0.7 Changes from the -07 version

- Added applicability statement
- Allowed space and a few other characters in IRIs to be consistent
  with XML, XLink, XML Schema,... (and added some heavy warning).
- Changed the start of non-ASCII characters from U+0080 to U+00A0 to
  exclude C1 control characters. Changes to various sections.
- Fixed several problems in the IRI syntax
- Added wording in various places to give more prominence to
  URI references/IRI references.
- Added some text about pure data-based URIs
- Changed IRI-URI mapping to distinguish cases where input is already
  in UCS
- Removed section about bidirectionality, pointing to
  draft-duerst-iri-bidi-00.txt
- Rewrote section 2.4, adding a reference to [Duer97]
- Generalized section 3.3
- Added pointers to [UNIXML], [Duer01], [XLink], [XMLSchema]
- Made adoption of IRIs by W3C clearer
- Mentioned 'conversion as late as possible' principle
- Moved from key input to general input
- Various wording charges and fixes

1. Introduction

1.1 Overview and Motivation

A URI is defined in [RFC2326] as a sequence of characters chosen from a limited subset of the repertoire of US-ASCII characters. This document defines a new protocol element, called IRI (Internationalized Resource Identifier), by extending the syntax of URIs to a much wider repertoire of characters. It also defines "internationalized" versions corresponding to other constructs from [RFC2326], such as URI references.

The characters in URIs are frequently used for representing words of natural languages. Using words from natural languages in URIs is very common. Such usage has many advantages: such URIs are easier to memorize, easier to interpret, easier to transcribe, easier to create, and easier to guess. For most languages other than English, however, the natural script uses characters other than A-Z. For many people, handling Latin characters is as difficult as handling the characters of other scripts is for people who use only the Latin alphabet. Many languages with non-Latin scripts do have transcriptions to Latin letters and such transcriptions are now often used in URIs, but they introduce additional ambiguities.

The infrastructure for the appropriate handling of characters from local scripts is now widely deployed in local versions of operating system and application software. Software that can handle a wide variety of scripts and languages at the same time is getting more

and more widespread. Also, there are more and more protocols and formats that can carry a wide range of characters.

Using characters outside of A-Z in IRIs brings with it some difficulties; a discussion of potential problems and workarounds can be found in the later sections of this document.

URIs often contain Internet host names embedded with them. There is an ongoing discussion of internationalization and host names; the specific issues of the relationship of IRIs and possible future "internationalized" host names are not discussed here. See [IDN-URI] for a separate proposal.

1.2 Applicability

IRIs are designed to work together with recent recommendations on URI syntax [RFC 2718]. In order to be able to use an IRI (or IRI reference in place of an URI (or URI reference) in a given protocol context, the following conditions have to be met:

a. The protocol or format carrying the IRI has to be able to represent the non-ASCII characters in the IRI, either natively or by some protocol- or format-specific escaping mechanism (e.g. numeric character references in [XML1]).

b. The protocol or format element used has to have been designated to carry IRIs (e.g. by designating it to be of type anyURI in [XMLSchema]).

c. In the URI scheme, or at least the actual URI in question (as recommended for new schemes in [RFC 2718]), the encoding of non-ASCII characters has to be based on UTF-8. This allows IRIs to be used with the URN syntax [RFC2141] as well as recent URL scheme definitions based on UTF-8, such as IMAP URLs [RFC 2192] and POP URLs [RFC 2384]). This condition may also apply to only a piece of an URI (reference), such as the fragment identifier.

   In cases and for pieces where another encoding than UTF-8 is used, and for raw binary data encoded in URIs (see [RFC 2397]), the functionalities of IRIs cannot be used.

Section 2 discusses the IRI syntax and conversion between IRIs and URIs. Limitations on characters appropriate for use in IRIs and processing of IRIs are dealt with in Section 3. Section 4 discusses software requirements for IRIs from an operational viewpoint. For additional discussion and examples, please see also [Duer01].

1.3 Definitions

The following definitions are used in this document; they follow the terms in [RFC 2130] and [RFC2277]:

character	An abstract object with a separate identity. "LATIN CAPITAL LETTER A" is a character.
octet	8 bits
character repertoire	A set of characters (in the Mathematical sense)
sequence of characters	A sequence (one after another) of characters
sequence of octets	A sequence (one after another) of octets
(character) encoding	A method of representing a sequence of characters as a sequence of octets (maybe with variants). A method of (unambiguously!) converting a sequence of octets into a sequence of characters.
code point	A placeholder for a character in a character encoding, for example to encode additional characters in future versions of the character encoding.
charset	The name of a parameter or attribute used to identify a character encoding.

2. IRI Syntax

This section defines the syntax of Internationalized Resource Identifiers (IRIs).

As with URIs, an IRI is defined as a sequence of characters. This definition accommodates the fact that IRIs may be written on paper or read over the radio as well as being transmitted over the network. Also, the same IRI may be represented as different sequences of octets in different protocols or documents if these protocols or documents use different character encodings. Using the same character encoding as the containing protocol or document assures that the characters in the IRI can be handled (searched, converted, displayed,...) in the same way as the rest of the protocol or document.

2.1 Summary of IRI syntax

IRIs are defined similarly to URIs in [RFC2386] (as modified by [RFC2732]), but the class of unreserved characters is extended by adding all the characters of the UCS (Universal Character Set, [ISO10646]) beyond U+0080, subject to the limitations given in Section 3.

Otherwise, the syntax and use of components and reserved characters is the same as that in [RFC2396]. All the operations defined in [RFC2396], such as the resolution of relative URIs, can be applied to IRIs by IRI-processing software exactly in the same way as this is done by URI-processing software.

Characters outside the ASCII range MUST NOT be used for syntactical

purposes such as to delimit components in newly defined schemes. As an example, it is not allowed to use U+00A2, CENT SIGN, as a delimiter, because it is in the 'iunreserved' category, in the same way as it is not possible to use '-' as a delimiter, because it is in the 'unreserved' category.

2.2 ABNF for IRI References and IRIs

While it might be possible to define IRI references and IRIs merely by their transformation to URIs, IRI references and IRIs can also be accepted and processed directly. Therefore, an ABNF definition for IRI references and IRIs is given here.

The following are different form [RFC2396]:

```
       IRI-reference  = [ absoluteIRI | relativeIRI ] [ "#" ifragment ]
       absoluteIRI    = scheme ":" ( ihier_part | iopaque_part )
       relativeIRI    = ( inet_path | iabs_path | irel_path )
                        [ "?" iquery ]
       ihier_part     = ( inet_path | iabs_path ) [ "?" iquery ]
       iopaque_part   = iric_no_slash *iric
       iric_no_slash  = iunreserved | escaped | ";" | "?" | ":" | "@" |
                        "&" | "=" | "+" | "$" | ","
       inet_path      = "//" iauthority [ iabs_path ]
       iabs_path      = "/"  ipath_segments
       irel_path      = irel_segment [ iabs_path ]
       irel_segment   = 1*( iunreserved | escaped |
                            ";" | "@" | "&" | "=" | "+" | "$" | "," )
       iauthority     = server | ireg_name
       ireg_name      = 1*( iunreserved | escaped | "$" | "," |
                            ";" | ":" | "@" | "&" | "=" | "+" )
       ipath_segments = isegment *( "/" isegment )
       isegment       = *ipchar *( ";" iparam )
       iparam         = *ipchar
       ipchar         = iunreserved | escaped |
                         ":" | "@" | "&" | "=" | "+" | "$" | ","
       iquery         = *iric
       ifragment      = *iric
       iric           = reserved | iunreserved | escaped
       iunreserved    = ichar | unreserved
       ichar          = << any character of the UCS [ISO10646] of U+00A0
                           and beyond, subject to limitations in Section
                            3.1. >> | space | delims | unwise
```

Please note that the space character and various delimiters are allowed in IRIs and IRI references. This is further discussed in section 3.1, point b.

The following are the same as [RFC2396] as modified by [RFC2732]:

```
       reserved    = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+" |
```

```
                "$" | "," | "[" | "]"
    unreserved    = alphanum | mark
    mark          = "-" | "_" | "." | "!" | "~" | "*" | "'" |
                    "(" | ")"
    escaped       = "%" HEXDIG HEXDIG
    server        = [ [ userinfo "@" ] hostport ]
    userinfo      = *( unreserved | escaped |
                       ";" | ":" | "&" | "=" | "+" | "$" | "," )
    hostport      = host [ ":" port ]
    host          = hostname | IPv4address | IPv6reference
    ipv6reference = "[" IPv6address "]"
    hostname      = *( domainlabel "." ) toplabel [ "." ]
    domainlabel   = alphanum | alphanum *( alphanum | "-" ) alphanum
    toplabel      = alpha | alpha *( alphanum | "-" ) alphanum
    IPv6address   = hexpart [ ":" IPv4address ]
    IPv4address   = 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
    IPv6prefix    = hexpart "/" 1*2DIGIT
    hexpart       = hexseq | hexseq "::" [ hexseq ] | "::"
                    [ hexseq ]
    hexseq        = hex4 *( ":" hex4)
    hex4          = 1*4HEXDIG
    port          = *DIGIT

    scheme        = alpha *( alpha | digit | "+" | "-" | "." )
    alphanum      = alpha | digit
    alpha         = lowalpha | upalpha
    lowalpha = "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"
    upalpha  = "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"
    space    = <US-ASCII coded character 20 hexadecimal>
    delims   = "<" | ">" | "#" | "%" | <">
    unwise   = "{" | "}" | "|" | "\" | "^" | "[" | "]" | "`"
```

2.3 Mapping of IRIs to URIs

This section defines how to map an IRI to a URI. Everything in this section applies also to IRI references and URI references, as well as components thereoff (e.g. fragment identifiers).

This mapping has two purposes:

a) Syntactical: Many URI schemes and components define additional syntactical restrictions not captured in Section 2.2. Such restrictions can be applied to IRIs by noting that IRIs are only valid if they map to syntactically valid URIs. This means that such syntactical restrictions do not have to be defined again on the IRI level.

b) Interpretational: URIs identify resources in various ways. IRIs also indentify resources. The resource that an IRI identifies is the same as the one identified by the URI obtained after converting the IRI according to the procedure defined here. This means that there is no need to define the association between identifier and resource again on the IRI level.

This mapping is accomplished in two parts. Part I is skipped if the input is already in an UCS-based encoding (e.g. UTF-8 or UTF-16). In that case, it is assumed that the IRI is already in NFC.

Part I)

1) Represent the IRI characters as a sequence of characters from the UCS.

2) Normalize the character sequence according to Normalization Form C, as defined in [UNI15]. (See further discussion in Section 3.1.)

Part II)

For each character that is disallowed in URI references, apply steps 3.1 through 3.3. The disallowed characters consist of all non-ASCII characters, plus the excluded characters listed in Section 2.4 of [RFC 2396], except for the number sign (#) and percent sign (%) and the square bracket characters re-allowed in [RFC 2732].

1) Convert the character to a sequence of one or more octets using UTF-8 [RFC 2279].

2) Convert each octet to %HH, where HH is the hexadecimal notation of the octet value. Note: This is identical to the escaping mechanism in Section 2.4.1 of [RFC 2396].

3) Replace the original character by the resulting character sequence.

In step 3) of part II), characters allowed in URI references, or octets already escaped, MUST NOT be escaped further, because they are already in their correct escaping stage in IRIs. This means that this mapping is similar to, but different from, the escaping applied when including arbitrary content into some part of an URI.

The above mapping produces an URI fully conforming to [RFC 2396] out of each IRI. In addition, it results in the identity transformation for URIs, i.e. applying the mapping a second time will not change anything anymore. Every URI is therefore by definition an IRI. However, this mapping SHOULD only be applied when necessary, as late as possible.

2.4 Converting URIs to IRIs

In some situations, it may be desirable to convert a URI into an equivalent IRI. This section gives a procedure to do such a conversion. However, it is important to note that it is not always possible to eliminate all escaping in an IRI. There are many reasons for this.

a. Some escape sequences (all below %80) are necessary to distinguish escaped and unescaped uses of reserved characters.

b. Some escape sequences cannot be interpreted as sequences of UTF-8 octets.
Note: Due to the regularities in the octet patterns of UTF-8, there is a very high probability, but no guarantee, that escape sequences that can be interpreted as sequences of UTF-8 octets actually originated from UTF-8. For a detailed discussion of the odds, see [Duer97].

c. The conversion may result in a character that is not appropriate. See section 3.1 for further details.

Conversion from an URI to an IRI is done using the following steps:

1) Convert all hexadecimal escapes (% followed by two hexadecimal digits) of %80 and higher to the corresponding octets. (The result is a sequence of intermixed 'characters' and 'octets'; it is very important to distinguish strictly between characters and octets in this procedure.)

2) Convert all sequences of octets that are strictly legal UTF-8 sequences to the corresponding sequences of characters based on UTF-8. Any subsequence that is found to correspond to a legal UTF-8 sequence can be converted. Note: The properties of UTF-8 make sure that this will in all cases lead to the same result.

3) Using UTF-8, convert back to octets all resulting characters that are identified as not appropriate according to Section 3.1.

4) Convert all the remaining octets (i.e. those not affected by step 2) and those produced by step 3) back to hexadecimal escapes.

This procedure will convert as many escaped non-ASCII characters as possible to characters in an IRI. Because there are some choices when applying step 3) (see Section 3.1), results may differ.

3. Considerations for use of IRIs

3.1 IRI Character Limitations

Not all characters of the UCS are appropriate for use as resource

identifiers. This section discusses the limitations on characters and character sequences usable for IRIs. The considerations in this section are relevant when creating IRIs and when converting from URIs to IRIs.

Because of the large and increasing number of characters in the UCS and the large number of situations where IRIs can be used, it is impossible to give general rules for which characters are allowed and which not. The following considerations are relevant:

a. The repertoire of characters allowed in each IRI component is limited by the definition of that component. For example, the definition of host names in URIs does not currently allow hex escapes, or "_", or many other punctuation characters. This specification does not relax those limits, and so IRIs currently may not contain any non-ASCII characters in host names. This specification likewise does not extended the scheme component beyond US-ASCII.

   Please note that in accordance with URI practice, generic IRI software cannot and should not check for such limitations.

b. In the URI syntax, characters that are likely to be used to delimit URIs in text and print ("space", "delims", and "unwise") were excluded. They are included in the IRI syntax, for the following reasons: 1) The syntax includes many other characters that are not appropriate in many cases; 2) Some implementation practice already allows them in URI references (e.g. spaces in fragment identifiers); 3) It is very convenient in some cases, e.g. for XPointers in XML attributes; and 4) Considering context is already necessary in the case of URIs, for example for "&" in XML.

   However, these characters should be used only with utmost care. Whenever there is a chance that an IRI will get used out of a context where these characters can be used without harm, they should be escaped from the start.

c. The UCS contains many areas of "characters" which have no simple way of inputing them. These should be avoided. Characters that fall into this category include Dingbats, Mathematical and other symbols, ligatures and presentation forms.

d. The UCS contains many areas of characters for which there are strong visual look-alikes. Because of the likelihood of transcription errors, these also should be avoided. This includes the full-width equivalents of ASCII characters, half-width Katakana characters for Japanese, and many others. This also includes many look-alikes of "space", "delims", and "unwise", characters excluded in [RFC 2396].

e. Characters with no visual representation may not be interoperably

entered. "Control characters" MUST NOT be used. This includes the traditional ranges of control characters (U+0000-U+001F and U+007F-U+009F) as well as other cases such as plane-14 language tag characters.

f. Some code points are reserved for private use or for special encoding purposes. They are not interoperable. Code points reserved for private use MUST NOT be used. Code points reserved for surrogates MUST NOT be used.

g. Where there exist duplicate ways of encoding a certain character as visible to the user, Normalization Form C as defined in [UNI15] MUST be used.

Additional information is available from [UNIXML]. Although this is written in a different context, it discusses many of the categories of characters and code points not appropriate for IRIs.

For reasons of transcribability, many characters have been excluded from IRIs above. These can nevertheless be encoded in an IRI if necessary. They have to be escaped using the procedure in Section 2.3. For example, a space can always be encoded in an URI and in an IRI as %20. A non-breaking space (U+00A0) has to be encoded as %C2%A0.

3.2 Bidirectional IRIs for Right-to-Left languages

Some UCS characters, such as those used in the Arabic and Hebrew script, have an inherent right-to-left writing direction. IRIs containing such characters, called bidirectional IRIs or Bidi IRIs) require additional attention because of the non-trivial relation between logical representation (used for digital representation as well as when reading/spelling) and visual representation (used for display/printing). This document does not address Bidi-specific issues. A proposal for addressing these issues can be found in [Bidi].

3.3. Processing IRIs

Processing of relative forms of IRIs against a base is handled straightforwardly; the algorithms of RFC 2396 may be applied directly, treating the characters additionally allowed in IRIs in the same way as unreserved characters in URIs. Other processing operations on IRIs and IRI references similarly work analogous to their URI complements.

Such processing and mapping to URIs is commutative, i.e. the same result is obtained independent of whether the processing or the mapping is done first. If both IRIs and URIs are involved in processing, the IRI parts SHOULD be preserved as long as possible. For example, it is possible to create an absolute IRI from a relative IRI and an URI base. When IRIs are compared, temporary mapping to URIs MAY be advisable to eliminate potential differences

in the degree of escaping.

4. Software requirements

This section explains the issues and difficulties in supporting IRIs in the same software components and operations that currently process URIs: software interfaces that handle URIs, software that allows users to enter URIs, software that generates URIs, software that displays URIs, formats and protocol that transport URIs, and software that interprets URIs, may all require more or less modification before functioning properly with IRIs. The considerations in this section also apply to URI references and IRI references.

4.1 URI/IRI software interfaces

Software interfaces that handle URIs, such as URI-handling APIs and protocols transferring URIs, need interfaces and protocol elements that are designed to carry IRIs.

Note that although an IRI is defined as a sequence of characters, software interfaces for URIs typically function on sequences of octets. Thus, it is necessary to define clearly which character encoding is used.

In case the current handling in an API or protocol is based on US-ASCII, UTF-8 is recommended as the encoding for IRIs, because this is compatible with US-ASCII, is in accordance with the recommendations of [RFC 2277], and makes it easy to convert to URIs where necessary. In any case, the encoding used must not be left undefined.

Intermediate software interfaces between IRI-capable components and URI-only components MUST map the IRIs as per section 2.3 above, when transferring from IRI-capable to URI-only components. However, such a mapping SHOULD be applied as late as possible. It should not be applied between components that are known to be able to handle IRIs.

The transfer from URI-only to IRI-capable components requires no mapping, although the conversion described in section 2.4 above may be performed. It is preferable not to perform this inverse conversion when there is a chance that this cannot be done correctly.

4.2 URI/IRI entry

There are components that allow users to enter URIs into the system, e.g., by typing or dictation. This software must be updated to allow for IRI entry.

A person viewing a visual representation of an IRI (as a sequence of glyphs, in some order, in some visual display) or hearing an IRI, will use a entry method for characters in that language to input

the IRI. Depending on the script and the input method used, this may be a more or less complicated process.

The process of IRI entry must assure, as far as possible, that the limitations defined in Section 3.1 are met. This may be done by choosing appropriate input methods or variants/settings thereof, by appropriately converting the characters being input, by eliminating characters that cannot be converted, and/or by issuing a warning or error message to the user.

An input field primarily or only used for the input of URIs/IRIs should allow the user to view an IRI as converted to an URI. Places where the input of IRIs is frequent should provide the possibility for viewing an IRI as converted to an URI. This will help users when some of the software they use does not yet accept IRIs.

An IRI input component that interfaces to components that handle URIs, but not IRIs, must escape the IRI before passing it to such a component.

The input of IRIs with right-to-left characters requires additional care to keep the visual and the internal representation in synch, and to eliminate control characters and marks used to control the display before passing the IRI over to a resolver. IRI input fields that allow the input of right-to-left characters MUST provide this functionality. IRI input fields that do not provide this functionality MUST NOT allow the input of right-to-left characters.

4.3 URI/IRI generation

Systems that are offering resources through the Internet, where those resources have logical names, sometimes automatically generate URIs for the resources they offer. For example, some HTTP servers can generate a 'directory listing' for a file directory under their purview, and then respond to the generated URIs with the files.

Many legacy character encodings are in use in various file systems. Many currently deployed systems do not transform the local character representation of the underlying system before generating URIs.

For maximum interoperability, systems that generate resource identifiers should do the appropriate transformations. They should use IRIs converted to URIs in cases where it cannot be expected that the recipient is able to handle IRIs. Due to the way most user agents currently work, native IRIs, encoded in UTF-8, may be used if the recipient announces that it can interpret UTF-8. This requires that the whole page is sent as UTF-8. If this is not possible, escaping can always be used.

This recommendation in particular applies to HTTP servers. For FTP servers, similar considerations apply, see in particular [RFC 2640].

4.4 URI/IRI selection

In some cases, resource owners and publishers have control over the IRIs used to identify their resources. Such control is mostly executed by controlling the resource names, such as file names, directly.

In such cases, it is recommended to avoid choosing IRIs that are easily confused. For example, for US-ASCII, the lower-case ell "l" is easily confused with the digit one "1", and the upper-case oh "O" is easily confused with the digit zero "0". Publishers should avoid to unintentionally confuse users with "br0ken" or "1ame" identifiers.

Outside of the US-ASCII range, there are many more opportunities for confusion; a complete set of guidelines is too lengthy to include here. As long as names are limited to characters from a single script, native writers of a given script or language will know best when ambiguities can appear, and how they can be avoided. What may look ambiguous to a stranger may be completely obvious to the average native user.

Note that the limitations defined in Section 3.1 and the recommendations given here are of a different nature. The limitations defined in Section 3.1 are necessary to avoid duplicate encodings that are artifacts of digital representation and that the user has no way to distinguish visually. On the other hand, in a given context, an identifier such as "BOX0021" can be completely appropriate, and it is impossible to find an algorithm that distinguishes the appropriate from the confusing identifiers.

In certain cases, there is a chance that letters from different scripts look the same. The best know example is the Latin 'A', the Greek 'Alpha', and the Cyrillic 'A'. To disambiguate such cases, it should be assumed that all letters in a component are from the same script. This is similar to the heuristics used to distinguish between letters and numbers in the examples above. Also, for the above three scripts, using lower case letters results in much fewer ambiguities than using upper-case letters.

4.5 Display of URI/IRIs

Many systems contain software that presents URIs to users as part of the system's user interface (sometimes presenting 'friendly' URIs, i.e., a shortened or more legible subset of the URI.) This section applies to this presentation, as well as to the strategy for printing URIs in magazines, newspapers, or reading them over the radio.

Software that displays identifiers to users should follow a general principle: "Don't display something to a user that the user would not be able to enter." The consequences of this principle require judgement about the availability of software that implements the entry

methods described in Section 3.2.

a) In situations where a viewer is not likely to have software that implements non-ASCII character entry (as described in Section 3.1), or where it can be expected that only a limited range of non-ASCII characters can be entered, any part of an IRI containing characters outside the range allowed in [RFC 2396] or any additions should be escaped before being displayed.

b) In situations where a viewer _is_ likely to have such software, IRIs may be displayed directly.

For display of BIDI IRIs, please see [Bidi].

4.6 Interpretation of URI/IRIs

Software that interprets IRIs as the names of local resources should accept IRIs in multiple forms, and convert and match them with the appropriate local resource names.

First, multiple representations includes both IRIs in the native character encoding of the protocol and also their URI counterparts.

Second, it may include URIs constructed based on other character encodings than UTF-8. Such URIs may be produced by user agents that do not conform to this specification and use legacy encodings to convert non-ASCII characters to URIs. Whether this is necessary, and what character encodings to cover, depends on a number of factors, such as the legacy character encodings used locally and the distribution of various versions of user agents. For example, software for Japanese may accept URIs in Shift_JIS and/or EUC-JP in addition to UTF-8.

Third, it may include additional mappings to be more user-friendly and robust against transmission errors. These would be similar to how currently some servers treat URIs as case-insensitive, or perform additional matchings to account for spelling errors. For characters beyond the ASCII repertoire, this may e.g. include ignoring the accents on received IRIs or resource names where appropriate. Please note that such mappings, including case mappings, are language-dependent.

It can be difficult to unambiguously identify a resource if too many mappings are taken into consideration. However, escaped and non-escaped parts of IRIs can always clearly be distinguished. Also, the regularity of UTF-8 (see [Duer97] makes the potential for collisions lower than it may seem at first sight.

4.7 Transportation of URI/IRIs in document formats and protocols

Document formats that transport URIs may need to be upgraded to allow the transport of IRIs. In those cases where the document as a whole has a native character encoding, IRIs should also be encoded in this

encoding, and converted accordingly by a parser or interpreter. IRI characters that are not expressible in the native encoding should be escaped according to Section 2.2, or may be escaped in another way if the document format provides a way to do this. For example, in HTML, XML, or SGML, numeric character references can be used.

Please note that some formats already IRIs, although they use different terminology. HTML 4.0 [HTML4] defines the conversion from IRIs to URIs as error-avoiding behavior. XML 1.0 [XML1], XLink [XLink], and XML Schema [XMLSchema] and specifications based upon them allow IRIs. Also, it is expected that all relevant new W3C formats and protocols will be required to handle IRIs [CharMod].

5. Upgrading strategy

As this recommendation places further constraints on software for which many instances are already deployed, it is important to introduce upgrades carefully, and to be aware of the various interdependencies.

If IRIs cannot be interpreted correctly, they should not be generated or transported. This suggests that upgrading URI interpreting software to accept IRIs should have highest priority.

On the other hand, a single IRI is interpreted only by a single or very few interpreters that are known in advance, while it may be entered and transported very widely.

Therefore, IRIs benefit most from a broad upgrade of software to be able to enter and transport IRIs, but before publishing any individual IRI, care should be taken to upgrade the corresponding interpreting software in order to cover the forms expected to be received by various versions of entry and transport software.

The upgrade of generating software to generate IRIs instead of a local encoding should happen only after the service is upgraded to accept IRIs. Similarly, IRIs should only be generated when the service accepts IRIs and the intervening infrastructure and protocol is known to transport them safely.

Display software should be upgraded only after upgraded entry software has been widely deployed to the population that will see the displayed result.

These recommendations, when taken together, will allow for the extension from URIs to IRIs in order to handle scripts other than ASCII while minimizing interoperability problems.

6. Security Considerations

If IRI entry software normalizes the characters entered, but the resource names on the interpreting side are not normalized accordingly, and the interpreting software does not take this into account, there is a possibility of "spoofing". Similar possibilities turn up when interpreting software accepts URIs in various native encodings or allows accents and similar things to be ignored.

"Spoofing" means that somebody may add a resource name that looks the same or similar to the user while actually being different, or a resource name that contains the same characters, but in a different encoding. The added resource may pretend to be the real resource by looking very similar, but may contain all kinds of changes that may be difficult to spot but can cause all kinds of problems.

Conceptually, this is no different from the problems surrounding the use of case-insensitive web servers. For example, a popular web page with a mixed case name (http://big.site/PopularPage.html) might be "spoofed" by someone who obtains access to (http://big.site/popularpage.html).

However, the introduction of character normalization, of additional mappings for user convenience, and of mappings for various encodings may increase the number of spoofing possibilities. In some cases, in particular for Latin-based resource names, this is usually easy to detect because UTF-8-encoded names, when interpreted and viewed as legacy encodings, produce mostly garbage. In other cases, when concurrently used encodings have a similar structure, but there are no characters that have exactly the same encoding, detection is more difficult. A good example may be the concurrent use of Shift_JIS and EUC-JP on a Japanese server.

Administrators of large sites which allow independent users to create subareas may need to be careful that the aliasing rules do not create chances for spoofing.

7. Acknowledgements

The issue addressed here has been discussed at numerous times over the last many years; for example, there was a thread in the HTML working group in August 1995 (under the topic of "Globalizing URIs") in the www-international mailing list in July 1996 (under the topic of "Internationalization and URLs"), and ad-hoc meetings at the Unicode conferences in September 1995 and September 1997.

Thanks to Francois Yergeau, Chris Wendt, Yaron Goland, Graham Klyne, Roy Fielding, M.T. Carrasco Benitez, James Clark, Andrea Vine, Leslie Daigle, Makoto MURATA and many others for help with understanding the issues and possible solutions. Thanks also to the members of the W3C I18N Working Group and Interest Group for their work on [CharMod], to the members of many other W3C WGs for adopting the ideas, and to

the members of the Montreal IAB Workshop on Internationalization and Localization for a healthy review.

8. Copyright

9. Author's addresses

Larry Masinter
Adobe Systems Incorporated
Mail Stop W14
345 Park Ave
San Jose, CA 95110
mailto:LMM@acm.org
http://larry.masinter.net
Tel: +1 408 536-3024

Martin J. Duerst
W3C/Keio University
5322 Endo, Fujisawa
252-8520 Japan
mailto:duerst@w3.org
http://www.w3.org/People/D%C3%BCrst/
Tel/Fax: +81 466 49 1170

Note: The homepage URI of the second author is the escaped form of an IRI.

Note: Please write "Duerst" with u-umlaut wherever possible, e.g. as "Dürst" in XML and HTML.

10. References

[Bidi] M. Duerst, "Internet Identifiers and Bidirectionality", Internet Draft, July 2001, <http://www.ietf.org/internet-drafts/draft-duerst-iri-bidi-00.txt>, work in progress.

[CharMod] M. Duerst, F. Yergeau et al., Ed., "Character Model for the World Wide Web", <http://www.w3.org/TR/charmod>, work in progress.

[Duer97] M. Duerst, "The Properties and Promizes of UTF-8", Proc. 11th International Unicode Conference, San Jose, September 1997. <http://www.ifi.unizh.ch/mml/mduerst/papers/PDF/IUC11-UTF-8.pdf>

[Duer01] M. Duerst, "Internationalized Resource Identifiers: From Specification to Testing", Proc. 19th International Unicode Conference, San Jose, September 2001, <http://www.w3.org/2001/Talks/0912-IUC-IRI/paper.html>

[HTML4] "HTML 4.0", World Wide Web Consortium, <http://www.w3.org/TR/REC-html40/appendix/notes.html#h-B.2>.

[IDN-URI] M. Duerst, "Internationalized Domain Names in URIs and IRIs", Internet Draft, November 2001, <http://www.ietf.org/internet-drafts/draft-ietf-idn-uri-01.txt>, work in progress.

[ISO10646] ISO/IEC, Information Technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane, Oct. 2000, with amendments.

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

[RFC 2141] R. Moats, "URN Syntax", May 1997.

[RFC 2192] C. Newman, "IMAP URL Scheme", September 1997.

[RFC 2277] H. Alvestrad, "IETF Policy on Character Sets and Languages".

[RFC 2279] F. Yergeau. "UTF-8, a transformation format of ISO 10646.", January 1998.

[RFC 2384] R. Gellens, "POP URL Scheme", August 1998.

[RFC 2396] T.Berners-Lee, R.Fielding, L.Masinter. "Uniform Resource Identifiers (URI): Generic Syntax." August, 1998.

[RFC 2397] L. Masinter, "The "data" URL scheme", August 1998.

[RFC 2616] R.Fielding, J.Gettys, et al, "Hypertext Transfer Protocol -- HTTP/1.1", June 1999.

[RFC 2640] B. Curtis, "Internationalization of the File Transfer Protocol", July 1999.

[RFC 2718] L. Masinter, H. Alvestrand, D. Zigmond, R. Petke, "Guidelines for new URL Schemes", November 1999.

[RFC 2732] R. Hinden, B. Carpenter, L. Masinter, "Format for Literal IPv6 Addresses in URL's", December 1999.

[UNIV3] The Unicode Consortium, "The Unicode Standard Version 3.0", Addison-Wesley, Reading, MA, 2000.

[UNI15] M.Davis and M.Duerst, "Unicode Normalization Forms", Unicode Technical Report #15, November 1999.
<http://www.unicode.org/unicode/reports/tr15/>

[UNIXML] M. Duerst and A. Freytag, "Unicode in XML and other Markup Languages", Unicode Technical Report #20, W3C Note 15 December 2000. <http://www.unicode.org/unicode/reports/tr20/> or
<http://www.w3.org/TR/unicode-xml/>

[W3C IRI] Internationalization - URIs and other identifiers
<http://www.w3.org/International/O-URL-and-ident.html>.

[XLink] Steve DeRose et al., "XML Linking Language (XLink) Version 1.0", World Wide Web Consortium Recommendation 27 June 2001.
<http://www.w3.org/TR/xlink/#link-locators>

[XML1] Tim Bray et al., "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium Recommendation, Oct. 2000. <http://www.w3.org/TR/REC-xml#sec-external-ent>, including Erratum 26 at
<http://www.w3.org/XML/xml-V10-2e-errata#E26>

[XMLSchema] P. Biron, A. Malhotra, "XML Schema Part 2: Datatypes", World Wide Web Consortium Recommendation 2 May 2001.
<http://www.w3.org/TR/xmlschema-2/#anyURI>