Network Working Group
Internet-Draft
Expires: May 31, 2005

M. Duerst
W3C
M. Suignard
Microsoft Corporation
November 30, 2004

Internationalized Resource Identifiers (IRIs)
draft-duerst-iri-11

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

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 Internet-Draft will expire on May 31, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004).

Abstract

This document defines a new protocol element, the Internationalized Resource Identifier (IRI), as a complement to the Uniform Resource Identifier (URI). An IRI is a sequence of characters from the Universal Character Set (Unicode/ISO 10646). A mapping from IRIs to URIs is defined, which means that IRIs can be used instead of URIs where appropriate to identify resources.

The approach of defining a new protocol element was chosen, instead of 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.

Table of Contents

1. Introduction

1.1 Overview and Motivation

A Uniform Resource Identifier (URI) is defined in [RFCYYYY] as a sequence of characters chosen from a limited subset of the repertoire of US-ASCII [ASCII] characters.

The characters in URIs are frequently used for representing words of natural languages. 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 have transcriptions to Latin letters. 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 increasingly widespread. Also, there are increasing numbers of protocols and formats that can carry a wide range of characters.

This document defines a new protocol element, called Internationalized Resource Identifier (IRI), by extending the syntax of URIs to a much wider repertoire of characters. It also defines "internationalized" versions corresponding to other constructs from [RFCYYYY], such as URI references. The syntax of IRIs is defined in Section 2, and the relationship between IRIs and URIs in Section 3.

Using characters outside of A-Z in IRIs brings with it some difficulties. Section 4 discusses the special case of bidirectional IRIs, Section 5 various forms of equivalence between IRIs, and Section 6 the use of IRIs in different situations. Section 7 gives additional informative guidelines, and Section 8 security considerations.

1.2 Applicability

IRIs are designed to be compatible with recommendations for new URI schemes [RFC2718]. The compatibility is provided by specifying a well defined and deterministic mapping from the IRI character sequence to the functionally equivalent URI character sequence. Practical use of IRIs (or IRI references) in place of URIs (or URI references) depends on the following conditions being met:

a) The protocol or format element where IRIs are used should be explicitly designated to be able to carry IRIs. That is, the intent is not to introduce IRIs into contexts that are not defined to accept them. For example, XML schema [XMLSchema] has an explicit type "anyURI" that includes IRIs and IRI references. Therefore, IRIs and IRI references can be in attributes and elements of type "anyURI". On the other hand, in the HTTP protocol [RFC2616], the Request URI is defined as an URI, which means that direct use of IRIs is not allowed in HTTP requests.

b) The protocol or format carrying the IRIs should have a mechanism to represent the wide range of characters used in IRIs, either natively or by some protocol- or format-specific escaping mechanism (for example numeric character references in [XML1]).

c) The URI corresponding to the IRI in question has to encode original characters into octets using UTF-8. For new URI schemes, this is recommended in [RFC2718]. It can apply to a whole scheme (e.g. IMAP URLs [RFC2192] and POP URLs [RFC2384], or the URN syntax [RFC2141]). It can apply to a specific part of a URI, such as the fragment identifier (e.g. [XPointer]). It can apply to a specific URI or part(s) thereof. For details, please see Section 6.4.

1.3 Definitions

The following definitions are used in this document; they follow the terms in [RFC2130], [RFC2277] and [ISO10646]:

character: A member of a set of elements used for the organization, control, or representation of data. For example, "LATIN CAPITAL LETTER A" names a character.

octet: An ordered sequence of eight bits considered as a unit

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.

charset: The name of a parameter or attribute used to identify a character encoding.

UCS: Universal Character Set; the coded character set defined by ISO/IEC 10646 [ISO10646] and the Unicode Standard [UNIV4].

IRI reference: The term "IRI reference" denotes the common usage of an Internationalized Resource Identifier. An IRI reference may be absolute or relative. However, the "IRI" that results from such a reference only includes absolute IRIs; any relative IRI references are resolved to their absolute form. Note that in [RFC2396], URIs did not include fragment identifiers, but in [RFCYYYY], fragment identifiers are part of URIs.

running text: Human text (paragraphs, sentences, phrases) with syntax according to orthographic conventions of a natural language, as opposed to syntax defined for ease of processing by machines (markup, programming languages,...).

protocol element: Any portion of a message which affects processing of that message by the protocol in question.

presentation element: Presentation form corresponding to a protocol element, for example using a wider range of characters.

create (an URI or IRI): With respect to URIs and IRIs, the word 'create' is used for the initial creation. This may be the initial creation of a resource with a certain identifier, or the initial exposition of a resource under a particular identifier.

generate (an URI or IRI): With respect to URIs and IRIs, the word 'generate' is used when the IRI is generated by derivation from other information.

1.4 Notation

RFCs and Internet Drafts currently do not allow any characters outside the US-ASCII repertoire. Therefore, this document uses various special notations to denote such characters in examples.

In text, characters outside US-ASCII are sometimes referenced by using a prefix of 'U+', followed by four to six hexadecimal digits.

To represent characters outside US-ASCII in examples, this document uses two notations called 'XML Notation' and 'Bidi Notation'.

XML Notation uses leading '&#x', trailing ';', and the hexadecimal

number of the character in the UCS in between. Example: я stands for CYRILLIC CAPITAL LETTER YA. In this notation, an actual '&' is denoted by '&'.

Bidi Notation is used for bidirectional examples: lower case letters stand for Latin letters or other letters that are written left-to-right, whereas upper case letters represent Arabic or Hebrew letters that are written right-to-left.

To denote actual octets in examples (as opposed to percent-encoded octets), the two hex digits denoting the octet are enclosed in "<" and ">". For example, the octet often denoted as 0xc9 is denoted here as <c9>.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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, not as a sequence of octets. This definition accommodates the fact that IRIs may be written on paper or read over the radio as well as being stored or transmitted digitally. 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 (and/or transfer 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 [RFCYYYY], but the class of unreserved characters is extended by adding the characters of the UCS (Universal Character Set, [ISO10646]) beyond U+007F, subject to the limitations given in the syntax rules below and in Section 6.1.

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

Characters outside the US-ASCII repertoire are not reserved and

therefore 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 in IRIs, 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 in URIs.

2.2 ABNF for IRI References and IRIs

While it might be possible to define IRI references and IRIs merely by their transformation to URI references and URIs, they can also be accepted and processed directly. Therefore, an ABNF definition for IRI references (which are the most general concept and the start of the grammar) and IRIs is given here. The syntax of this ABNF is described in [RFC2234]. Character numbers are taken from the UCS, without implying any actual binary encoding. Terminals in the ABNF are characters, not bytes.

The following grammar closely follows the URI grammar in [RFCYYYY], except that the range of unreserved characters is expanded to include UCS characters, with the restriction that private UCS characters can occur only in query parts and not elsewhere. The grammar is split into two parts, rules that differ from [RFCYYYY] because of the above-mentioned expansion, and rules that are the same as in [RFCYYYY]. For rules that are different than in [RFCYYYY], the names of the non-terminals have been changed as follows: If the non-terminal contains 'URI', this has been changed to 'IRI'. Otherwise, an 'i' has been prefixed.

The following rules are different from [RFCYYYY]:

```
   IRI            = scheme ":" ihier-part [ "?" iquery ]
                      [ "#" ifragment ]

   ihier-part     = "//" iauthority ipath-abempty
                  / ipath-absolute
                  / ipath-rootless
                  / ipath-empty

   IRI-reference  = IRI / irelative-ref

   absolute-IRI   = scheme ":" ihier-part [ "?" iquery ]

   irelative-ref  = irelative-part [ "?" iquery ] [ "#" ifragment ]

   irelative-part = "//" iauthority ipath-abempty
                   / ipath-absolute
                  / ipath-noscheme
```

```
                  / ipath-empty

    iauthority    = [ iuserinfo "@" ] ihost [ ":" port ]
    iuserinfo     = *( iunreserved / pct-encoded / sub-delims / ":" )
    ihost         = IP-literal / IPv4address / ireg-name

    ireg-name     = *( iunreserved / pct-encoded / sub-delims )

    ipath         = ipath-abempty   ; begins with "/" or is empty
                  / ipath-absolute  ; begins with "/" but not "//"
                  / ipath-noscheme  ; begins with a non-colon segment
                  / ipath-rootless  ; begins with a segment
                  / ipath-empty     ; zero characters

    ipath-abempty  = *( "/" isegment )
    ipath-absolute = "/" [ isegment-nz *( "/" isegment ) ]
    ipath-noscheme = isegment-nz-nc *( "/" isegment )
    ipath-rootless = isegment-nz *( "/" isegment )
    ipath-empty    = 0<ipchar>

    isegment       = *ipchar
    isegment-nz    = 1*ipchar
    isegment-nz-nc = 1*( iunreserved / pct-encoded / sub-delims
                       / "@" )
                  ; non-zero-length segment without any colon ":"

    ipchar         = iunreserved / pct-encoded / sub-delims / ":"
                  / "@"

    iquery         = *( ipchar / iprivate / "/" / "?" )

    ifragment      = *( ipchar / "/" / "?" )

    iunreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~" / ucschar

    ucschar        = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF
                  / %x10000-1FFFD / %x20000-2FFFD / %x30000-3FFFD
                  / %x40000-4FFFD / %x50000-5FFFD / %x60000-6FFFD
                  / %x70000-7FFFD / %x80000-8FFFD / %x90000-9FFFD
                  / %xA0000-AFFFD / %xB0000-BFFFD / %xC0000-CFFFD
                  / %xD0000-DFFFD / %xE1000-EFFFD

    iprivate       = %xE000-F8FF / %xF0000-FFFFD / %x100000-10FFFD
```

Some productions are ambiguous. The "first-match-wins" (a.k.a. "greedy") algorithm applies. For details, see [RFCYYYY].

The following are the same as in [RFCYYYY]:

```
      scheme         = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

      port           = *DIGIT

      IP-literal     = "[" ( IPv6address / IPvFuture  ) "]"

      IPvFuture      = "v" 1*HEXDIG "." 1*( unreserved / sub-delims
                     / ":" )

      IPv6address    =                            6( h16 ":" ) ls32
                     /                       "::" 5( h16 ":" ) ls32
                     / [               h16 ] "::" 4( h16 ":" ) ls32
                     / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                     / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                     / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                     / [ *4( h16 ":" ) h16 ] "::"              ls32
                     / [ *5( h16 ":" ) h16 ] "::"              h16
                     / [ *6( h16 ":" ) h16 ] "::"

      h16             = 1*4HEXDIG
      ls32           = ( h16 ":" h16 ) / IPv4address

      IPv4address    = dec-octet "." dec-octet "." dec-octet
                        "." dec-octet

      dec-octet      = DIGIT                 ; 0-9
                     / %x31-39 DIGIT         ; 10-99
                     / "1" 2DIGIT            ; 100-199
                     / "2" %x30-34 DIGIT     ; 200-249
                     / "25" %x30-35          ; 250-255

      pct-encoded    = "%" HEXDIG HEXDIG

      unreserved     = ALPHA / DIGIT / "-" / "." / "_" / "~"
      reserved       = gen-delims / sub-delims
      gen-delims     = ":" / "/" / "?" / "#" / "[" / "]" / "@"
      sub-delims     = "!" / "$" / "&" / "'" / "(" / ")"
                     / "*" / "+" / "," / ";" / "="
```

This syntax does not support IPv6 scoped addressing zone identifiers.

3. Relationship between IRIs and URIs

IRIs are meant to replace URIs in identifying resources for protocols, formats and software components which use a UCS-based character repertoire. These protocols and components may never need to use URIs directly, especially when the resource identifier is used simply for identification purposes. However, when the resource

identifier is used for resource retrieval, it is in many cases necessary to determine the associated URI because most retrieval mechanisms currently only are defined for URIs. In this case, IRIs can serve as presentation elements for URI protocol elements. An example would be an address bar in a Web user agent. (Additional rationale is given in Section 3.1.)

3.1 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 thereof (for example fragment identifiers).

This mapping has two purposes:

a) Syntactical: Many URI schemes and components define additional syntactical restrictions not captured in Section 2.2. Scheme-specific restrictions are applied to IRIs by converting IRIs to URIs and checking the URIs against the scheme-specific restrictions.

b) Interpretational: URIs identify resources in various ways. IRIs also identify resources. When the IRI is used solely for identification purposes, it is not necessary to map the IRI to a URI (see Section 5). However, when an IRI is used for resource retrieval, the resource that the IRI locates is the same as the one located by the URI obtained after converting the IRI according to the procedure defined here. This means that there is no need to define resolution separately on the IRI level.

Applications MUST map IRIs to URIs using the following two steps.

Step 1) This step generates a UCS character sequence from the original IRI format. This step has three variants, depending on the form of the input.

Variant A) If the IRI is written on paper or read out loud, or otherwise represented as a sequence of characters independent of any character encoding: Represent the IRI as a sequence of characters from the UCS normalized according to Normalization Form C (NFC, [UTR15]).

Variant B) If the IRI is in some digital representation (e.g. an octet stream) in some known non-Unicode character encoding: Convert the IRI to a sequence of characters from the UCS normalized according to NFC.

Variant C) If the IRI is in an Unicode-based character encoding (for example UTF-8 or UTF-16): Do not normalize (see Section 5.3.2.2 for details). Apply Step 2 directly to the encoded Unicode character sequence.

Step 2) For each character in 'ucschar' or 'iprivate', apply Steps 2.1 through 2.3 below.

2.1) Convert the character to a sequence of one or more octets using UTF-8 [RFC3629].

2.2) Convert each octet to %HH, where HH is the hexadecimal notation of the octet value. Note that this is identical to the percent-encoding mechanism in Section 2.1 of [RFCYYYY]. To reduce variability, the hexadecimal notation SHOULD use upper case letters.

2.3) Replace the original character with the resulting character sequence (i.e., a sequence of %HH triplets).

The above mapping from IRIs to URIs produces URIs fully conforming to [RFCYYYY]. The mapping is also an identity transformation for URIs and is idempotent -- applying the mapping a second time will not change anything. Every URI is by definition an IRI.

Infrastructure accepting IRIs MAY convert the ireg-name component of an IRI as follows (before Step 2 above) for schemes that are known to use domain names in ireg-name, but where the scheme definition does not allow percent-encoding for ireg-name: Replace the ireg-name part of the IRI by the part converted using the ToASCII operation specified in Section 4.1 of [RFC3490] on each dot-separated label, and using U+002E (FULL STOP) as a label separator, with the flag UseSTD3ASCIIRules set to TRUE and the flag AllowUnassigned set to FALSE for creating IRIs and set to TRUE otherwise. The ToASCII operation may fail, but this would mean that the IRI cannot be resolved. This conversion SHOULD be used when the goal is to maximize interoperability with legacy URI resolvers. For example, the IRI
http://résumé.example.org may be converted to
http://xn--rsum-bpad.example.org instead of
http://r%C3%A9sum%C3%A9.example.org.

An IRI with a scheme that is known to use domain names in ireg-name, but where the scheme definition does not allow percent-encoding for ireg-name, meets scheme-specific restrictions if either the straightforward conversion or the conversion using the ToASCII operation on ireg-name result in an URI that meets the scheme-specific restrictions. Such an IRI resolves to the URI

obtained after converting the IRI including using the ToASCII operation on ireg-name. Implementations do not need to do this conversion as long as they produce the same result.

Note: The difference between Variants B and C in Step 1 (Variant B using normalization with NFC while Variant C not using any normalization) is to account for the fact that in many non-Unicode character encodings, some text cannot be represented directly. For example, 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), whereas direct transcoding of other 8-bit encodings of Vietnamese may lead to other representations.

Note: The uniform treatment of the whole IRI in Step 2 above is important to not make processing dependent on URI scheme. See [Gettys] for an in-depth discussion.

Note: In practice, the difference above will not be noticed if mapping from IRI to URI and resolution is tightly integrated (e.g. carried out in the same user agent). But conversion using [RFC3490] may be able to better deal with backwards compatibility issues in case mapping and resolution are separated, as in the case of using an HTTP proxy.

Note: Internationalized Domain Names may be contained in parts of an IRI other than the ireg-name part. It is the responsibility of scheme-specific implementations (if the Internationalized Domain Name is part of the scheme syntax) or of server-side implementations (if the Internationalized Domain Name is part of 'iquery') to apply the necessary conversions at the appropriate point. Example: Trying to validate the Web page at http://résumé.example.org would lead to an IRI of http://validator.w3.org/check?uri=http%3A%2F%2Frésumé.example.org, which would convert to a URI of http://validator.w3.org/check?uri=http%3A%2F%2Fr%C3%A9sum%C3%A9.example.org. The server side implementation would be responsible to do the necessary conversions in order to be able to retrieve the Web page.

Infrastructure accepting IRIs MAY also deal with the printable characters in US-ASCII that are not allowed in URIs, namely "<", ">", '"', Space, "{", "}", "|", "\", "^", and "`", in Step 2 above. If such characters are found but are not converted, then the conversion SHOULD fail. Please note that the number sign ("#"), the percent sign ("%"), and the square bracket characters ("[", "]") are not part

of the above list, and MUST NOT be converted. Protocols and formats that have used earlier definitions of IRIs including these characters MAY require percent-encoding of these characters as a preprocessing step to extract the actual IRI from a given field. Such preprocessing MAY also be used by applications allowing the user to enter an IRI.

Note: In this process (in Step 2.3), characters allowed in URI references as well as existing percent-encoded sequences are not encoded further. (This mapping is similar to, but different from, the encoding applied when including arbitrary content into some part of a URI.) For example, an IRI of http://www.example.org/red%09rosé#red (in XML notation) is converted to http://www.example.org/red%09ros%C3%A9#red, not to something like http%3A%2F%2Fwww.example.org%2Fred%2509ros%C3%A9%23red.

Note: Some older software transcoding to UTF-8 may produce illegal output for some input, in particular for characters outside the BMP (Basic Multilingual Plane). As an example, for the following IRI with non-BMP characters (in XML Notation): http://example.com/𐌀𐌁𐌂 (the first three letters of the Old Italic alphabet) the correct conversion to a URI is: http://example.com/%F0%90%8C%80%F0%90%8C%81%F0%90%8C%82

3.2 Converting URIs to IRIs

In some situations, it may be desirable to try to convert a URI into an equivalent IRI. This section gives a procedure to do such a conversion. The conversion described in this section will always result in an IRI which maps back to the URI that was used as an input for the conversion (except for potential case differences in percent-encoding and for potential percent-encoded unreserved characters). However, the IRI resulting from this conversion may not be exactly the same as the original IRI (if there ever was one).

URI to IRI conversion removes percent-encodings, but not all percent-encodings can be eliminated. There are several reasons for this:

a) Some percent-encodings are necessary to distinguish percent-encoded and unencoded uses of reserved characters.

b) Some percent-encodings cannot be interpreted as sequences of UTF-8 octets.

(Note: The octet patterns of UTF-8 are highly regular. Therefore, there is a very high probability, but no guarantee, that percent-encodings that can be interpreted as sequences of UTF-8 octets actually originated from UTF-8. For a detailed discussion, see [Duerst97].)

c) The conversion may result in a character that is not appropriate in an IRI. See Section 2.2, Section 4.1, and Section 6.1 for further details.

Conversion from a URI to an IRI is done using the following steps (or any other algorithm that produces the same result):

1) Represent the URI as a sequence of octets in US-ASCII.

2) Convert all percent-encodings (% followed by two hexadecimal digits) except those corresponding to '%', characters in 'reserved', and characters in US-ASCII not allowed in URIs, to the corresponding octets.

3) Re-percent-encode any octet produced in Step 2 that is not part of a strictly legal UTF-8 octet sequence.

4) Re-percent-encode all octets produced in Step 3 that in UTF-8 represent characters that are not appropriate according to Section 2.2, Section 4.1, and Section 6.1.

5) Interpret the resulting octet sequence as a sequence of characters encoded in UTF-8.

This procedure will convert as many percent-encoded characters as possible to characters in an IRI. Because there are some choices when applying Step 4 (see Section 6.1), results may vary.

Conversions from URIs to IRIs MUST NOT use any other character encoding than UTF-8 in Steps 3 and 4 above, even if it might be possible from context to guess that another character encoding than UTF-8 was used in the URI. As an example, the URI http://www.example.org/r%E9sum%E9.html might with some guessing be interpreted to contain two e-acute characters encoded as iso-8859-1. It must not be converted to an IRI containing these e-acute characters. Otherwise, the IRI will in the future be mapped to http://www.example.org/r%C3%A9sum%C3%A9.html, which is a different URI than http://www.example.org/r%E9sum%E9.html.

3.2.1 Examples

This section shows various examples of converting URIs to IRIs. Each

example shows the result after applying each of the Steps 1 to 5. XML Notation is used for the final result.

The following example contains the sequence '%C3%BC', which is a strictly legal UTF-8 sequence, and which is converted into the actual character U+00FC LATIN SMALL LETTER U WITH DIAERESIS (also known as u-umlaut).

1) http://www.example.org/D%C3%BCrst

2) http://www.example.org/D<c3><bc>rst

3) http://www.example.org/D<c3><bc>rst

4) http://www.example.org/D<c3><bc>rst

5) http://www.example.org/Dürst

The following example contains the sequence '%FC', which might represent U+00FC LATIN SMALL LETTER U WITH DIAERESIS in the iso-8859-1 character encoding. (It might represent other characters in other character encodings. For example, the octet <fc> in iso-8859-5 represents U+045C CYRILLIC SMALL LETTER KJE.) Because <fc> is not part of a strictly legal UTF-8 sequence, it is re-percent-encoded in Step 3.

1) http://www.example.org/D%FCrst

2) http://www.example.org/D<fc>rst

3) http://www.example.org/D%FCrst

4) http://www.example.org/D%FCrst

5) http://www.example.org/D%FCrst

The following example contains '%e2%80%ae', which is the percent-encoded
UTF-8 character encoding of U+202E, RIGHT-TO-LEFT OVERRIDE. Section 4.1 forbids the direct use of this character in an IRI. Therefore, the corresponding octets are re-percent-encoded in Step 4. This example shows that the case (upper or lower) of letters used in percent-encodes may not be preserved. The example also contains a punycode-encoded domain name label (xn--99zt52a), which is not converted.

1) http://xn--99zt52a.example.org/%e2%80%ae

2) http://xn--99zt52a.example.org/<e2><80><ae>

3) http://xn--99zt52a.example.org/<e2><80><ae>

4) http://xn--99zt52a.example.org/%E2%80%AE

5) http://xn--99zt52a.example.org/%E2%80%AE

Implementations with scheme-specific knowledge MAY convert punycode-encoded domain name labels to the corresponding characters using the ToUnicode procedure. Thus, for the example above, the label xn--99zt52a may be converted to U+7D0D U+8C46 (Japanese Natto), leading to the overall IRI of http://納豆.example.org/%E2%80%AE

4. 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 (rtl) 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).

Because of the complex interaction between the logical representation, the visual representation, and the syntax of a Bidi IRI, a balance is needed between various requirements. The main requirements are:

1) user-predictable conversion between visual and logical representation;

2) the ability to include a wide range of characters in various parts of the IRI;

3) minor or no changes or restrictions for implementations.

4.1 Logical Storage and Visual Presentation

When stored or transmitted in digital representation, bidirectional IRIs MUST be in full logical order, and MUST conform to the IRI syntax rules (which includes the rules relevant to their scheme). This assures that bidirectional IRIs can be processed in the same way as other IRIs.

When rendered, bidirectional IRIs MUST be rendered using the Unicode Bidirectional Algorithm [UNIV4], [UNI9]. Bidirectional IRIs MUST be rendered in the same way as they would be rendered if they were in an left-to-right embedding, i.e. as if they were preceded by U+202A, LEFT-TO-RIGHT EMBEDDING (LRE), and followed by U+202C, POP DIRECTIONAL FORMATTING (PDF). Setting the embedding direction can also be done in a higher-level protocol (e.g. the dir='ltr' attribute in HTML).

There is no requirement to actually use the above embedding if the display is still the same without the embedding. For example, a bidirectional IRI in a text with left-to-right base directionality (such as used for English or Cyrillic) that is preceded and followed by whitespace and strong left-to-right characters does not need an embedding. Also, a bidirectional relative IRI reference that only contains strong right-to-left characters and weak characters and that starts and ends with a strong rigth-to-left character and appears in a text with right-to-left base directionality (such as used for Arabic or Hebrew) and is preceded and followed by whitespace and strong characters does not need an embedding.

In some other cases, using U+200E, LEFT-TO-RIGHT MARK (LRM) may be sufficient to force the correct display behavior. However, the details of the Unicode Bidirectional algorithm are not always easy to understand. Implementers are strongly advised to err on the side of caution and to use embedding in all cases where they are not completely sure that the display behavior is unaffected without the embedding.

The Unicode Bidirectional Algorithm ([UNI9], Section 4.3) permits higher-level protocols to influence bidirectional rendering. Such changes by higher-level protocols MUST NOT be used if they change the rendering of IRIs.

The bidirectional formatting characters that may be used before or after the IRI to assure correct display are themselves not part of the IRI. IRIs MUST NOT contain bidirectional formatting characters (LRM, RLM, LRE, RLE, LRO, RLO, and PDF). They affect the visual rendering of the IRI, but do not themselves appear visually. It would therefore not be possible to correctly input an IRI with such characters.

4.2 Bidi IRI Structure

The Unicode Bidirectional Algorithm is designed mainly for running text. To make sure that it does not affect the rendering of bidirectional IRIs too much, some restrictions on bidirectional IRIs are necessary. These restrictions are given in terms of delimiters

(structural characters, mostly punctuation such as '@', '.', ':', '/') and components (usually consisting mostly of letters and digits).

The following syntax rules from Section 2.2 correspond to components for the purpose of Bidi behavior: iuserinfo, ireg-name, isegment, isegment-nz, isegment-nz-nc, ireg-name, iquery, and ifragment.

Specifications that define the syntax of any of the above components MAY divide them further and define smaller parts to be components according to this document. As an example, the restrictions of [RFC3490] on bidirectional domain names correspond to treating each label of a domain name as a component for those schemes where ireg-name is a domain name. Even where the components are not defined formally, it may be helpful to think about some syntax in terms of components and to apply the relevant restrictions. For example, for the usual name/value syntax in query parts, it is convenient to treat each name and each value as a component. As another example, the extensions in a resource name can be treated as separate components.

For each component, the following restrictions apply:

1) A component SHOULD NOT use both right-to-left and left-to-right characters.

2) A component using right-to-left characters SHOULD start and end with right-to-left characters.

The above restrictions are given as shoulds, rather than as musts. For IRIs that are never presented visually, they are not relevant. However, for IRIs in general, they are very important to insure consistent conversion between visual presentation and logical representation, in both directions.

Note: In some components, the above restrictions may actually be strictly enforced. For example, [RFC3490] requires that these restrictions apply to the labels of a host name for those schemes where ireg-name is a host name. In some other components, for example path components, following these restrictions may not be too difficult. For other components, such as parts of the query part, it may be very difficult to enforce the restrictions, because the values of query parameters may be arbitrary character sequences.

If the above restrictions cannot be satisfied otherwise, the affected component can always be mapped to URI notation as described in Section 3.1. Please note that the whole component needs to be mapped

(see also Example 9 below).

4.3 Input of Bidi IRIs

Bidi input methods MUST generate Bidi IRIs in logical order while rendering them according to Section 4.1. During input, rendering SHOULD be updated after every new character that is input to avoid end user confusion.

4.4 Examples

This section gives examples of bidirectional IRIs, in Bidi Notation. It shows legal IRIs with the relationship between logical and visual representation, and explains how certain phenomena in this relationship may look strange to somebody not familiar with bidirectional behavior, but familiar to users of Arabic and Hebrew. It also shows what happens if the restrictions given in Section 4.2 are not followed. The examples below can be seen at [BidiEx], in Arabic, Hebrew, and Bidi Notation variants.

To read the bidi text in the examples, read the visual representation from left to right until you encounter a block of rtl text. Read the rtl block (including slashes and other special characters) from right to left, then continue at the next unread ltr character.

Example 1: A single component with rtl characters is inverted:
logical representation: http://ab.CDEFGH.ij/kl/mn/op.html
visual representation: http://ab.HGFEDC.ij/kl/mn/op.html
Components can be read one-by-one, and each component can be read in its natural direction.

Example 2: More than one consecutive component with rtl characters is inverted as a whole:
logical representation: http://ab.CDE.FGH/ij/kl/mn/op.html
visual representation: http://ab.HGF.EDC/ij/kl/mn/op.html
A sequence of rtl components is read rtl, in the same way as a sequence of rtl words is read rtl in a bidi text.

Example 3: All components of an IRI (except for the scheme) are rtl. All rtl components are inverted overall:
logical representation: http://AB.CD.EF/GH/IJ/KL?MN=OP;QR=ST#UV
visual representation: http://VU#TS=RQ;PO=NM?LK/JI/HG/FE.DC.BA
The whole IRI (except the scheme) is read rtl. Delimiters between rtl components stay between the respective components; delimiters between ltr and rtl components don't move.

Example 4: Several sequences of rtl components are each inverted on their own:

logical representation: http://AB.CD.ef/gh/IJ/KL.html
visual representation: http://DC.BA.ef/gh/LK/JI.html
Each sequence of rtl components is read rtl, in the same way as each sequence of rtl words in an ltr text is read rtl.

Example 5: Example 2, applied to components of different kinds:
logical representation: http://ab.cd.EF/GH/ij/kl.html
visual representation: http://ab.cd.HG/FE/ij/kl.html
The inversion of the domain name label and the path component may be unexpected, but is consistent with other bidi behavior. For reassurance that the domain component really is "ab.cd.EF", it may be helpful to read aloud the visual representation following the bidi algorithm. After "http://ab.cd." one reads the RTL block "E-F-slash-G-H", which corresponds to the logical representation.

Example 6: Same as example 5, with more rtl components:
logical representation: http://ab.CD.EF/GH/IJ/kl.html
visual representation: http://ab.JI/HG/FE.DC/kl.html
The inversion of the domain name labels and the path components may be easier to identify because the delimiters also move.

Example 7: A single rtl component with included digits:
logical representation: http://ab.CDE123FGH.ij/kl/mn/op.html
visual representation: http://ab.HGF123EDC.ij/kl/mn/op.html
Numbers are written ltr in all cases, but are treated as an additional embedding inside a run of rtl characters. This is completely consistent with usual bidirectional text.

Example 8 (not allowed): Numbers at the start or end of a rtl component:
logical representation: http://ab.cd.ef/GH1/2IJ/KL.html
visual representation: http://ab.cd.ef/LK/JI1/2HG.html
The sequence '1/2' is interpreted by the bidi algorithm as a fraction, fragmenting the components and leading to confusion. There are other characters that are interpreted in a special way close to numbers, in particular '+', '-', '#', '$', '%', ',', '.', and ':'.

Example 9 (not allowed): The numbers in the previous example are percent-encoded:
logical representation: http://ab.cd.ef/GH%31/%32IJ/KL.html,
visual representation (Hebrew): http://ab.cd.ef/%31HG/LK/JI%32.html
visual representation (Arabic): http://ab.cd.ef/31%HG/%LK/JI32.html
Depending on whether the upper-case letters represent Arabic or Hebrew, the visual representation is different.

Example 10 (allowed, but not recommended):
logical representation: http://ab.CDEFGH.123/kl/mn/op.html
visual representation: http://ab.123.HGFEDC/kl/mn/op.html

Components consisting of only numbers are allowed (it would be rather difficult to prohibit them), but may interact with adjacent RTL components in ways that are not easy to predict.

5. Normalization and Comparison

> Note: The structure and much of the material for this section is taken from section 6 of [RFCYYYY]; the differences are due to the specifics of IRIs.

One of the most common operations on IRIs is simple comparison: determining if two IRIs are equivalent without using the IRIs or the mapped URIs to access their respective resource(s). A comparison is performed every time 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 in respect to some particular purpose, and implementations with differing purposes will often be subject to differing design trade-offs in regards to how much effort should be spent in reducing aliased identifiers. This section describes a variety of methods that may be used to compare IRIs, the trade-offs between them, and the types of applications that might use them.

5.1 Equivalence

Since IRIs exist to identify resources, presumably they should be considered equivalent when they identify the same resource. However, such a definition of equivalence is not of much practical use, since there is no way for an implementation to compare two resources that are not under its own control. 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 applicationdependent versions of equivalence.

Even though it is possible to determine that two IRIs are equivalent, IRI comparison is not sufficient to determine if 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 being compared for the purpose of selecting (or avoiding) a network action, such as retrieval of a representation, fragment components (if any) should 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 5.3.1). All other applications MUST select one of the comparison practices from the Comparison Ladder (see Section 5.3, or, after IRI-to-URI conversion, select one of the comparison practices from the URI comparison ladder [RFCYYYY], Section 6.2.

5.2 Preparation for Comparison

Any kind of IRI comparison REQUIRES that all escapings or encodings in the protocol or format that carries an IRI are resolved. This is usually done when parsing the protocol or format. 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) all get resolved into what is denoted in this document (see Section 1.4) 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 not based on characters, but on octets, and additional care is required to make sure that characters, and not just arbitrary octets, are compared (see Section 5.3.1).

5.3 Comparison Ladder

A variety of methods are used in practice 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 those 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.

5.3.1 Simple String Comparison

If two IRIs, 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 and when a definitive answer to the question of IRI equivalence is needed that is independent of the scheme used and 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 of strings for equality is normally based on pairwise comparison of the characters that make up the strings, starting from the first and proceeding until both strings are exhausted and all characters found to be equal, a pair of characters compares unequal, or one of the strings is exhausted before the other.

Such character comparisons require 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 be 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 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 IRIs SHOULD NOT be modified when being transported if there is any chance that this IRI might be used as an identifier.

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 choose to 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 consistent enough to negate any efficiency that might be obtained from further normalization.

5.3.2 Syntax-based Normalization

Implementations may use logic based on the definitions provided by this specification to reduce the probability of false negatives. Such 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/ros&#xE9;
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.

5.3.2.1 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 5.3.3. 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 US-ASCII characters should be avoided because such a case normalization may be cultural dependant 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.

5.3.2.2 Character Normalization

The Unicode Standard [UNIV4] 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).

Equivalence of IRIs MUST rely on the assumption that IRIs are appropriately pre-character-normalized, rather than applying 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 an 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 using NFC. Using NFKC may avoid even more problems, for example by choosing half-width Latin letters instead of full-width, and full-width Katakana instead of half-width.

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, the latter uses 'e' characters followed by combining acute accents. Both usages are defined to be canonically equivalent in [UNIV4].

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 upper-case/lower-case problems in character-normalized as possible (i.e. NFC or even NFKC). URIs. Some parts of a URI are case-insensitive (domain name). For others, it is unclear whether they are case-sensitive or case-insensitive, or something in between (e.g. case-sensitive, but if the wrong case is used, a multiple choice selection is provided instead of a direct negative result). The best recipe is that the creator uses a reasonable capitalization, and when transferring the URI, that capitalization is never changed.

Various IRI schemes may allow the usage of Internationalized Domain Names (IDN) [RFC3490] either in the ireg-name part or elsewhere. Character Normalization also applies to IDNs, as discussed in Section 5.3.3.

5.3.2.3 Percent-Encoding Normalization

The percent-encoding mechanism (Section 2.1 of [RFCYYYY]) 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. Such IRIs should be normalized by decoding any percent-encoded octet sequence that corresponds to an unreserved character, as described in Section 2.3 of [RFCYYYY].

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 the purpose of local comparison.

5.3.2.4 Path Segment Normalization

The complete path segments "." and ".." are intended only for use within relative references (Section 4.1 of [RFCYYYY]) and are removed as part of the reference resolution process (Section 5.2 of [RFCYYYY]). 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 [RFCYYYY].

5.3.3 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, since 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", where 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 subcomponent is 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 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 may allow the usage of Internationalized Domain Names (IDN) [RFC3490] either in their ireg-name part or elsewhere. When in use in IRIs, those names SHOULD be validated using the ToASCII operation defined in [RFC3490], with the flags "UseSTD3ASCIIRules" and "AllowUnassigned". An IRI containing an invalid IDN cannot successfully be resolved. Validated IDN components of IRIs SHOULD be character normalized using the Nameprep process [RFC3491]; however, 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.

5.3.4 Protocol-based Normalization

Web spiders, for which substantial effort to reduce the incidence of false negatives is often cost-effective, are observed to 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).

6. Use of IRIs

6.1 Limitations on UCS Characters Allowed in IRIs

This section discusses limitations on characters and character sequences usable for IRIs beyond those given in Section 2.2 and Section 4.1. The considerations in this section are relevant when creating IRIs and when converting from URIs to IRIs.

a) The repertoire of characters allowed in each IRI component is limited by the definition of that component. For example, the definition of the scheme component does not allow characters beyond US-ASCII.

   (Note: In accordance with URI practice, generic IRI software cannot and should not check for such limitations.)

b) 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 Latin characters, half-width Katakana characters for Japanese, and many others. This also includes many look-alikes of "space", "delims", and "unwise", characters excluded in [RFC3491].

Additional information is available from [UNIXML]. [UNIXML] is written in the context of running text rather than in the context of identifiers. Nevertheless, it discusses many of the categories of

characters not appropriate for IRIs.

6.2 Software Interfaces and Protocols

Although an IRI is defined as a sequence of characters, software interfaces for URIs typically function on sequences of octets or other kinds of code units. Thus, software interfaces and protocols MUST define which character encoding is used.

Intermediate software interfaces between IRI-capable components and URI-only components MUST map the IRIs per Section 3.1, when transferring from IRI-capable to URI-only components. 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.

6.3 Format of URIs and IRIs in Documents 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 MUST also be encoded in this character encoding, and converted accordingly by a parser or interpreter. IRI characters that are not expressible in the native character encoding SHOULD be escaped using the escaping conventions of the document format if such conventions are available. Alternatively, they MAY be percent-encoded according to Section 3.1. For example, in HTML or XML, numeric character references SHOULD be used. If a document as a whole has a native character encoding, and that character encoding is not UTF-8, then IRIs MUST NOT be placed into the document in the UTF-8 character encoding.

Note: Some formats already accommodate 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].

6.4 Use of UTF-8 for Encoding Original Characters

This section discusses details and gives examples for point c) in Section 1.2. In order to be able to use IRIs, the URI corresponding to the IRI in question has to encode original characters into octets using UTF-8. This can be specified for all URIs of a URI scheme, or can apply to individual URIs for schemes that do not specify how to encode original characters. It can apply to the whole URI, or only some part. For background information on encoding characters into URIs, see also Section 2.5 of [RFCYYYY].

For new URI schemes, using UTF-8 is recommended in [RFC2718]. Examples where UTF-8 is already used are the URN syntax [RFC2141], IMAP URLs [RFC2192], and POP URLs [RFC2384]. On the other hand, because the HTTP URL scheme does not specify how to encode original characters, only some HTTP URLs can have corresponding but different IRIs.

For example, for a document with a URI of http://www.example.org/r%C3%A9sum%C3%A9.html, it is possible to construct a corresponding IRI (in XML notation, see Section 1.4): http://www.example.org/résumé.html (é stands for the e-acute character, and %C3%A9 is the UTF-8 encoded and percent-encoded representation of that character). On the other hand, for a document with a URI of http://www.example.org/r%E9sum%E9.html, the percent-encoding octets cannot be converted to actual characters in an IRI, because the percent-encoding is not based on UTF-8.

This means that for most URI schemes, there is no need to upgrade their scheme definition in order for them to work with IRIs. The main case where upgrading a scheme definition makes sense is when a scheme definition, or a particular component of a scheme, is strictly limited to the use of US-ASCII characters with no provision to include non-ASCII characters/octets via percent-encoding, or if a scheme definition currently uses highly scheme-specific provisions for the encoding of non-ASCII characters. An example of such a scheme might be the mailto: scheme [RFC2368].

This specification does not upgrade any scheme specifications in any way, this has to be done separately. Also, it should be noted that there is no such thing as an "IRI scheme"; all IRIs use URI schemes, and all URI schemes can be used with IRIs, even though in some cases only by using URIs directly as IRIs, without any conversion.

URI schemes can impose restrictions on the syntax of scheme-specific URIs, ie. URIs that are admissable under the generic URI syntax [RFCYYYY] may not be admissable due to narrower syntactic constraints imposed by a URI scheme specification. URI scheme definitions cannot broaden the syntactic restrictions of the generic URI syntax, otherwise it would be possible to generate URIs that satisfied the scheme specific syntactic constraints without satisfying the syntactic constraints of the generic URI syntax. However, additional syntactic constraints imposed by URI scheme specifications are applicable to IRI since the corresponding URI resulting from the mapping defined in Section 3.1 MUST be a valid URI under the syntactic restrictions of generic URI syntax and any narrower restrictions imposed by the corresponding URI scheme specification.

The requirement for the use of UTF-8 applies to all parts of a URI (with the potential exception of the ireg-name part, see Section 3.1). However, it is possible that the capability of IRIs to represent a wide range of characters directly is used just in some parts of the IRI (or IRI reference). The other parts of the IRI may only contain US-ASCII characters, or they may not be based on UTF-8. They may be based on another character encoding, or they may directly encode raw binary data (see also [RFC2397]).

For example, it is possible to have a URI reference of http://www.example.org/r%E9sum%E9.xml#r%C3%A9sum%C3%A9, where the document name is encoded in iso-8859-1 based on server settings, but the fragment identifier is encoded in UTF-8 according to [XPointer]. The IRI corresponding to the above URI would be (in XML notation) http://www.example.org/r%E9sum%E9.xml#résumé.

Similar considerations apply to query parts. The functionality of IRIs (namely to be able to include non-ASCII characters) can only be used if the query part is encoded in UTF-8.

6.5 Relative IRI References

Processing of relative IRI references against a base is handled straightforwardly; the algorithms of [RFCYYYY] can be applied directly, treating the characters additionally allowed in IRI references in the same way as unreserved characters in URI references.

7. URI/IRI Processing Guidelines (informative)

This informative section provides guidelines for 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 creates or generates URIs, software that displays URIs, formats and protocols that transport URIs, and software that interprets URIs. These 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.

7.1 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.

In case the current handling in an API or protocol is based on US-ASCII, UTF-8 is recommended as the character encoding for IRIs,

because this is compatible with US-ASCII, is in accordance with the recommendations of [RFC2277], and makes it easy to convert to URIs where necessary. In any case, the API or protocol definition must clearly define the character encoding to be used.

The transfer from URI-only to IRI-capable components requires no mapping, although the conversion described in Section 3.2 above may be performed. It is preferable not to perform this inverse conversion when there is a chance that this cannot be done correctly.

7.2 URI/IRI Entry

There are components that allow users to enter URIs into the system, for example 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 the user's 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 restrictions defined in Section 2.2 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.

As an example of variant settings, input method editors for East Asian Languages usually allow the input of Latin letters and related characters in full-width or half-width versions. For IRI input, the input method editor should be set so that it produces half-width Latin letters and punctuation, and full-width Katakana.

An input field primarily or only used for the input of URIs/IRIs may allow the user to view an IRI as mapped to a URI. Places where the input of IRIs is frequent may provide the possibility for viewing an IRI as mapped to a 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 map the IRI to a URI before passing it to such a component.

For the input of IRIs with right-to-left characters, please see Section 4.3.

7.3 URI/IRI Transfer Between Applications

Many applications, in particular many mail user agents, try to detect URIs appearing in plain text. For this, they use some heuristics based on URI syntax. They then allow the user to click on such URIs and retrieve the corresponding resource in an appropriate (usually scheme-dependent) application.

Such applications have to be upgraded to use the IRI syntax rather than the URI syntax as a base for heuristics. In particular, a non-ASCII character should not be taken as the indication of the end of an IRI. Such applications also have to make sure that they correctly convert the detected IRI from the character encoding of the document or application where the IRI appears to the character encoding used by the system-wide IRI invocation mechanism, or to a URI (according to Section 3.1) if the system-wide invocation mechanism only accepts URIs.

The clipboard is another frequently used way to transfer URIs and IRIs from one application to another. On most platforms, the clipboard is able to store and transfer text in many languages and scripts. Correctly used, the clipboard transfers characters, not bytes, which will do the right thing with IRIs.

7.4 URI/IRI Generation

Systems that offer 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, 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. For example, if a file system contains a file named résumé.html, a server should expose this as r%C3%A9sum%C3%A9.html in a URI, which allows to use résumé.html in an IRI, even if the file name locally is kept in a character encoding other than UTF-8.

This recommendation in particular applies to HTTP servers. For FTP servers, similar considerations apply, see in particular [RFC2640].

7.5 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 confusing users with "br0ken" or "1ame" identifiers.

Outside of the US-ASCII repertoire, 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. On the other hand, in some cases, the UCS contains variants for compatibility reasons, for example for typographic purposes. These should be avoided wherever possible. Although there may be exceptions, in general newly created resource names should be in NFKC [UTR15] (which means that they are also in NFC).

As an example, the UCS contains the 'fi' ligature at U+FB01 for compatibility reasons. Wherever possible, IRIs should use the two letters 'f' and 'i' rather than the 'fi' ligature. An example where the latter may be used is in the query part of an IRI for an explicit search for a word written containing the 'fi' ligature.

In certain cases, there is a chance that characters from different scripts look the same. The best known example is the Latin 'A', the Greek 'Alpha', and the Cyrillic 'A'. To avoid such cases, only IRIs should be created where all the characters in a single component are used together in a given language. This usually means that all these characters will be from the same script, but there are languages that mix characters from different scripts (such as Japanese). This is similar to the heuristics used to distinguish between letters and numbers in the examples above. Also, for Latin, Greek, and Cyrillic, using lower-case letters results in fewer ambiguities than using upper-case letters.

7.6 Display of URIs/IRIs

In situations where the rendering software is not expected to display non-ASCII parts of the IRI correctly using the available layout and font resources, these parts should be percent-encoded before being

displayed.

For display of Bidi IRIs, please see Section 4.1.

7.7 Interpretation of URIs and 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 include 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 character 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 matching to account for spelling errors. For characters beyond the US-ASCII repertoire, this may for example 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, percent-encoded and not percent-encoded parts of IRIs can always clearly be distinguished. Also, the regularity of UTF-8 (see [Duerst97]) makes the potential for collisions lower than it may seem at first sight.

7.8 Upgrading Strategy

Where 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 created, 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 using a local character 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.

Software converting from URIs to IRIs for display should be upgraded only after upgraded entry software has been widely deployed to the population that will see the displayed result.

It is often possible to reduce the effort and dependencies for upgrading to IRIs by using UTF-8 rather than another character encoding where there is a free choice of character encodings. For example, when setting up a new file-based Web server, using UTF-8 as the character encoding for file names will make the transition to IRIs easier. Likewise, when setting up a new Web form using UTF-8 as the character encoding of the form page, the returned query URIs will use UTF-8 as the character encoding (unless the user, for whatever reason, changes the character encoding) and will therefore be compatible with IRIs.

These recommendations, when taken together, will allow for the extension from URIs to IRIs in order to handle characters other than US-ASCII while minimizing interoperability problems. For considerations regarding the upgrade of URI scheme definitions, please see Section 6.4.

8. Security Considerations

The security considerations discussed in [RFCYYYY] also apply to IRIs. In addition, the following issues require particular care for IRIs.

Incorrect encoding or decoding can lead to security problems. In particular, some UTF-8 decoders do not check against overlong byte sequences. As an example, a '/' is encoded with the byte 0x2F both in UTF-8 and in US-ASCII, but some UTF-8 decoders also wrongly interpret the sequence 0xC0 0xAF as a '/'. A sequence such as

'%C0%AF..' may pass some security tests and then be interpreted as '/..' in a path if UTF-8 decoders are fault-tolerant, if conversion and checking are not done in the right order, and/or if reserved characters and unreserved characters are not clearly distinguished.

There are various ways in which "spoofing" can occur with IRIs. "Spoofing" means that somebody may add a resource name that looks the same or similar to the user, but points to a different resource. 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 and can cause all kinds of problems. Most spoofing possibilities for IRIs are extensions of those for URIs.

Spoofing can occur for various reasons. A first reason is that normalization expectations of a user or actual normalization when entering an IRI, or when transcoding an IRI from a legacy character encoding, do not match the normalization used on the server side. 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.example.com/PopularPage.html) might be "spoofed" by someone who is able to create http://big.example.com/popularpage.html. However, the use of unnormalized character sequences, and of additional mappings for user convenience, may increase the chance for spoofing. Protocols and servers that allow the creation of resources with names that are not normalized are particularly vulnerable to such attacks. This is an inherent security problem of the relevant protocol, server, or resource, and not specific to IRIs, but mentioned here for completeness.

Spoofing can occur in various IRI components, such as the domain name part or a path part. For considerations specific to the domain name part, see [RFC3491]. For the path part, administrators of sites which allow independent users to create resources in the same subarea may need to be careful to check for spoofing.

Spoofing can occur because in the UCS, there are many characters that look very similar. Details are discussed in Section 7.5. Again, this is very similar to spoofing possibilities on US-ASCII, e.g. using 'br0ken' or '1ame' URIs.

Spoofing can occur when URIs with percent-encodings based on various character encodings are accepted to deal with older user agents. 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 character encodings, produce mostly garbage. In other cases, when concurrently used character encodings have a similar structure, but there are no characters that have exactly the

same encoding, detection is more difficult.

Spoofing can occur with bidirectional IRIs, if the restrictions in Section 4.2 are not followed. The same visual representation may be interpreted as different logical representations, and vice versa. It is also very important that a correct Unicode bidirectional implementation is used.

9. IANA Considerations

This document has no actions for IANA.

10. Acknowledgements

We would like to thank Larry Masinter for his work as coauthor of many earlier versions of this document (draft-masinter-url-i18n-xx).

The discussion on the issue addressed here has started a long time ago. There was a thread in the HTML working group in August 1995 (under the topic of "Globalizing URIs") and 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.

Many thanks go to Francois Yergeau, Matitiahu Allouche, Roy Fielding, Tim Berners-Lee, Mark Davis, M.T. Carrasco Benitez, James Clark, Tim Bray, Chris Wendt, Yaron Goland, Andrea Vine, Misha Wolf, Leslie Daigle, Ted Hardie, Bill Fenner, Margaret Wasserman, Russ Housley, Makoto MURATA, Steven Atkin, Ryan Stansifer, Tex Texin, Graham Klyne, Bjoern Hoehrmann, Chris Lilley, Ian Jacobs, Adam Costello, Dan Oscarson, Elliotte Rusty Harold, Mike J. Brown, Roy Badami, Jonathan Rosenne, Asmus Freytag, Simon Josefsson, Carlos Viegas Damasio, Chris Haynes, Walter Underwood, and many others for help with understanding the issues and possible solutions, and getting the details right.

This document is a product of the Internationalization Working Group (I18N WG) of the World Wide Web Consortium (W3C). Thanks to the members of the W3C I18N Working Group and Interest Group for their contributions and their work on [CharMod]. Thanks also go to the members of many other W3C Working Groups for adopting IRIs, and to the members of the Montreal IAB Workshop on Internationalization and Localization for their review.

11. References

11.1 Normative References

[ASCII] American National Standards Institute, "Coded Character

Set -- 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

[ISO10646] International Organization for Standardization, "ISO/IEC 10646:2003: Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO Standard 10646, December 2003.

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

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 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.

[RFCYYYY] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax (Note to the RFC Editor: Please update this reference with the RFC resulting from draft-fielding-uri-rfc2396bis-xx.txt, and remove this Note)", draft-fielding-uri-rfc2396bis-07 (work in progress), April 2004.

[UNI9] Davis, M., "The Bidirectional Algorithm", Unicode Standard Annex #9, March 2004, <http://www.unicode.org/reports/tr9/tr9-13.html>.

[UNIV4] The Unicode Consortium, "The Unicode Standard, Version 4.0.1, defined by: The Unicode Standard, Version 4.0 (Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1), as amended by Unicode 4.0.1 (http://www.unicode.org/versions/Unicode4.0.1/)", March 2004.

[UTR15] Davis, M. and M. Duerst, "Unicode Normalization Forms", Unicode Standard Annex #15, April 2003, <http://www.unicode.org/unicode/reports/tr15/tr15-23.html>.

11.2 Non-normative References

[BidiEx] "Examples of bidirectional IRIs", <http://www.w3.org/International/iri-edit/BidiExamples>.

[CharMod] Duerst, M., Yergeau, F., Ishida, R., Wolf, M. and T. Texin, "Character Model for the World Wide Web", World Wide Web Consortium Working Draft, February 2004, <http://www.w3.org/TR/charmod>.

[Duerst97] Duerst, M., "The Properties and Promises 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>.

[Gettys] Gettys, J., "URI Model Consequences", <http://www.w3.org/DesignIssues/ModelConsequences>.

[HTML4] Raggett, D., Le Hors, A. and I. Jacobs, "HTML 4.01 Specification", World Wide Web Consortium Recommendation, December 1999, <http://www.w3.org/TR/REC-html40/appendix/notes.html#h-B.2>.

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

[RFC2130] Weider, C., Preston, C., Simonsen, K., Alvestrand, H., Atkinson, R., Crispin, M. and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 February - 1 March, 1996", RFC 2130, April 1997.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC2192] Newman, C., "IMAP URL Scheme", RFC 2192, September 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2368] Hoffman, P., Masinter, L. and J. Zawinski, "The mailto URL scheme", RFC 2368, July 1998.

[RFC2384] Gellens, R., "POP URL Scheme", RFC 2384, August 1998.

[RFC2396] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform

Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, August 1998.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2640] Curtin, B., "Internationalization of the File Transfer Protocol", RFC 2640, July 1999.

[RFC2718] Masinter, L., Alvestrand, H., Zigmond, D. and R. Petke, "Guidelines for new URL Schemes", RFC 2718, November 1999.

[UNIXML] Duerst, M. and A. Freytag, "Unicode in XML and other Markup Languages", Unicode Technical Report #20, World Wide Web Consortium Note, February 2002, <http://www.w3.org/TR/unicode-xml/>.

[XLink] DeRose, S., Maler, E. and D. Orchard, "XML Linking Language (XLink) Version 1.0", World Wide Web Consortium Recommendation, June 2001, <http://www.w3.org/TR/xlink/#link-locators>.

[XML1] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E. and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium Recommendation, February 2004, <http://www.w3.org/TR/REC-xml#sec-external-ent>.

[XMLNamespace] Bray, T., Hollander, D. and A. Layman, "Namespaces in XML", World Wide Web Consortium Recommendation, January 1999, <http://www.w3.org/TR/REC-xml-names>.

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

[XPointer] Grosso, P., Maler, E., Marsh, J. and N. Walsh, "XPointer Framework", World Wide Web Consortium Recommendation, March 2003, <http://www.w3.org/TR/xptr-framework/#escaping>.

Authors' Addresses

Martin Duerst (Note: Please write "Duerst" with u-umlaut wherever
possible, for example as "Dürst" in XML and HTML.)
World Wide Web Consortium
5322 Endo
Fujisawa, Kanagawa 252-8520
Japan

Phone: +81 466 49 1170
Fax: +81 466 49 1171
EMail: mailto:duerst@w3.org
URI: http://www.w3.org/People/D%C3%BCrst/
(Note: This is the percent-encoded form of an IRI.)

Michel Suignard
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
U.S.A.

Phone: +1 425 882-8080
EMail: mailto:michelsu@microsoft.com
URI: http://www.suignard.com

Appendix A. Design Alternatives

This section shortly summarizes major design alternatives and the reasons for why they were not chosen.

Appendix A.1 New Scheme(s)

Introducing new schemes (for example httpi:, ftpi:,...) or a new metascheme (e.g. i:, leading to URI/IRI prefixes such as i:http:, i:ftp:,...) was proposed to make IRI-to-URI conversion scheme-dependent or to distinguish between percent-encodings resulting from IRI-to-URI conversion and percent-encodings from legacy character encodings.

New schemes are not needed to distinguish URIs from true IRIs (i.e. IRIs that contain non-ASCII characters). The benefit of being able to detect the origin of percent-encodings is marginal, because UTF-8 can be detected with very high reliability. Deploying new schemes is extremely hard, so not requiring new schemes for IRIs makes deployment of IRIs vastly easier. Making conversion scheme-dependent is highly inadvisable, and would be encouraged by separate schemes for IRIs. Using an uniform convention for conversion from IRIs to URIs makes IRI implementation orthogonal to the introduction of

actual new schemes.

Appendix A.2 Other Character Encodings than UTF-8

At an early stage, UTF-7 was considered as an alternative to UTF-8 when converting IRIs to URIs. UTF-7 would not have needed percent-encoding, and would in most cases have been shorter than percent-encoded UTF-8.

Using UTF-8 avoids a double layering and overloading of the use of the "+" character. UTF-8 is fully compatible with US-ASCII, and has therefore been recommended by the IETF, and is being used widely, while UTF-7 has never been used much and is now clearly being discouraged. Requiring implementations to convert from UTF-8 to UTF-7 and back would be an additional implementation burden.

Appendix A.3 New Encoding Convention

Instead of using the existing percent-encoding convention of URIs, which is based on octets, the idea was to create a new encoding convention, for example to use '%u' to introduce UCS code points.

Using the existing octet-based percent-encoding mechanism does not need an upgrade of the URI syntax, and does not need corresponding server upgrades.

Appendix A.4 Indicating Character Encodings in the URI/IRI

Some proposals suggested indicating the character encodings used in an URI or IRI with some new syntactic convention in the URI itself, similar to the 'charset' parameter for emails and Web pages. As an example, the label in square brackets in http://www.example.org/ros[iso-8859-1]é indicated that the following é had to be interpreted as iso-8859-1.

Using UTF-8 only does not need an upgrade to the URI syntax. It avoids potentially multiple labels that have to be copied correctly in all cases, even on the side of a bus or on a napkin, leading to usability problems to the extent of being prohibitively annoying. Using UTF-8 only also reduces transcoding errors and confusions.

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.