

Internationalized Resource Identifiers
(iri)
Internet-Draft
Obsoletes: 3987 (if approved)
Intended status: Standards Track
Expires: April 23, 2013

M. Duerst
Aoyama Gakuin University
M. Suignard
Unicode Consortium
L. Masinter
Adobe
October 20, 2012

Internationalized Resource Identifiers (IRIs)
draft-ietf-iri-3987bis-13

Abstract

This document defines the Internationalized Resource Identifier (IRI) protocol element, as an extension of the Uniform Resource Identifier (URI). An IRI is a sequence of characters from the Universal Character Set (Unicode/ISO 10646). Grammar and processing rules are given for IRIs and related syntactic forms.

Defining IRI as a new protocol element (rather than updating or extending the definition of URI) allows independent orderly transitions: protocols and languages that use URIs must explicitly choose to allow IRIs.

Guidelines are provided for the use and deployment of IRIs and related protocol elements when revising protocols, formats, and software components that currently deal only with URIs.

This document is part of a set of documents intended to replace RFC 3987.

RFC Editor: Please remove the next paragraph before publication.

This document, and several companion documents, are intended to obsolete RFC 3987. For discussion and comments on these drafts, please join the IETF IRI WG by subscribing to the mailing list public-iri@w3.org, archives at <http://lists.w3.org/archives/public/public-iri/>. For a list of open issues, please see the issue tracker of the WG at <http://trac.tools.ietf.org/wg/iri/trac/report/1>. For a list of individual edits, please see the change history at <http://trac.tools.ietf.org/wg/iri/trac/log/draft-ietf-iri-3987bis>.

This document is available in (line-printer ready) plaintext ASCII and PDF. It is also available in HTML from <http://www.sw.it.aoyama.ac.jp/2012/pub/draft-ietf-iri-3987bis-13.html>, and in UTF-8 plaintext from <http://>

www.sw.it.aoyama.ac.jp/2012/pub/draft-ietf-iri-3987bis-13.utf8.txt. While all these versions are identical in their technical content, the HTML, PDF, and UTF-8 plaintext versions show non-Unicode characters directly. This often makes it easier to understand examples, and readers are therefore advised to consult these versions in preference or as a supplement to the ASCII version.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on April 23, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format

it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	5
1.1.	Overview and Motivation	5
1.2.	Applicability	6
1.3.	Definitions	7
1.4.	Notation	8
2.	IRI Syntax	9
2.1.	Summary of IRI Syntax	9
2.2.	ABNF for IRI References and IRIs	10
3.	Processing IRIs and related protocol elements	12
3.1.	Converting to UCS	13
3.2.	Parse the IRI into IRI components	13
3.3.	General percent-encoding of IRI components	13
3.4.	Mapping ireg-name	14
3.4.1.	Mapping using Percent-Encoding	14
3.4.2.	Mapping using Punycode	14
3.4.3.	Additional Considerations	15
3.5.	Mapping query components	16
3.6.	Mapping IRIs to URIs	16
4.	Converting URIs to IRIs	16
4.1.	Limitations	16
4.2.	Conversion	17
4.3.	Examples	18
5.	Use of IRIs	19
5.1.	Limitations on UCS Characters Allowed in IRIs	19
5.2.	Software Interfaces and Protocols	20
5.3.	Format of URIs and IRIs in Documents and Protocols	20
5.4.	Use of UTF-8 for Encoding Original Characters	21
5.5.	Relative IRI References	22
6.	Legacy Extended IRIs (LEIRIs)	23
6.1.	Legacy Extended IRI Syntax	23
6.2.	Conversion of Legacy Extended IRIs to IRIs	23
6.3.	Characters Allowed in Legacy Extended IRIs but not in IRIs	23
7.	Processing of URIs/IRIs/URLs by Web Browsers	25
8.	URI/IRI Processing Guidelines (Informative)	26
8.1.	URI/IRI Software Interfaces	26
8.2.	URI/IRI Entry	26
8.3.	URI/IRI Transfer between Applications	27
8.4.	URI/IRI Generation	27
8.5.	URI/IRI Selection	28
8.6.	Display of URIs/IRIs	29
8.7.	Interpretation of URIs and IRIs	29

8.8. Upgrading Strategy	30
9. IANA Considerations	31
10. Security Considerations	31
11. Acknowledgements	32
12. Main Changes Since RFC 3987	32
12.1. Split out Bidi, processing guidelines, comparison sections	32
12.2. Major restructuring of IRI processing model	32
12.2.1. OLD WAY	33
12.2.2. NEW WAY	33
12.2.3. Extension of Syntax	33
12.2.4. More to be added	34
12.3. Change Log	34
12.3.1. Changes after draft-ietf-iri-3987bis-01	34
12.3.2. Changes from draft-duerst-iri-bis-07 to draft-ietf-iri-3987bis-00	34
12.3.3. Changes from -06 to -07 of draft-duerst-iri-bis	34
12.4. Changes from -00 to -01	34
12.5. Changes from -05 to -06 of draft-duerst-iri-bis-00	34
12.6. Changes from -04 to -05 of draft-duerst-iri-bis	35
12.7. Changes from -03 to -04 of draft-duerst-iri-bis	35
12.8. Changes from -02 to -03 of draft-duerst-iri-bis	35
12.9. Changes from -01 to -02 of draft-duerst-iri-bis	35
12.10. Changes from -00 to -01 of draft-duerst-iri-bis	35
12.11. Changes from RFC 3987 to -00 of draft-duerst-iri-bis	35
13. References	36
13.1. Normative References	36
13.2. Informative References	37
Authors' Addresses	39

1. Introduction

1.1. Overview and Motivation

A Uniform Resource Identifier (URI) is defined in [RFC3986] 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. This 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 those who use only the Latin script. Many languages with non-Latin scripts are transcribed with Latin letters. These transcriptions are now often used in URIs, but they introduce additional difficulties.

The infrastructure for the appropriate handling of characters from additional scripts is now widely deployed in operating system and application software. Software that can handle a wide variety of scripts and languages at the same time is increasingly common. Also, an increasing number of protocols and formats can carry a wide range of characters.

URIs are composed out of a very limited repertoire of characters; this design choice was made to support global transcription (see [RFC3986] section 1.2.1.). Reliable transition between a URI (as an abstract protocol element composed of a sequence of characters) and a presentation of that URI (written on a napkin, read out loud) and back is relatively straightforward, because of the limited repertoire of characters used. IRIs are designed to satisfy a different set of use requirements; in particular, to allow IRIs to be written in ways that are more meaningful to their users, even at the expense of global transcribability. However, ensuring reliability of the transition between an IRI and its presentation and back is more difficult and complex when dealing with the larger set of Unicode characters. For example, Unicode supports multiple ways of encoding complex combinations of characters and accents, with multiple character sequences that can result in the same presentation.

This document defines the protocol element called Internationalized Resource Identifier (IRI), which allows applications of URIs to be extended to use resource identifiers that have a much wider repertoire of characters. It also provides corresponding "internationalized" versions of other constructs from [RFC3986], such as URI references. The syntax of IRIs is defined in Section 2.

Within this document, Section 5 discusses the use of IRIs in different situations. Section 8 gives additional informative guidelines. Section 10 discusses IRI-specific security considerations.

This specification is part of a collection of specifications intended to replace [RFC3987]. [Bidi] discusses the special case of bidirectional IRIs, IRIs using characters from scripts written right-to-left. [Equivalence] gives guidelines for applications wishing to determine if two IRIs are equivalent, as well as defining some equivalence methods. [RFC4395bis] updates the URI scheme registration guidelines and procedures to note that every URI scheme is also automatically an IRI scheme and to allow scheme definitions to be directly described in terms of Unicode characters.

1.2. Applicability

IRIs are designed to allow protocols and software that deal with URIs to be updated to handle IRIs. Processing of IRIs is accomplished by extending the URI syntax while retaining (and not expanding) the set of "reserved" characters, such that the syntax for any URI scheme may be extended to allow non-ASCII characters. In addition, following parsing of an IRI, it is possible to construct a corresponding URI by first encoding characters outside of the allowed URI range and then reassembling the components.

Practical use of IRIs forms in place of URIs forms depends on the following conditions being met:

- a. A protocol or format element **MUST** be explicitly designated to be able to carry IRIs. The intent is to avoid introducing 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 used in attributes and elements of type "anyURI". On the other hand, in HTTP/1.1 ([RFC2616]), the Request URI is defined as a URI, which means that direct use of IRIs is not allowed in HTTP requests.
- b. The protocol or format carrying the IRIs **MUST** 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 scheme definition, if it explicitly allows a percent sign ("%") in any syntactic component, **SHOULD** define the interpretation of sequences of percent-encoded octets (using "%XX" hex octets) as octets from sequences of UTF-8 encoded characters; this is

recommended in the guidelines for registering new schemes, [RFC4395bis]. For example, this is the practice for IMAP URLs [RFC2192], POP URLs [RFC2384] and the URN syntax [RFC2141]). Note that use of percent-encoding may also be restricted in some situations, for example, URI schemes that disallow percent-encoding might still be used with a fragment identifier which is percent-encoded (e.g., [XPointer]). See Section 5.4 for further discussion.

1.3. Definitions

Various terms used in this document are defined in [RFC6365] and [RFC3986]. In addition, we define the following terms for use in this document.

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

sequence of characters: A sequence of characters (one after another).

sequence of octets: A sequence of octets (one after another).

character encoding: A method of representing a sequence of characters as a sequence of octets (maybe with variants). Also, 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 [UNIV6].

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 [RFC3986] fragment identifiers are part of URIs.

LEIRI (Legacy Extended IRI): This term is used in various XML specifications to refer to strings that, although not valid IRIs, are acceptable input to the processing rules in Section 6.2.

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

create (a URI or IRI): With respect to URIs and IRIs, the term 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 (a URI or IRI): With respect to URIs and IRIs, the term is used when the identifier is generated by derivation from other information.

parsed URI component: When a URI processor parses a URI (following the generic syntax or a scheme-specific syntax, the result is a set of parsed URI components, each of which has a type (corresponding to the syntactic definition) and a sequence of URI characters.

parsed IRI component: When an IRI processor parses an IRI directly, following the general syntax or a scheme-specific syntax, the result is a set of parsed IRI components, each of which has a type (corresponding to the syntactic definition) and a sequence of IRI characters. (This definition is analogous to "parsed URI component".)

IRI scheme: A URI scheme may also be known as an "IRI scheme" if the scheme's syntax has been extended to allow non-US-ASCII characters according to the rules in this document.

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 for 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 a document format that is limited to US-ASCII, this document uses 'XML Notation'. XML Notation uses a leading '&#x', a trailing ';', and the hexadecimal number of the character in the UCS in between. For example, Я stands for CYRILLIC CAPITAL LETTER YA. In this notation, an actual '&' is denoted by '&'. This notation is only used in the ASCII version(s) of this document, because in the other versions, non-ASCII characters are used directly.

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>.

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" 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 stored or transmitted digitally. The same IRI might 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 ensures that the characters in the IRI can be handled (e.g., searched, converted, displayed) in the same way as the rest of the protocol or document.

2.1. Summary of IRI Syntax

The IRI syntax extends the URI syntax in [RFC3986] by extending the class of unreserved characters, primarily 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 5.1.

The syntax and use of components and reserved characters is the same as that in [RFC3986]. Each URI scheme thus also functions as an IRI scheme, in that scheme-specific parsing rules for URIs of a scheme are extended to allow parsing of IRIs using the same parsing rules.

All the operations defined in [RFC3986], such as the resolution of relative references, can be applied to IRIs by IRI-processing software in exactly the same way as they are for URIs by URI-processing software.

Characters outside the US-ASCII repertoire MUST NOT be reserved and therefore MUST NOT be used for syntactical purposes, such as to delimit components in newly defined schemes. For example, U+00A2, CENT SIGN, is not allowed as a delimiter in IRIs, because it is in the 'unreserved' category. This is similar to the fact that it is

not possible to use '-' as a delimiter in URIs, because it is in the 'unreserved' category.

2.2. ABNF for IRI References and IRIs

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 [STD68]. Character numbers are taken from the UCS, without implying any actual binary encoding. Terminals in the ABNF are characters, not octets.

The following grammar closely follows the URI grammar in [RFC3986], 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. The grammar is split into two parts: Rules that differ from [RFC3986] because of the above-mentioned expansion, and rules that are the same as those in [RFC3986]. For rules that are different than those in [RFC3986], 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 rule <pct-form> has been introduced in order to be able to reference it from other parts of the document.

The following rules are different from those in [RFC3986]:

```

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-form / sub-delims / ":" )
ihost         = IP-literal / IPv4address / ireg-name

```

```

pct-form      = pct-encoded

ireg-name     = *( iunreserved / 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 = *( path-sep isegment )
ipath-absolute = path-sep [ isegment-nz *( path-sep isegment ) ]
ipath-noscheme = isegment-nz-nc *( path-sep isegment )
ipath-rootless = isegment-nz *( path-sep isegment )
ipath-empty    = ""
path-sep       = "/"

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

ipchar         = iunreserved / pct-form / 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 / %xE0000-E0FFF / %xF0000-FFFFD
              / %x100000-10FFFF

```

Some productions are ambiguous. The "first-match-wins" (a.k.a. "greedy") algorithm applies. For details, see [RFC3986].

The following rules are the same as those in [RFC3986]:

```

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. Processing IRIs and related protocol elements

IRIs are meant to replace URIs in identifying resources within new versions of protocols, formats, and software components that use a UCS-based character repertoire. Protocols and components may use and process IRIs directly. However, there are still numerous systems and

protocols which only accept URIs or components of parsed URIs; that is, they only accept sequences of characters within the subset of US-ASCII characters allowed in URIs.

This section defines specific processing steps for IRI consumers which establish the relationship between the string given and the interpreted derivatives. These processing steps apply to both IRIs and IRI references (i.e., absolute or relative forms); for IRIs, some steps are scheme specific.

3.1. Converting to UCS

Input that is already in a Unicode form (i.e., a sequence of Unicode characters or an octet-stream representing a Unicode-based character encoding such as UTF-8 or UTF-16) should be left as is and not normalized or changed.

An IRI or IRI reference is a sequence of characters from the UCS. For input from presentations (written on paper, read aloud) or translation from other representations (a text stream using a legacy character encoding), convert the input to Unicode. Note that some character encodings or transcriptions can be converted to or represented by more than one sequence of Unicode characters. Ideally the resulting IRI would use a normalized form, such as Unicode Normalization Form C [UTR15], since that ensures a stable, consistent representation that is most likely to produce the intended results. Previous versions of this specification required normalization at this step. However, attempts to require normalization in other protocols have met with strong enough resistance that requiring normalization here was considered impractical. Implementers and users are cautioned that, while denormalized character sequences are valid, they might be difficult for other users or processes to reproduce and might lead to unexpected results.

3.2. Parse the IRI into IRI components

Parse the IRI, either as a relative reference (no scheme) or using scheme specific processing (according to the scheme given); the result is a set of parsed IRI components.

3.3. General percent-encoding of IRI components

Except as noted in the following subsections, IRI components are mapped to the equivalent URI components by percent-encoding those characters not allowed in URIs. Previous processing steps will have removed some characters, and the interpretation of reserved characters will have already been done (with the syntactic reserved characters outside of the IRI component). This mapping is defined

for all sequences of Unicode characters, whether or not they are valid for the component in question.

For each character which is not allowed anywhere in a valid URI apply the following steps.

Convert to UTF-8: Convert the character to a sequence of one or more octets using UTF-8 [STD63].

Percent encode: Convert each octet of this sequence to %HH, where HH is the hexadecimal notation of the octet value. The hexadecimal notation SHOULD use uppercase letters. (This is the general URI percent-encoding mechanism in Section 2.1 of [RFC3986].)

Note that the mapping is an identity transformation for parsed URI components of valid URIs, and is idempotent: applying the mapping a second time will not change anything.

3.4. Mapping ireg-name

The mapping from <ireg-name> to a <reg-name> requires a choice between one of the two methods described below.

3.4.1. Mapping using Percent-Encoding

The ireg-name component SHOULD be converted according to the general procedure for percent-encoding of IRI components described in Section 3.3.

For example, the IRI
"http://résumé.example.org"
will be converted to
"http://r%C3%A9sum%C3%A9.example.org".

This conversion for ireg-name is in line with Section 3.2.2 of [RFC3986], which does not mandate a particular registered name lookup technology. For further background, see [RFC6055] and [Gettys].

3.4.2. Mapping using Punycode

In situations where it is certain that <ireg-name> is intended to be used as a domain name to be processed by Domain Name Lookup (as per [RFC5891]), an alternative method MAY be used, converting <ireg-name> as follows:

If there is any percent-encoding, and the corresponding octets all represent valid UTF-8 octet sequences, then convert these back to Unicode character sequences. (If any percent-encodings are not valid

UTF-8 octet sequences, then leave the entire field as is without any change, since punycode encoding would not succeed.)

Replace the ireg-name part of the IRI by the part converted using the Domain Name Lookup procedure (Subsections 5.3 to 5.5) of [RFC5891]. on each dot-separated label, and by using U+002E (FULL STOP) as a label separator. This procedure may fail, but this would mean that the IRI cannot be resolved. In such cases, if the domain name conversion fails, then the entire IRI conversion fails. Processors that have no mechanism for signalling a failure MAY instead substitute an otherwise invalid host name, although such processing SHOULD be avoided.

For example, the IRI
"http://résumé.example.org"
is converted to
"http://xn--rsum-bad.example.org".

This conversion for ireg-name will be better able to deal with legacy infrastructure that cannot handle percent-encoding in domain names.

3.4.3. Additional Considerations

Domain Names can appear 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 is responsible for making the necessary conversions to be able to retrieve the Web page.

In this process, characters allowed in URI references and existing percent-encoded sequences are not encoded further. (This mapping is similar to, but different from, the encoding applied when arbitrary content is included in 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"`.

3.5. Mapping query components

For compatibility with existing deployed HTTP infrastructure, the following special case applies for the schemes "http" and "https" when an IRI is found in a document whose charset is not based on UCS (e.g., not UTF-8 or UTF-16). In such a case, the "query" component of an IRI is mapped into a URI by using the document charset rather than UTF-8 as the binary representation before percent-encoding. This mapping is not applied for any other schemes or components.

3.6. Mapping IRIs to URIs

The mapping from an IRI to URI is accomplished by applying the mapping above (from IRI to URI components) and then reassembling a URI from the parsed URI components using the original punctuation that delimited the IRI components.

4. Converting URIs to IRIs

In some situations, for presentation and further processing, it is desirable to convert a URI into an equivalent IRI without unnecessary percent encoding. Of course, every URI is already an IRI in its own right without any conversion. This section gives one possible procedure for converting a URI to an IRI.

4.1. Limitations

The conversion described in this section, if given a valid URI, will result in an IRI that maps back to the URI 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 differ from 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:

1. Some percent-encodings are necessary to distinguish percent-encoded and unencoded uses of reserved characters.
2. 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].)

3. The conversion may result in a character that is not appropriate in an IRI. See Section 2.2, and Section 5.1 for further details.
4. As described in Section 3.5, IRI to URI conversion may work somewhat differently for query components.

4.2. Conversion

Conversion from a URI to an IRI MAY be done by using the following steps:

1. Represent the URI as a sequence of octets in US-ASCII.
2. Convert all percent-encodings ("% followed by two hexadecimal digits) to the corresponding octets, except those corresponding to "%, characters in "reserved", and characters in US-ASCII not allowed in URIs.
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 and Section 5.1.
5. Optionally, re-percent-encode octets in the query component if the scheme is one of those mentioned in Section 3.5.
6. Interpret the resulting octet sequence as a sequence of characters encoded in UTF-8.
7. URIs known to contain domain names in the reg-name component SHOULD convert punycode-encoded domain name labels to the corresponding characters using the ToUnicode procedure.

This procedure will convert as many percent-encoded characters as possible to characters in an IRI. Because there are some choices in steps 4 (see also Section 5.1) and 5, results may vary.

Conversions from URIs to IRIs MUST NOT use any character encoding other than UTF-8 in steps 3 and 4, even if it might be possible to guess from the context that another character encoding than UTF-8 was used in the URI. For 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, in the future the IRI will be mapped to "http://www.example.org/r%C3%A9sum%C3%A9.html", which is a different URI from "http://www.example.org/r%E9sum%E9.html".

4.3. Examples

This section shows various examples of converting URIs to IRIs. Each example shows the result after each of the steps 1 through 6 is applied. XML Notation is used for the final result. Octets are denoted by "<" followed by two hexadecimal digits followed by ">".

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
6. 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

6. `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. The direct use of this character is forbidden in an IRI. Therefore, the corresponding octets are re-percent-encoded in step 4. This example shows that the case (upper- or lowercase) of letters used in percent-encodings 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`
6. `http://納豆.example.org/%E2%80%AE`

Note that the label `"xn--99zt52a"` is converted to U+7D0D U+8C46 (Japanese Natto). ((EDITOR NOTE: There is some inconsistency in this note.))

5. Use of IRIs

5.1. 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. The considerations in this section are relevant when IRIs are created and when URIs are converted 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. It also includes many look-alikes of "space", "delims", and "unwise", characters excluded in [RFC3491].
- c. At the start of a component, the use of combining marks is strongly discouraged. As an example, a COMBINING TILDE OVERLAY (U+0334) would be very confusing at the start of a <isegment>. Combined with the preceeding '/', it might look like a solidus with combining tilde overlay, but IRI processing software will parse and process the '/' separately.
- d. The ZERO WIDTH NON-JOINER (U+200C) and ZERO WIDTH JOINER (U+200D) are invisible in most contexts, but are crucial in some very limited contexts. Appendix A of [RFC5892] contains contextual restrictions for these and some other characters. The use of these characters are strongly discouraged except in the relevant contexts.

Additional information is available from [UNIXML]. [UNIXML] is written in the context of general purpose text rather than in that of identifiers. Nevertheless, it discusses many of the categories of characters not appropriate for IRIs.

5.2. 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 3.6, when transferring from IRI-capable to URI-only components. This mapping SHOULD be applied as late as possible. It SHOULD NOT be applied between components that are known to be able to handle IRIs.

5.3. Format of URIs and IRIs in Documents and Protocols

Document formats that transport URIs may have to be upgraded to allow the transport of IRIs. In 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 not expressible in the native character encoding SHOULD be escaped by using the escaping conventions of the

document format if such conventions are available. Alternatively, they MAY be percent-encoded according to Section 3.6. 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.

((UPDATE THIS NOTE)) 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], XML Schema [XMLSchema], and specifications based upon them allow IRIs. Also, it is expected that all relevant new W3C formats and protocols will be required to handle IRIs [CharMod].

5.4. Use of UTF-8 for Encoding Original Characters

This section discusses details and gives examples for point c) in Section 1.2. To be able to use IRIs, the URI corresponding to the IRI in question has to encode original characters into octets by 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 to some part. For background information on encoding characters into URIs, see also Section 2.5 of [RFC3986].

For new URI/IRI schemes, using UTF-8 is recommended in [RFC4395bis]. Examples where UTF-8 is already used are the URN syntax [RFC2141], IMAP URLs [RFC2192], POP URLs [RFC2384], XMPP URLs [RFC5122], and the 'mailto:' scheme [RFC6068]. On the other hand, because the HTTP URI 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-encoded octets cannot be converted to actual characters in an IRI, as the percent-encoding is not based on UTF-8.

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 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.

Scheme definitions can impose restrictions on the syntax of scheme-specific URIs; i.e., URIs that are admissible under the generic URI syntax [RFC3986] may not be admissible 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, as the corresponding URI resulting from the mapping defined in Section 3.6 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 generally applies to all parts of a URI. 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 where 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.

5.5. Relative IRI References

Processing of relative IRI references against a base is handled straightforwardly; the algorithms of [RFC3986] can be applied directly, treating the characters additionally allowed in IRI references in the same way that unreserved characters are treated in URI references.

6. Legacy Extended IRIs (LEIRIs)

In some cases, there have been formats which have used a protocol element which is a variant of the IRI definition; these variants have usually been somewhat less restricted in syntax. This section provides a definition and a name (Legacy Extended IRI or LEIRI) for one of these variants used widely in XML-based protocols. This variant has to be used with care; it requires further processing before being fully interchangeable as IRIs. New protocols and formats SHOULD NOT use Legacy Extended IRIs. Even where Legacy Extended IRIs are allowed, only IRIs fully conforming to the syntax definition in Section 2.2 SHOULD be created, generated, and used. The provisions in this section also apply to Legacy Extended IRI references.

6.1. Legacy Extended IRI Syntax

This section defines Legacy Extended IRIs (LEIRIs). The syntax of Legacy Extended IRIs is the same as that for <IRI-reference>, except that the ucschar production is replaced by the leiri-ucschar production:

```
leiri-ucschar = " " / "<" / ">" / "'" / "{" / "}" / "|"
               / "\" / "^" / "`" / %x0-1F / %x7F-D7FF
               / %xE000-FFFD / %x10000-10FFFF
```

The restriction on bidirectional formatting characters in [Bidi] is lifted. The iprivate production becomes redundant.

Likewise, the syntax for Legacy Extended IRI references (LEIRI references) is the same as that for IRI references with the above replacement of ucschar with leiri-ucschar.

6.2. Conversion of Legacy Extended IRIs to IRIs

To convert a Legacy Extended IRI (reference) to an IRI (reference), each character allowed in a Legacy Extended IRI (reference) but not allowed in an IRI (reference) (see Section 6.3) MUST be percent-encoded by applying the steps in Section 3.3.

6.3. Characters Allowed in Legacy Extended IRIs but not in IRIs

This section provides a list of the groups of characters and code points that are allowed in Legacy Extended IRIs, but are not allowed in IRIs or are allowed in IRIs only in the query part. For each group of characters, advice on the usage of these characters is also given, concentrating on the reasons for why not to use them.

Space (U+0020): Some formats and applications use space as a delimiter, e.g., for items in a list. Appendix C of [RFC3986] also mentions that white space may have to be added when displaying or printing long URIs; the same applies to long IRIs. Spaces might disappear, or a single Legacy Extended IRI might incorrectly be interpreted as two or more separate ones.

Delimiters "<" (U+003C), ">" (U+003E), and "'" (U+0022): Appendix C of [RFC3986] suggests the use of double-quotes ("http://example.com/") and angle brackets (<http://example.com/>) as delimiters for URIs in plain text. These conventions are often used, and also apply to IRIs. Legacy Extended IRIs using these characters might be cut off at the wrong place.

Unwise characters "\" (U+005C), "^" (U+005E), "`" (U+0060), "{" (U+007B), "|" (U+007C), and "}" (U+007D): These characters originally were excluded from URIs because the respective codepoints are assigned to different graphic characters in some 7-bit or 8-bit encoding. Despite the move to Unicode, some of these characters are still occasionally displayed differently on some systems, e.g., U+005C as a Japanese Yen symbol. Also, the fact that these characters are not used in URIs or IRIs has encouraged their use outside URIs or IRIs in contexts that may include URIs or IRIs. In case a Legacy Extended IRI with such a character is used in such a context, the Legacy Extended IRI will be interpreted piecemeal.

The controls (C0 controls, DEL, and C1 controls, #x0 - #x1F #x7F - #x9F): There is no way to transmit these characters reliably except potentially in electronic form. Even when in electronic form, some software components might silently filter out some of these characters, or may stop processing altogether when encountering some of them. These characters may affect text display in subtle, unnoticable ways or in drastic, global, and irreversible ways depending on the hardware and software involved. The use of some of these characters may allow malicious users to manipulate the display of a Legacy Extended IRI and its context.

Bidi formatting characters (U+200E, U+200F, U+202A-202E): These characters affect the display ordering of characters. Displayed Legacy Extended IRIs containing these characters cannot be converted back to electronic form (logical order) unambiguously. These characters may allow malicious users to manipulate the display of a Legacy Extended IRI and its context.

Specials (U+FFF0-FFFD): These code points provide functionality beyond that useful in a Legacy Extended IRI, for example byte order identification, annotation, and replacements for unknown

characters and objects. Their use and interpretation in a Legacy Extended IRI serves no purpose and may lead to confusing display variations.

Private use code points (U+E000-F8FF, U+F0000-FFFFD, U+100000-10FFFFD): Display and interpretation of these code points is by definition undefined without private agreement. Therefore, these code points are not suited for use on the Internet. They are not interoperable and may have unpredictable effects.

Tags (U+E0000-E0FFF): These characters provide a way to language tag in Unicode plain text. They are not appropriate for Legacy Extended IRIs because language information in identifiers cannot reliably be input, transmitted (e.g., on a visual medium such as paper), or recognized.

Non-characters (U+FDD0-FDEF, U+1FFFE-1FFFF, U+2FFFE-2FFFF, U+3FFFE-3FFFF, U+4FFFE-4FFFF, U+5FFFE-5FFFF, U+6FFFE-6FFFF, U+7FFFE-7FFFF, U+8FFFE-8FFFF, U+9FFFE-9FFFF, U+AFFFE-AFFFF, U+BFFFE-BFFFF, U+CFFFE-CFFFF, U+DFFFE-DFFFF, U+EFFFE-EFFFF, U+FFFFE-FFFFF, U+10FFFE-10FFFF): These code points are defined as non-characters. Applications may use some of them internally, but are not prepared to interchange them.

For reference, we here also list the code points and code units not even allowed in Legacy Extended IRIs:

Surrogate code units (D800-DFFF): These do not represent Unicode codepoints.

Non-characters (U+FFFE-FFFF): These are not allowed in XML nor LEIRIs.

7. Processing of URIs/IRIs/URLs by Web Browsers

For legacy reasons, many web browsers exhibit some irregularities when processing URIs, IRIs, and URLs. This is being documented in [HTMLURL], in the hope that it will lead to more uniform implementations of these irregularities across web browsers.

As far as currently known, creators of content for web browsers (such as HTML) can use all URIs without problems. They can also use all IRIs without problems except that they should be aware of the fact that query parts for HTTP/HTTPS IRIs should be percent-escaped.

8. 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 modification before functioning properly with IRIs. The considerations in this section also apply to URI references and IRI references.

8.1. 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, as it is compatible with US-ASCII, is in accordance with the recommendations of [RFC2277], and makes converting to URIs easy. 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 4 above may be performed. It is preferable not to perform this inverse conversion unless it is certain this can be done correctly.

8.2. URI/IRI Entry

Some components allow users to enter URIs into the system by typing or dictation, for example. This software must be updated to allow for IRI entry.

A person viewing a visual presentation of an IRI (as a sequence of glyphs, in some order, in some visual display) will use an 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 ensure, as much 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 solely used for the input of URIs/IRIs might allow the user to view an IRI as it is 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 interfacing to components that handle URIs, but not IRIs, must map the IRI to a URI before passing it to these components.

For the input of IRIs with right-to-left characters, please see [Bidi].

8.3. URI/IRI Transfer between Applications

Many applications (for example, 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 would need to be upgraded, in order to use the IRI 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 would need 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 3.6) 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 octets, which will do the right thing with IRIs.

8.4. 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 make 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 use of "résumé.html" in an IRI, even if locally the file name is kept in a character encoding other than UTF-8.

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

8.5. URI/IRI Selection

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

In these 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 "lame" identifiers.

Outside 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, newly created resource names should generally 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 similarity of the Latin "A", the Greek "Alpha", and the Cyrillic "A". To avoid such cases, IRIs should only be created where all the characters in a single component are used together in a given language. This usually means that all of 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 lowercase letters results in fewer ambiguities than using uppercase letters would.

8.6. 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 [Bidi].

8.7. 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 character encodings other than UTF-8. These URIs may be produced by user agents that do not conform to this specification and that 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 some servers currently 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. Please note that such mappings, including case mappings, are language dependent.

It can be difficult to identify a resource unambiguously if too many mappings are taken into consideration. However, percent-encoded and not percent-encoded parts of IRIs can always be clearly distinguished. Also, the regularity of UTF-8 (see [Duerst97]) makes the potential for collisions lower than it may seem at first.

8.8. 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, although 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. However, before an individual IRI is published, 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.

Where there is a free choice of character encodings, it is often possible to reduce the effort and dependencies for upgrading to IRIs by using UTF-8 rather than another encoding. For example, when a new file-based Web server is set up, using UTF-8 as the character encoding for file names will make the transition to IRIs easier. Likewise, when a new Web form is set up 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, see Section 5.4.

9. IANA Considerations

This specification does not affect IANA. For details on how to define a URI/IRI scheme and register it with IANA, see [RFC4395bis].

10. Security Considerations

The security considerations discussed in [RFC3986] also apply to IRIs. In addition, the following issues require particular care for IRIs.

Incorrect encoding or decoding can lead to security problems. For example, some UTF-8 decoders do not check against overlong byte sequences. See [UTR36] Section 3 for details.

There are serious difficulties with relying on a human to verify that a an IRI (whether presented visually or aurally) is the same as another IRI or is the one intended. These problems exist with ASCII-only URIs (b100mberg.com vs. bloomberg.com) but are strongly exacerbated when using the much larger character repertoire of Unicode. For details, see Section 2 of [UTR36]. Using administrative and technical means to reduce the availability of such exploits is possible, but they are difficult to eliminate altogether. User agents SHOULD NOT rely on visual or perceptual comparison or verification of IRIs as a means of validating or assuring safety, correctness or appropriateness of an IRI. Other means of presenting users with the validity, safety, or appropriateness of visited sites are being developed in the browser community as an alternative means of avoiding these difficulties.

Besides the large character repertoire of Unicode, reasons for confusion include different forms of normalization and different normalization expectations, use of percent-encoding with various legacy encodings, and bidirectionality issues. See also [Bidi].

Confusion can occur in various IRI components, such as the domain name part or the path part, or between IRI components. For considerations specific to the domain name part, see [RFC5890]. For considerations specific to particular protocols or schemes, see the security sections of the relevant specifications and registration templates. Administrators of sites that allow independent users to

create resources in the same sub area have to be careful. Details are discussed in Section 8.5.

The characters additionally allowed in Legacy Extended IRIs introduce additional security issues. For details, see Section 6.3.

11. Acknowledgements

This document was derived from [RFC3987]; the acknowledgments from that specification still apply.

In addition, this document was influenced by contributions from (in no particular order) Norman Walsh, Richard Tobin, Henry S. Thomson, John Cowan, Paul Grosso, the XML Core Working Group of the W3C, Chris Lilley, Bjoern Hoehrmann, Felix Sasaki, Jeremy Carroll, Frank Ellermann, Michael Everson, Cary Karp, Matitiah Allouche, Richard Ishida, Addison Phillips, Jonathan Rosenne, Najib Tounsi, Debbie Garside, Mark Davis, Sarmad Hussain, Ted Hardie, Konrad Lanz, Thomas Roessler, Lisa Dusseault, Julian Reschke, Giovanni Campagna, Anne van Kesteren, Mark Nottingham, Erik van der Poel, Marcin Hanclik, Marcos Caceres, Roy Fielding, Greg Wilkins, Pieter Hintjens, Daniel R. Tobias, Marko Martin, Maciej Stanchowiak, Wil Tan, Yui Naruse, Michael A. Puls II, Dave Thaler, Tom Petch, John Klensin, Shawn Steele, Peter Saint-Andre, Geoffrey Sneddon, Chris Weber, Alex Melnikov, Slim Amamou, S. Moonesamy, Tim Berners-Lee, Yaron Goland, Sam Ruby, Adam Barth, Abdulrahman I. ALGhadir, Aharon Lanin, Thomas Milo, Murray Sargent, Marc Blanchet, and Mykyta Yevstifeyev.

Anne van Kesteren is also gratefully acknowledged for his ongoing work documenting browser behavior with respect to URIs/URIs/URLs (see [HTMLURL]).

12. Main Changes Since RFC 3987

This section describes the main changes since [RFC3987].

12.1. Split out Bidi, processing guidelines, comparison sections

Move some components (comparison, bidi, processing) into separate documents.

12.2. Major restructuring of IRI processing model

Major restructuring of IRI processing model to make scheme-specific translation necessary to handle IDNA requirements and for consistency with web implementations.

Starting with IRI, you want one of:

- a IRI components (IRI parsed into UTF8 pieces)
- b URI components (URI parsed into ASCII pieces, encoded correctly)
- c whole URI (for passing on to some other system that wants whole URIs)

12.2.1. OLD WAY

1. Percent-encoding on the whole thing to a URI. (c1) If you want a (maybe broken) whole URI, you might stop here.
2. Parsing the URI into URI components. (b1) If you want (maybe broken) URI components, stop here.
3. Decode the components (undoing the percent-encoding). (a) if you want IRI components, stop here.
4. reencode: Either using a different encoding some components (for domain names, and query components in web pages, which depends on the component, scheme and context), and otherwise using percent-encoding. (b2) if you want (good) URI components, stop here.
5. reassemble the reencoded components. (c2) if you want a (*good*) whole URI stop here.

12.2.2. NEW WAY

1. Parse the IRI into IRI components using the generic syntax. (a) if you want IRI components, stop here.
2. Encode each components, using percent-encoding, IDN encoding, or special query part encoding depending on the component scheme or context. (b) If you want URI components, stop here.
3. reassemble the a whole URI from URI components. (c) if you want a whole URI stop here.

12.2.3. Extension of Syntax

Added the tag range (U+E0000-E0FFF) to the iprivate production. Some IRIs generated with the new syntax may fail to pass very strict checks relying on the old syntax. But characters in this range should be extremely infrequent anyway.

12.2.4. More to be added

TODO: There are more main changes that need to be documented in this section.

12.3. Change Log

Note to RFC Editor: Please completely remove this section before publication.

12.3.1. Changes after draft-ietf-iri-3987bis-01

Changes from draft-ietf-iri-3987bis-01 onwards are available as changesets in the IETF tools subversion repository at <http://trac.tools.ietf.org/wg/iri/trac/log/draft-ietf-iri-3987bis/draft-ietf-iri-3987bis.xml>.

12.3.2. Changes from draft-duerst-iri-bis-07 to draft-ietf-iri-3987bis-00

Changed draft name, date, last paragraph of abstract, and titles in change log, and added this section in moving from draft-duerst-iri-bis-07 (personal submission) to draft-ietf-iri-3987bis-00 (WG document).

12.3.3. Changes from -06 to -07 of draft-duerst-iri-bis

Major restructuring of the processing model, see Section 12.2.

12.4. Changes from -00 to -01

- o Removed 'mailto:' before mail addresses of authors.
- o Added "<to be done>" as right side of 'href-strip' rule. Fixed '|' to '/' for alternatives.

12.5. Changes from -05 to -06 of draft-duerst-iri-bis-00

- o Add HyperText Reference, change abstract, acks and references for it
- o Add Masinter back as another editor.
- o Masinter integrates HRef material from HTML5 spec.
- o Rewrite introduction sections to modernize.

12.6. Changes from -04 to -05 of draft-duerst-iri-bis

- o Updated references.
- o Changed IPR text to pre5378Trust200902.

12.7. Changes from -03 to -04 of draft-duerst-iri-bis

- o Added explicit abbreviation for LEIRIs.
- o Mentioned LEIRI references.
- o Completed text in LEIRI section about tag characters and about specials.

12.8. Changes from -02 to -03 of draft-duerst-iri-bis

- o Updated some references.
- o Updated Michel Suginard's coordinates.

12.9. Changes from -01 to -02 of draft-duerst-iri-bis

- o Added tag range to iprivate (issue private-include-tags-115).
- o Added Specials (U+FFFF0-FFFFD) to Legacy Extended IRIs.

12.10. Changes from -00 to -01 of draft-duerst-iri-bis

- o Changed from "IRIs with Spaces/Controls" to "Legacy Extended IRI" based on input from the W3C XML Core WG. Moved the relevant subsections to the back and promoted them to a section.
- o Added some text re. Legacy Extended IRIs to the security section.
- o Added a IANA Consideration Section.
- o Added this Change Log Section.
- o Added a section about "IRIs with Spaces/Controls" (converting from a Note in RFC 3987).

12.11. Changes from RFC 3987 to -00 of draft-duerst-iri-bis

Fixed errata (see
<http://www.rfc-editor.org/cgi-bin/errataSearch.pl?rfc=3987>).

13. References

13.1. 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 10646:2011: Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO Standard 10646, March 2001, <[http://standards.iso.org/ittf/PubliclyAvailableStandards/c051273_ISO_IEC_10646_2011\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c051273_ISO_IEC_10646_2011(E).zip)>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.
- [RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.
- [STD63] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [STD68] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [UNIV6] The Unicode Consortium, "The Unicode Standard, Version 6.2.0 (Mountain View, CA, The Unicode Consortium, 2012, ISBN 978-1-936213-07-8)", October 2012.

- [UTR15] Davis, M. and M. Duerst, "Unicode Normalization Forms", Unicode Standard Annex #15, March 2008, <<http://www.unicode.org/unicode/reports/tr15/tr15-23.html>>.

13.2. Informative References

- [Bidi] Duerst, M., Masinter, L., and A. Allawi, "Guidelines for Internationalized Resource Identifiers with Bi-directional Characters (Bidi IRIs)", draft-ietf-iri-bidi-guidelines-02 (work in progress), March 2012.
- [CharMod] Duerst, M., Yergeau, F., Ishida, R., Wolf, M., and T. Texin, "Character Model for the World Wide Web 1.0: Resource Identifiers", W3C Candidate Recommendation CR-charmod-resid-20041122, November 2004, <<http://www.w3.org/TR/2004/CR-charmod-resid/>>.
- [Duerst97] Duerst, M., "The Properties and Promises of UTF-8", Proc. 11th International Unicode Conference, San Jose , September 1997, <<http://www.sw.it.aoyama.ac.jp/2012/pub/IUC11-UTF-8.pdf>>.
- [Equivalence] Masinter, L. and M. Duerst, "Equivalence and Canonicalization of Internationalized Resource Identifiers (IRIs)", draft-ietf-iri-comparison-01 (work in progress), March 2012.
- [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", W3C Recommendation REC-html401-19991224, December 1999, <<http://www.w3.org/TR/1999/REC-html401>>.
- [HTMLURL] van Kesteren, A., "URL", October 2012, <<http://url.spec.whatwg.org/>>.
- [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.
- [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., Frystyk, 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.
- [RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.
- [RFC4395bis]
Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI/IRI Schemes", draft-ietf-iri-4395bis-irireg-04 (work in progress), December 2011.
- [RFC5122] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)", RFC 5122, February 2008.
- [RFC6055] Thaler, D., Klensin, J., and S. Cheshire, "IAB Thoughts on Encodings for Internationalized Domain Names", RFC 6055, February 2011.
- [RFC6068] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, October 2010.
- [RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, September 2011.
- [UNIXML] Duerst, M. and A. Freytag, "Unicode in XML and other Markup Languages", Unicode Technical Report #20, World Wide Web Consortium Note, June 2003, <<http://www.w3.org/TR/unicode-xml/>>.
- [UTR36] Davis, M. and M. Suignard, "Unicode Security Considerations", Unicode Technical Report #36, August 2010, <<http://unicode.org/reports/tr36/>>.

- [XLink] DeRose, S., Maler, E., Orchard, D., and N. Walsh, "XML Linking Language (XLink) Version 1.1", W3C Recommendation REC-xlink11-20100506, May 2010, <<http://www.w3.org/TR/xlink11/#link-locators>>.
- [XML1] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3C Recommendation REC-xml-20081126, November 2008, <<http://www.w3.org/TR/2008/REC-xml/>>.
- [XMLSchema] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation REC-xmlschema-2-20041028, October 2004, <<http://www.w3.org/TR/xmlschema-2/#anyURI>>.
- [XPointer] Grosso, P., Maler, E., Marsh, J., and N. Walsh, "XPointer Framework", W3C Recommendation REC-xptr-framework-20030325, March 2003, <<http://www.w3.org/TR/xptr-framework/#escaping>>.

Authors' Addresses

Martin J. Duerst (Note: Please write "Duerst" with u-umlaut wherever possible, for example as "Dürst" in XML and HTML.)

Aoyama Gakuin University
5-10-1 Fuchinobe
Chuo-ku
Sagamihara, Kanagawa 252-5258
Japan

Phone: +81 42 759 6329
Fax: +81 42 759 6495
Email: duerst@it.aoyama.ac.jp
URI: <http://www.sw.it.aoyama.ac.jp/D%C3%BCrst/>
(Note: This is the percent-encoded form of an IRI)

Michel Suignard
Unicode Consortium
P.O. Box 391476
Mountain View, CA 94039-1476
U.S.A.

Phone: +1-650-693-3921
Email: michel@unicode.org
URI: <http://www.suignard.com>

Larry Masinter
Adobe
345 Park Ave
San Jose, CA 95110
U.S.A.

Phone: +1-408-536-3024
Email: masinter@adobe.com
URI: <http://larry.masinter.net>

Network Working Group
Internet-Draft
Obsoletes: 4395 (if approved)
Intended status: BCP
Expires: June 16, 2012

T. Hansen
AT&T Laboratories
T. Hardie
Panasonic Wireless Research Lab
L. Masinter
Adobe
December 14, 2011

Guidelines and Registration Procedures for New URI/IRI Schemes
draft-ietf-iri-4395bis-irireg-04

Abstract

This document updates the guidelines and recommendations for the definition of Uniform Resource Identifier (URI) schemes, and extends the registry and guidelines to apply when the schemes are used with Internationalized Resource Identifiers (IRIs). It also updates the process and IANA registry for URI/IRI schemes. It obsoletes RFC 4395.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on June 16, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conformance Guidelines	4
3. Guidelines for Permanent URI/IRI Scheme Definitions	4
3.1. Demonstratable, New, Long-Lived Utility	4
3.2. Syntactic Compatibility	5
3.3. Well-Defined	5
3.4. Definition of Operations	6
3.5. Context of Use	7
3.6. Internationalization and Character Encoding	7
3.7. Clear Security Considerations	7
3.8. Scheme Name Considerations	8
4. Guidelines for Provisional URI/IRI Scheme Registration	8
5. Guidelines for Historical URI/IRI Scheme Registration	9
6. URI/IRI Scheme Registration Procedure	9
6.1. General	9
6.2. Registration Procedures	10
6.3. Change Control	11
6.4. URI/IRI Scheme Registration Template	11
7. The "example" Scheme	12
8. IANA Considerations	13
9. Security Considerations	13
10. Acknowledgements	14
Appendix A. Changes Since RFC 4395	14
11. References	14
11.1. Normative References	14
11.2. Informative References	15
Authors' Addresses	15

1. Introduction

The Uniform Resource Identifier (URI) protocol element and generic syntax is defined by [RFC3986]. Each URI begins with a scheme name, as defined by Section 3.1 of RFC 3986, that refers to a specification for identifiers within that scheme. The URI syntax provides a federated and extensible naming system, where each scheme's specification may further restrict the syntax and define the semantics of identifiers using that scheme. As originally defined, URIs only allowed a limited repertoire of characters chosen from US-ASCII.

An Internationalized Resource Identifier (IRI), as defined by [RFC3987bis], extends the URI syntax to allow characters from a much greater repertoire, to accomodate resource identifiers from the world's languages. The same schemes used in URIs are used in IRIs. The term Resource Identifier (RI) is used as a shorthand for both URIs and IRIs. [RFC3987] introduced IRIs by defining a mapping between URIs and IRIs; [RFC3987bis] updates that definition, allowing an IRI to be interpreted directly without translating into a URI.

This document obsoletes [RFC4395], which in turn obsoleted [RFC2717] and [RFC2718]. Recent documents have used the terms "URI"/"IRI" for all resource identifiers, avoiding the term "URL" and reserving the term "URN" explicitly for those URIs/IRIs using the "urn" scheme name ([RFC2141]). URN "namespaces" ([RFC3406]) are specific to the "urn" scheme and are not covered explicitly by this specification.

This document extends the URI scheme registry to be a registry of URI/IRI schemes (i.e., applicable to both URIs and IRIs). This document also provides updated guidelines for the definition of new schemes, for consideration by those who are defining, registering, or evaluating those definitions, as well as a process and mechanism for registering URI/IRI schemes within the IANA URI scheme registry. There is a single namespace for registered schemes. Within that namespace, there are values that are approved as meeting a set of criteria for permanent URI/IRI schemes. Other scheme names may also be registered provisionally or historically, without necessarily meeting those criteria. The intent of the registry is to:

- o provide a central point of discovery for established URI/IRI scheme names, and easy location of their defining documents;
- o discourage use of the same scheme name for different purposes;
- o help those proposing new scheme names to discern established trends and conventions, and avoid names that might be confused with existing ones;

- o encourage registration by setting a low barrier for provisional registrations.

There is no separate, independent registry or registration process for IRIs: the URI Scheme Registry is to be used for both URIs and IRIs. Previously, those who wish to describe resource identifiers that are useful as IRIs were encouraged to define the corresponding URI syntax, and note that the IRI usage follows the rules and transformations defined in [RFC3987]. This document changes that advice to encourage explicit definition of the scheme and allowable syntax elements within the larger character repertoire of IRIs, as defined by [RFC3987bis].

A scheme definition cannot override the overall syntax for IRIs. For example, this means that fragment identifiers (#) cannot be re-used outside the generic syntax restrictions, and in particular scheme-specific syntax cannot override the fragment identifier syntax because it is generic.

2. Conformance Guidelines

Within this document, the key words MUST, MAY, SHOULD, REQUIRED, RECOMMENDED, and so forth are used within the general meanings established in [RFC2119], within the context that they are requirements on future registration specifications.

3. Guidelines for Permanent URI/IRI Scheme Definitions

This section gives considerations for new URI/IRI schemes. Meeting these guidelines is REQUIRED for permanent scheme registration. Meeting these guidelines is also RECOMMENDED for provisional registration, as described in Section 4.

3.1. Demonstratable, New, Long-Lived Utility

The use and deployment of new URI/IRI schemes in the Internet infrastructure may be costly; some parts of URI/IRI processing may be scheme-dependent, and deployed software already processes URIs and IRIs of well-known schemes. Introducing a new scheme may require additional software, not only for client software and user agents but also in additional parts of the network infrastructure (gateways, proxies, caches) [W3CWebArch]. URI/IRI schemes constitute a single, global namespace; it is desirable to avoid contention over use of short, mnemonic scheme names. For these reasons, the unbounded registration of new schemes is harmful. New schemes should have utility to the Internet community beyond that available with already

registered schemes. The registration document SHOULD discuss the utility of the scheme being registered.

3.2. Syntactic Compatibility

[RFC3986] defines the generic syntax for all URI schemes, along with the syntax of common URI components that are used by many URI schemes to define hierarchical identifiers. [RFC3987] and subsequently [RFC3987bis] extended this generic syntax to cover IRIs. All URI/IRI scheme specifications MUST define their own syntax such that all strings matching their scheme-specific syntax will also match the <absolute-URI> grammar described in [RFC3987bis].

New schemes SHOULD reuse the common components of [RFC3987bis] for the definition of hierarchical naming schemes. However, if there is a strong reason for a scheme not to use the hierarchical syntax, then the new scheme definition SHOULD follow the syntax of previously registered schemes.

Schemes that are not intended for use with relative URIs/IRIs SHOULD avoid use of the forward slash "/" character, which is used for hierarchical delimiters, and the complete path segments "." and ".." (dot-segments).

Avoid improper use of "//". The use of double slashes in the first part of a URI/IRI is not an artistic indicator that what follows is a URI/IRI: Double slashes are used ONLY when the syntax of the <scheme-specific-part> contains a hierarchical structure. In URIs and IRIs from such schemes, the use of double slashes indicates that what follows is the top hierarchical element for a naming authority. (Section 3.2 of RFC 3986 has more details.) Schemes that do not contain a conformant hierarchical structure in their <scheme-specific-part> SHOULD NOT use double slashes following the "<scheme>:" string.

New schemes SHOULD clearly define the role of [RFC3986] reserved characters in URIs/IRIs of the scheme being defined. The syntax of the new scheme should be clear about which of the "reserved" set of characters are used as delimiters within the URIs/IRIs of the new scheme, and when those characters must be escaped, versus when they may be used without escaping.

3.3. Well-Defined

While URIs/IRIs may or may not be defined as locators in practice, a scheme definition itself MUST be clear as to how it is expected to function. Schemes that are not intended to be used as locators SHOULD describe how the resource identified can be determined or

accessed by software that obtains a URI/IRI of that scheme.

For schemes that function as locators, it is important that the mechanism of resource location be clearly defined. This might mean different things depending on the nature of the scheme.

In many cases, new schemes are defined as ways to translate between other namespaces or protocols and the general framework of URIs. For example, the "ftp" scheme translates into the FTP protocol, while the "mid" scheme translates into a Message-ID identifier of an email message. For such schemes, the description of the mapping must be complete, and in sufficient detail so that the mapping in both directions is clear: how to map from a URI/IRI into an identifier or set of protocol actions or name in the target namespace, and how legal values in the base namespace, or legal protocol interactions, might be represented in a valid URI or IRI. In particular, the mapping should describe the mechanisms for encoding binary or character strings within valid character sequences in a URI/IRI (See Section 3.6 for guidelines). If not all legal values or protocol interactions of the base standard can be represented using the scheme, the definition should be clear about which subset are allowed, and why.

3.4. Definition of Operations

As part of the definition of how a URI/IRI identifies a resource, a scheme definition SHOULD define the applicable set of operations that may be performed on a resource using the RI as its identifier. A model for this is HTTP; an HTTP resource can be operated on by GET, POST, PUT, and a number of other operations available through the HTTP protocol. The scheme definition should describe all well-defined operations on the resource identifier, and what they are supposed to do.

Some schemes don't fit into the "information access" paradigm of URIs/IRIs. For example, "telnet" provides location information for initiating a bi-directional data stream to a remote host; the only operation defined is to initiate the connection. In any case, the operations appropriate for a scheme should be documented.

Note: It is perfectly valid to say that "no operation apart from GET is defined for this RI". It is also valid to say that "there's only one operation defined for this RI, and it's not very GET-like". The important point is that what is defined on this scheme is described.

3.5. Context of Use

In general, URIs/IRIs are used within a broad range of protocols and applications. Most commonly, URIs/IRIs are used as references to resources within directories or hypertext documents, as hyperlinks to other resources. In some cases, a scheme is intended for use within a different, specific set of protocols or applications. If so, the scheme definition SHOULD describe the intended use and include references to documentation that define the applications and/or protocols cited.

3.6. Internationalization and Character Encoding

When describing schemes in which (some of) the elements of the URI or IRI are actually representations of human-readable text, care should be taken not to introduce unnecessary variety in the ways in which characters are encoded into octets and then into characters; see [RFC3987bis] and Section 2.5 of [RFC3986] for guidelines. If URIs/IRIs of a scheme contain any text fields, the scheme definition MUST describe the ways in which characters are encoded and any compatibility issues with IRIs of the scheme.

Specifications for IRIs schemes MUST be described in terms of processing an IRI as a sequence of Unicode codepoints, without reference to the encoding of those code points as a sequence of bytes, using UTF-8 or UTF-16. The scheme specification SHOULD be as restrictive as possible regarding what characters are allowed in the URI/IRI, because some characters can create several different security considerations (see for example [RFC4690]).

All percent-encoded variants are automatically included by definition for any character given in an IRI production. This means that if you want to restrict the URI percent-encoded forms in some way, you must restrict the Unicode forms that would lead to them.

3.7. Clear Security Considerations

Definitions of schemes MUST be accompanied by a clear analysis of the security implications for systems that use the scheme; this follows the practice of Security Consideration sections within IANA registrations [RFC5226].

In particular, Section 7 of RFC 3986 [RFC3986] describes general security considerations for URIs, while [RFC3987bis] gives those for IRIs. The definition of an individual URI/IRI scheme should note which of these apply to the specified scheme.

3.8. Scheme Name Considerations

Section 3.1 of RFC 3986 defines the syntax of a URI scheme name; this syntax remains the same for IRIs. New registered schemes registrations MUST follow this syntax, which only allows a limited repertoire of characters (taken from US-ASCII). Although the syntax for the scheme name in URI/IRIs is case insensitive, the scheme names itself MUST be registered using lowercase letters.

URI/IRI scheme names should be short, but also sufficiently descriptive and distinguished to avoid problems.

Avoid names or other symbols that might cause problems with rights to use the name in IETF specifications and Internet protocols. For example, be careful with trademark and service mark names. (See Section 7.4 of [RFC3978].)

Avoid using names that are either very general purpose or associated in the community with some other application or protocol. Avoid scheme names that are overly general or grandiose in scope (e.g., that allude to their "universal" or "standard" nature.)

Organizations that desire a private name space for URI scheme names are encouraged to use a prefix based on their domain name, expressed in reverse order. For example, a URI scheme name of com-example-info might be registered by the vendor that owns the example.com domain name.

4. Guidelines for Provisional URI/IRI Scheme Registration

Provisional registration can be an intermediate step on the way to permanent registration, e.g., before the scheme specification is finalized. Provisional registration is also appropriate for schemes that are known to be used, but where a definitive specification is not available. There is no time limit for provisional registration.

While the guidelines in Section 3 are REQUIRED for permanent registration, they are RECOMMENDED for provisional registration. For a provisional registration, the following are REQUIRED:

- o The scheme name meets the syntactic requirements of Section 3.8 and the encoding requirements of Section 3.6.
- o There is not already an entry with the same scheme name. (In the unfortunate case that there are multiple, different uses of the same scheme name, the IESG may approve a request to modify an existing entry to note the separate use.)

- o Contact information identifying the person supplying the registration is included. Previously unregistered schemes discovered in use may be registered by third parties (even if not on behalf of those who created the scheme). In this case, both the registering party and the scheme creator SHOULD be identified.
- o If no permanent, citable specification for the scheme definition is included, credible reasons for not providing it should be given.
- o The scheme definition SHOULD include a clear Security Considerations (Section 3.7) or explain why a full security analysis is not available (e.g., in a third-party scheme registration).
- o If the scheme definition does not meet the guidelines laid out in Section 3, the differences and reasons SHOULD be noted.

5. Guidelines for Historical URI/IRI Scheme Registration

In some circumstances, it is appropriate to note a URI scheme that was once in use or registered but for whatever reason is no longer in common use or the use is not recommended. In this case, it is possible for an individual to request that the scheme be registered (newly, or as an update to an existing registration) as 'historical'. Any scheme that is no longer in common use MAY be designated as historical; the registration should contain some indication to where the scheme was previously defined or documented.

6. URI/IRI Scheme Registration Procedure

6.1. General

The URI/IRI registration process is described in the terminology of [RFC5226]. The registration process is an optional mailing list review, followed by "Expert Review". The registration request should note the desired status. The Designated Expert will evaluate the request against the criteria of the requested status. In the case of a permanent registration request, the Designated Expert may:

- o Accept the specification of the scheme for permanent registration.
- o Suggest provisional registration instead.
- o Request IETF review and IESG approval; in the meanwhile, suggest provisional registration.

URI/IRI scheme definitions contained within other IETF documents (Informational, Experimental, or Standards-Track RFCs) must also undergo Expert Review; in the case of Standards-Track documents, permanent registration status approval is required.

The registration procedure for URI schemes is intended to be very lightweight for non-contentious registrations. For the most part, we expect the good sense of submitters and reviewers, guided by these procedures, to achieve an acceptable and useful consensus for the community.

In exceptional cases, where the negotiating parties cannot form a consensus, the final arbiter of any contested registration shall be the IESG.

If parties achieve consensus on a registration proposal that does not fully conform to the strict wording of this procedure, this should be drawn to the attention of a relevant member of the IESG.

6.2. Registration Procedures

Someone wishing to register a new URI/IRI scheme MUST:

1. Check the IANA URI scheme registry to see whether or not there is already an entry for the desired name. If there is already an entry under the name, choose a different URI scheme name, or update the existing scheme definition.
2. Prepare a URI/IRI scheme registration template, as specified in Section 6.4. The scheme registration template may be contained in an Internet Draft, submitted alone, or as part of some other permanently available, stable, protocol specification. The template may also be submitted in some other form (as part of another document or as a stand-alone document), but the contents will be treated as an "IETF Contribution" under the guidelines of [RFC3978].
3. Send a copy of the template or a pointer to the containing document (with specific reference to the section with the template) to the mailing list uri-review@ietf.org, requesting review. In addition, request review on other relevant mailing lists as appropriate. For example, general discussion of URI/IRI syntactical issues could be discussed on uri@w3.org; schemes for a network protocol could be discussed on a mailing list for that protocol. Allow a reasonable time for discussion and comments. Four weeks is reasonable for a permanent registration requests.
4. Respond to review comments and make revisions to the proposed registration as needed to bring it into line with the guidelines given in this document.
5. Submit the (possibly updated) registration template (or pointer to document containing it) to IANA at iana@iana.org, specifying whether 'permanent' or 'provisional' registration is requested.

Upon receipt of a URI/IRI scheme registration request, the following steps MUST be followed:

1. IANA checks the submission for completeness; if sections are missing or citations are not correct, IANA may reject the registration request.
2. IANA checks the current registry for a entry with the same name; if such a registry exists, IANA may reject the registration request.
3. IANA requests Expert Review of the registration request against the corresponding guidelines (from this document.)
4. The Designated Expert may request additional review or discussion, as necessary.
5. If Expert Review recommends registration 'provisional' or 'permanent' registration, IANA adds the registration to the appropriate registry.
6. Unless Expert Review has explicitly rejected the registration request within two weeks, IANA should automatically add the registration in the 'provisional' registry.

Either based on an explicit request or independently initiated, the Designated Expert or IESG may request the upgrade of a 'provisional' registration to a 'permanent' one. In such cases, IANA should move the corresponding entry from the provisional registry.

6.3. Change Control

Registrations may be updated in each registry by the same mechanism as required for an initial registration. In cases where the original definition of the scheme is contained in an IESG-approved document, update of the specification also requires IESG approval.

Provisional registrations may be updated by the original registrant or anyone designated by the original registrant. In addition, the IESG may reassign responsibility for a provisional registration scheme, or may request specific changes to a scheme registration. This will enable changes to be made to schemes where the original registrant is out of contact, or unwilling or unable to make changes.

Transition from 'provisional' to 'permanent' status may be requested and approved in the same manner as a new 'permanent' registration. Transition from 'permanent' to 'historical' status requires IESG approval. Transition from 'provisional' to 'historical' may be requested by anyone authorized to update the provisional registration.

6.4. URI/IRI Scheme Registration Template

This template describes the fields that must be supplied in a URI/IRI scheme registration request:

Resource Identifier (RI) Scheme name:

See Section 3.8 for guidelines.

Status:

This reflects the status requested, and should be one of 'permanent', 'provisional', or 'historical'.

Scheme syntax:

See Section 3.2 for guidelines.

Scheme semantics:

See Section 3.3 and Section 3.4 for guidelines.

Encoding considerations:

See Section 3.3 and Section 3.6 for guidelines.

Applications/protocols that use this scheme name:

See Section 3.5.

Interoperability considerations:

If the person or group registering the scheme is aware of any details regarding the scheme that might impact interoperability, identify them here. For example: proprietary or uncommon encoding methods; inability to support multibyte character sets; incompatibility with types or versions of any underlying protocol.

Security considerations:

See Section 3.7 for guidelines.

Contact:

Person (including contact information) to contact for further information.

Author/Change controller:

Person (including contact information) authorized to change this.

References:

Include full citations for all referenced documents. Registration templates for provisional registration may be included in an Internet Draft; when the documents expire or are approved for publication as an RFC, the registration will be updated.

7. The "example" Scheme

There is a need for a URI/IRI Scheme name that can be used for examples in documentation without fear of conflicts with current or future actual schemes. The URI/IRI Scheme "example" is hereby registered as a Permanent URI/IRI Scheme for that purpose.

Scheme name: example

Status: permanent

Scheme syntax: The entire range of allowable syntax for URI/IRI schemes specified in [RFC3987bis] is allowed for "example" URI/IRIs.

Scheme semantics: URI/IRIs in the "example" scheme should be used for documentation purposes only. The use of "example" URIs/IRIs must not be used as locators, identify any resources, or specify any particular set of operations.

Encoding considerations: See Section 2.5 of [RFC3986] for guidelines.

Applications/protocols that use this URI scheme name: The "example" URI should be used for documentation purposes only. It MUST not be used for any protocol.

Interoperability considerations: None.

Security considerations: None.

Contact: N/A

Author/Change controller: IETF

References: This RFC XXXX.

RFC Editor Note: Replace XXXX with this RFC's reference.

8. IANA Considerations

Previously, the former "URL Scheme" registry was replaced by the Uniform Resource Identifier scheme registry. The process was based on [RFC5226] "Expert Review" with an initial (optional) mailing list review.

The updated template has an additional field for the status of the scheme, and the procedures for entering new name schemes have been augmented. Section 6 establishes the process for new URI/IRI scheme registration.

IANA is requested to update the name of the registry "URI Schemes" to "URI/IRI Schemes". The registry should be updated to point to this document. For the tables within that registry "Permanent URI Schemes" should become "Permanent URI/IRI Schemes", "Provisional URI Schemes" should become "Provisional URI/IRI Schemes", and "Historical URI Schemes" should become "Historical URI/IRI Schemes".

The example URI scheme "example" is hereby registered. (See the template above for registration.)

9. Security Considerations

All registered values are expected to contain accurate security consideration sections; 'permanent' registered scheme names are expected to contain complete definitions.

Information concerning possible security vulnerabilities of a protocol may change over time. Consequently, claims as to the

security properties of a registered URI/IRI scheme may change as well. As new vulnerabilities are discovered, information about such vulnerabilities may need to be attached to existing documentation, so that users are not misled as to the true security properties of a registered URI scheme.

10. Acknowledgements

Many thanks to Patrick Faltstrom for his comments on this version.

Many thanks to Paul Hoffmann, Ira McDonald, Roy Fielding, Stu Weibel, Tony Hammond, Charles Lindsey, Mark Baker, and other members of the uri@w3.org mailing list for their comments on earlier versions.

Parts of this document are based on [RFC2717], [RFC2718] and [RFC3864]. Some of the ideas about use of URIs were taken from the "Architecture of the World Wide Web" [W3CWebArch].

Appendix A. Changes Since RFC 4395

1. Significant edits to be clear that a "URI scheme" and an "IRI scheme" are the same thing.
2. Added the "example:" URL Scheme.
3. Allow for IRI-specific scheme registration.
4. Clarify that the URI scheme registry is also the IRI scheme registry.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC3978] Bradner, S., "IETF Rights in Contributions", RFC 3978, March 2005.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66,

RFC 3986, January 2005.

[RFC3987bis]

Duerst, M., Masinter, L., and M. Suignard,
"Internationalized Resource Identifiers (IRIs)",
September 2010,
<<http://tools.ietf.org/id/draft-ietf-iri-3987bis>>.

11.2. Informative References

- [RFC2717] Petke, R. and I. King, "Registration Procedures for URL Scheme Names", BCP 35, RFC 2717, November 1999.
- [RFC2718] Masinter, L., Alvestrand, H., Zigmond, D., and R. Petke, "Guidelines for new URL Schemes", RFC 2718, November 1999.
- [RFC3406] Daigle, L., van Gulik, D., Iannella, R., and P. Faltstrom, "Uniform Resource Names (URN) Namespace Definition Mechanisms", BCP 66, RFC 3406, October 2002.
- [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.
- [RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.
- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.
- [RFC4690] Klensin, J., Faltstrom, P., Karp, C., and IAB, "Review and Recommendations for Internationalized Domain Names (IDNs)", RFC 4690, September 2006.
- [W3CWebArch]
W3C Technical Architecture Group, "Architecture of the World Wide Web, Volume One", December 2004,
<<http://www.w3.org/TR/webarch/>>.

Authors' Addresses

Tony Hansen
AT&T Laboratories
200 Laurel Ave.
Middletown, NJ 07748
USA

Email: tony+urireg@maillennium.att.com

Ted Hardie
Panasonic Wireless Research Lab
10900 Tantau Ave.
Cupertino, CA
USA

Phone: +1 408 628 5864
Email: ted.ietf@gmail.com

Larry Masinter
Adobe
345 Park Ave.
San Jose, CA 95110
US

Phone: +1 408 536 3024
Email: masinter@adobe.com
URI: <http://larry.masinter.net>

Internationalized Resource Identifiers
(iri)
Internet-Draft
Intended status: BCP
Expires: April 24, 2013

M. Duerst
Aoyama Gakuin University
L. Masinter
Adobe
A. Allawi
Diwan Software Limited
October 21, 2012

Guidelines for Internationalized Resource Identifiers with Bi-
directional Characters (Bidi IRIs)
draft-ietf-iri-bidi-guidelines-03

Abstract

This specification gives guidelines for selection, use, and presentation of International Resource Identifiers (IRIs) which include characters with inherent right-to-left (rtl) writing direction.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on April 24, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Overview	3
1.2. Availability	3
1.3. Notation	4
2. Logical Storage and Visual Presentation	4
3. Bidi IRI Structure	5
4. Input of Bidi IRIs	6
5. Examples	6
6. IANA Considerations	8
7. Security Considerations	9
8. Acknowledgements	9
9. Main Changes Since RFC 3987	9
10. References	9
10.1. Normative References	9
10.2. Informative References	10
Appendix A. List of ASCII Symbols and their Bidirectional Character Types	10
Authors' Addresses	11

1. Introduction

1.1. Overview

Some UCS characters, such as those used in the Arabic and Hebrew scripts, have an inherent right-to-left (rtl) writing direction as opposed to characters, such as those in the Latin script, that have an inherent left-to-right (ltr) direction. IRIs containing rtl characters (called bidirectional IRIs or Bidi IRIs) require additional attention because of the non-trivial relation between their logical and visual ordering. The logical order represents the order in which characters are stored on computers and read by people. The visual order is the order in which the characters appear (or are expected to appear) on a computer display or printout.

Generally, alphabetic characters in scripts like Arabic and Hebrew are drawn rtl while numbers are drawn ltr. Symbols such as slash ('/') and period ('.') take their visual direction from the surrounding characters. A list of all ASCII symbols with their bidirectional character type and their function in URIs and IRIs is given in Appendix A.

Because of this 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; and
3. minor or no changes or restrictions for implementations.

1.2. Availability

This document is available in (line-printer ready) plaintext ASCII and in PDF. It is also available in HTML from <http://www.sw.it.aoyama.ac.jp/2012/pub/draft-ietf-iri-bidi-guidelines-03.html>, and in UTF-8 plaintext from <http://www.sw.it.aoyama.ac.jp/2012/pub/draft-ietf-iri-bidi-guidelines-03.utf8.txt>. While all these versions are identical in their technical content, the HTML, PDF, and UTF-8 plaintext versions show non-Unicode characters directly. This often makes it easier to understand examples, and readers are therefore strongly advised to consult one of these versions in preference to or as a supplement to the ASCII version.

1.3. Notation

In this document, "Bidi Notation", abbreviated "BN" is used for the given Bidi IRI examples as follows: Lower case letters a-z stand for characters that are written with a left to right ordering (such as Latin characters), whereas upper case letters A-Z represent characters that are written right to left (such as Arabic or Hebrew characters). Numbers and symbols are the same.

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

2. Logical Storage and Visual Presentation

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

Bidi IRIs MUST be visually ordered by the Unicode Bidirectional Algorithm [UNIV6], [UNI9]. Bidi IRIs MUST be rendered in the same way as they would be if they were in a left-to-right embedding.

In conformance with the Unicode Bidirectional Algorithm, embedding MAY be done in one of two ways:

1. precede the IRI with U+202A, LEFT-TO-RIGHT EMBEDDING (LRE), and follow with U+202C, POP DIRECTIONAL FORMATTING (PDF); or
2. use a higher-level protocol (e.g., the `dir='ltr'` attribute in HTML).

Preceding and following the Bidi IRI with U+200E, LEFT-TO-RIGHT MARK (LRM) is NOT RECOMMENDED as, there are cases where this may not be sufficient to match full left to right embedding.

There is no requirement to use embedding if the display is still the same without the embedding. For example, a Bidi 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 (such as symbols) and that starts and ends with a strong right-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.

However, implementers are RECOMMENDED 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 ensure correct display are not themselves 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 appear themselves. It would therefore not be possible to input an IRI with such characters correctly.

3. Bidi IRI Structure

The Unicode Bidirectional Algorithm is designed for general purpose text. To make sure that it does not affect the rendering of Bidi IRIs outside of the requirements of this document, some restrictions on Bidi IRIs are necessary. These restrictions are given in terms of delimiters (structural characters, mostly punctuation such as "@", ".", ":", and "/") and components (usually consisting mostly of letters and digits).

The following syntax rules from the ABNF of [RFC3987bis] 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 schemes with `ireg-name` as 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 with a right-to-left character, and end with a right-to-left character potentially followed by one or more nonspacing mark (bidi class NSM).

The above restrictions are given as "SHOULD"s, rather than as "MUST"s. For IRIs that are never presented visually, they are not relevant. However, for IRIs in general, they are very important to ensure 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 using the general percent-encoding of IRI components, as described in [RFC3987bis]. Please note that the whole component has to be mapped (see also Example 9 below).

4. Input of Bidi IRIs

Bidi input methods MUST generate Bidi IRIs in logical order while rendering them according to Section 2. During input, rendering SHOULD be updated after every new character is input to avoid end-user confusion.

5. Examples

This section gives examples of Bidi IRIs in Bidi Notation. It shows legal IRIs with the relationship between their 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 3 are not followed. Please see <Availability> for versions of the examples

in Arabic and Hebrew script.

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.

Please note that "BN" stands for "Bidi Notation", see <Notation>. AR stands for Arabic, HE for Hebrew.

Example 1: A single component with rtl characters is inverted:
Logical representation (BN): "http://ab.CDEFGH.ij/kl/mn/op.html"
Visual representation (BN): "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 (BN): "http://ab.CDE.FGH/ij/kl/mn/op.html"

Visual representation (BN): "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 (BN):

"http://AB.CD.EF/GH/IJ/KL?MN=OP;QR=ST#UV"

Visual representation (BN): "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: Each of several sequences of rtl components is inverted on its own:

Logical representation (BN): "http://AB.CD.ef/gh/IJ/KL.html"

Visual representation (BN): "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 (BN): "http://ab.cd.EF/GH/ij/kl.html"

Visual representation (BN): "http://ab.cd.HG/FE/ij/kl.html"

The inversion of the domain name label and the path component may be unexpected, but it 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 Unicode Bidirectional 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 (BN): "http://ab.CD.EF/GH/IJ/kl.html"

Visual representation (BN): "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 includes digits:

Logical representation (BN): "http://ab.CDE123FGH.ij/kl/mn/op.html"

Visual representation (BN): "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 are at the start or end of an rtl component:

Logical representation (BN): "http://ab.cd.ef/GH1/2IJ/KL.html"

Visual representation (BN): "http://ab.cd.ef/LK/JI1/2HG.html"

The sequence "1/2" is interpreted by the Bidirectional 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 (BN): "http://ab.cd.ef/GH%31/%32IJ/KL.html"

Visual representation (BN): "http://ab.cd.ef/LK/JI%32/%31HG.html"

Example 10 (allowed but not recommended):

Logical representation (BN): "http://ab.CDEFGH.123/kl/mn/op.html"

Visual representation (BN): "http://ab.123.HGFEDC/kl/mn/op.html"

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

Example 11 (allowed but not recommended):

Logical representation (BN): "http://ab.CDEFGH.123ij/kl/mn/op.html"

Visual representation (BN): "http://ab.123.HGFEDCij/kl/mn/op.html"

Components consisting of numbers and left-to-right characters are allowed, but these may interact with adjacent RTL components in ways that are not easy to predict.

6. IANA Considerations

This document makes no changes to IANA registries.

7. Security Considerations

Confusion can occur with bidirectional IRIs, if the restrictions in Section 3 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 be used.

8. Acknowledgements

This document was derived from [RFC3987] and [RFC3987bis] and the acknowledgments of those documents apply. Shunsuke Oshima provided the data for Appendix A.

9. Main Changes Since RFC 3987

This section describes the main changes since [RFC3987].

- o Separated out the section on bidi in [RFC3987] to this document.
- o Added examples in Arabic and Hebrew, which can be seen in html/pdf/utf8.txt versions.
- o Allowed NSMs at the end of components, for Dhivehi, Yiddish,...
- o TODO: check for major changes between RFC3987 and draft -02.

Note to RFC Editor: Please remove this paragraph before publication. Detailed change logs are available in the IETF tools subversion repository at <http://trac.tools.ietf.org/wg/iri/trac/log/draft-ietf-iri-3987bis/draft-ietf-iri-bidi-guidelines.xml>.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.
- [RFC3987bis] Duerst, M., Masinter, L., and M. Suignard,

"Internationalized Resource Identifiers (IRIs)",
October 2012,
<<http://tools.ietf.org/id/draft-ietf-iri-3987bis>>.

[UNI9] Davis, M., "The Unicode Bidirectional Algorithm", Unicode
Standard Annex #9, September 2012,
<<http://www.unicode.org/reports/tr9/tr9-27.html>>.

[UNIV6] The Unicode Consortium, "The Unicode Standard, Version
6.2.0 (Mountain View, CA, The Unicode Consortium, 2012,
ISBN 978-1-936213-07-8)", October 2012.

10.2. Informative References

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

Appendix A. List of ASCII Symbols and their Bidirectional Character Types

To help understand the influence of various symbols on IRI display,
this appendix lists all of them, giving the character itself, the
Unicode codepoint, the character name, the bidirectional character
type (BCT) and the rule and relevance in the IRI syntax.

The most important ones in practice are ":", delimiting schem and
port (CS, Common Number Separator), "/" to indicate generic
(hierarchical) schemes and as a path separator (CS, Common Number
Separator), "?" to introduce a query part (ON, Other Neutral), "#" to
introduce a fragment identifier (ET, European Number Terminator), "."
to separate labels in a domain name (CS, Common Number Separator),
"&" to separate form parameters (ON, Other Neutral), and "@" to
separate user information (ON, Other Neutral).

Char	Codepoint	Character Name	BCT	IRI syntax
"#"	U+0023	NUMBER SIGN	ET	gen-delims, fragments
"/"	U+002F	SOLIDUS	CS	gen-delims, paths
":"	U+003A	COLON	CS	gen-delims, scheme, port
"?"	U+003F	QUESTION MARK	ON	gen-delims, query part
"@"	U+0040	COMMERCIAL AT	ON	gen-delims, user
"["	U+005B	LEFT SQUARE BRACKET	ON	gen-delims
"]"	U+005D	RIGHT SQUARE BRACKET	ON	gen-delims
"%"	U+0025	PERCENT SIGN	ET	pcd-encoded
"!"	U+0021	EXCLAMATION MARK	ON	sub-delims
","	U+002C	COMMA	CS	sub-delims
"+"	U+002B	PLUS SIGN	ES	sub-delims
"\$"	U+0024	DOLLAR SIGN	ET	sub-delims
"("	U+0028	LEFT PARENTHESIS	ON	sub-delims
"'"	U+0027	APOSTROPHE	ON	sub-delims
")"	U+0029	RIGHT PARENTHESIS	ON	sub-delims
"*"	U+002A	ASTERISK	ON	sub-delims
";"	U+003B	SEMICOLON	ON	sub-delims
"="	U+003D	EQUALS SIGN	ON	sub-delims, forms
"&"	U+0026	AMPERSAND	ON	sub-delims, forms
"."	U+002E	FULL STOP	CS	unreserved, domain names
"-"	U+002D	HYPHEN-MINUS	ES	unreserved
"_"	U+005F	LOW LINE	ON	unreserved
"~"	U+007E	TILDE	ON	unreserved
" "	U+0020	SPACE	WS	excluded, delim
' "'	U+0022	QUOTATION MARK	ON	excluded, delim
"\"	U+005C	REVERSE SOLIDUS	ON	excluded, unwise
"^"	U+005E	CIRCUMFLEX ACCENT	ON	excluded, unwise
"<"	U+003C	LESS-THAN SIGN	ON	excluded, delim
">"	U+003E	GREATER-THAN SIGN	ON	excluded, delim
"`"	U+0060	GRAVE ACCENT	ON	excluded, unwise
" "	U+007C	VERTICAL LINE	ON	excluded, unwise
"{"	U+007B	LEFT CURLY BRACKET	ON	excluded, delim
"}"	U+007D	RIGHT CURLY BRACKET	ON	excluded, delim

Authors' Addresses

Martin J. Duerst (Note: Please write "Duerst" with u-umlaut wherever possible, for example as "Dürst" in XML and HTML.)
Aoyama Gakuin University
5-10-1 Fuchinobe
Chuo-ku
Sagamihara, Kanagawa 252-5258
Japan

Phone: +81 42 759 6329
Fax: +81 42 759 6495
Email: duerst@it.aoyama.ac.jp
URI: <http://www.sw.it.aoyama.ac.jp/D%C3%BCrst/>
(Note: This is the percent-encoded form of an IRI)

Larry Masinter
Adobe
345 Park Ave
San Jose, CA 95110
U.S.A.

Phone: +1-408-536-3024
Email: masinter@adobe.com
URI: <http://larry.masinter.net>

Adil Allawi
Diwan Software Limited
37-39 Peckham Road
London SE5 8UH
United Kingdom

Phone: +44 7718 785850
Fax: +44 20 72525444
Email: adil@diwan.com
URI: <http://ironymark.diwan.com/>

Internationalized Resource Identifiers
(iri)
Internet-Draft
Updates: 3986 (if approved)
Intended status: Standards Track
Expires: April 26, 2013

L. Masinter
Adobe
M. Duerst
Aoyama Gakuin University
October 23, 2012

Comparison, Equivalence and Canonicalization of Internationalized
Resource Identifiers
draft-ietf-iri-comparison-02

Abstract

Internationalized Resource Identifiers (IRIs) are Unicode strings used to identify resources on the Internet. Applications that use IRIs often define a means of comparing IRIs to determine when two IRIs are equivalent for the purpose of that application. Some applications also define a method for canonicalizing an IRI -- translating one IRI into another which is equivalent under the comparison method used.

This document gives guidelines and best practices for defining and using IRI comparison and canonicalization methods.

Comparison methods are used to determine equivalence. As URIs are a subset of IRIs, the guidelines apply to URI comparison as well.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on April 26, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. General guidelines	4
3. Preparation for Comparison	5
4. Comparison Hierarchy	6
4.1. Simple String Comparison	6
4.2. Syntax-Based Equivalence	7
4.2.1. Case Equivalence	8
4.2.2. Unicode Character Normalization	8
4.2.3. Percent-Encoding Equivalence	9
4.2.4. Path Segment Equivalence	10
4.3. Scheme-Based Comparison	10
4.4. Protocol-Based Comparison	11
5. Security Considerations	12
6. Acknowledgements	12
7. References	12
7.1. Normative References	12
7.2. Informative References	13
Authors' Addresses	14

1. Introduction

Internationalized Resource Identifiers (IRIs) are Unicode strings used to identify resources on the Internet. Applications that use IRIs often define a means of comparing IRIs to determine when two IRIs are equivalent for the purpose of that application. Some applications also define a method for canonicalizing an IRI -- translating one IRI into another which is equivalent under the comparison method used.

This document gives guidelines and best practices for defining and using IRI comparison and canonicalization methods.

As every URI is also an IRI, the comparison and canonicalization methods also apply to URIs.

IRI comparison is expected to determine whether two IRIs are equivalent without using the IRIs to access their respective resource(s). For example, comparisons are performed whenever a response cache is accessed, a browser checks its history to color a link, or an XML parser processes tags within a namespace.

Comparison for equivalence is often accomplished by canonicalization: (sometimes called normalization): a process for converting data that has more than one possible representation into a "standard", "normal", or "canonical" form. Extensive canonicalization prior to comparison of IRIs may be used by spiders and indexing engines to prune a search space or reduce duplication of request actions and response storage.

IRI comparison is performed for some particular purpose. Protocols or implementations that compare IRIs for different purposes will often be subject to differing design trade-offs in regards to how much effort should be spent in reducing aliased identifiers. This document describes various methods that may be used to compare IRIs, the trade-offs between them, and the types of applications that might use them.

2. General guidelines

Because IRIs exist to identify resources, one might expect two IRIs to be considered equivalent when they identify the same resource. However, this definition of equivalence is not of much practical use, as there is in general no way for an implementation to compare two resources to determine if they are "the same" unless it has full knowledge or control of them. Comparison methods for IRIs are generally based strictly on examining the characters that make up the

IRI, without performing any network access.

We use the terms "different" and "equivalent" to describe the possible outcomes of such comparisons, but there are many application-dependent versions of equivalence.

Even when it is possible to determine that two IRIs are equivalent, IRI comparison is not sufficient to determine whether two IRIs identify different resources. For example, an owner of two different domain names could decide to serve the same resource from both, resulting in two different IRIs. For this reason, false negatives (e.g., returning "different" even with the resources are "the same") cannot be completely avoided. Comparison methods often try to minimize false negatives while strictly avoiding false positives. However, in some cases (such as cache invalidation), false negatives are more harmful than false positives.

A comparison method for determining equivalence might have multiple values, for example, returning "equivalent", "different", or "equivalence cannot be determined".

Multiple canonicalization (normalizations) methods might be defined, where sequential application of each results in greater sets of equivalent values.

In testing for equivalence, applications should not directly compare relative references; the references should be converted to their respective target IRIs before comparison. [[ref 3987bis]]

Some IRIs contain fragment identifiers. In general, the equivalence of two IRIs is determined first by comparing the IRIs without any fragment identifiers, and then (if appropriate) the fragment components (if any) compared.

Some applications (such as XML namespaces) use IRIs as identity tokens without any relationship to accessing the resources. Those applications use the Simple String Comparison (see Section 4.1).

3. Preparation for Comparison

Any kind of IRI comparison REQUIRES that any additional contextual processing is first performed, including undoing higher-level escapings or encodings in the protocol or format that carries an IRI. This preprocessing is usually done when the protocol or format is parsed.

NOTE: This document has not yet been updated to use in-line Unicode

examples.

Examples of such escapings or encodings are entities and numeric character references in [HTML4] and [XML1]. As an example, "http://example.org/rosé" (in HTML), "http://example.org/rosé" (in HTML or XML), and "http://example.org/rosé" (in HTML or XML) are all resolved into what is denoted in this document (see 'Notation' section of [RFC3987bis]) as "http://example.org/rosé" (the "é" here standing for the actual e-acute character, to compensate for the fact that this document cannot contain non-ASCII characters).

An IRI is a sequence of Unicode characters. IRIs are sometimes represented in documents as sequences of bytes in a charset, either Unicode-based (UTF-8) or using some other character encoding (e.g., ISO-8859-1). Before comparing two such sequences, they must both be converted into sequences of Unicode characters.

Similarly, encodings such as Transfer Codings in HTTP (see [RFC2616]) and Content Transfer Encodings in MIME ([RFC2045]) must be unencoded. In these cases, the encoding is based not on characters but on octets, and additional care is required to make sure that characters, and not just arbitrary octets, are compared (see Section 4.1).

4. Comparison Hierarchy

In practice, a variety of methods are used to test IRI equivalence. These methods generally 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.

The following discussion starts with comparison methods that are cheap but have a relatively higher chance of producing false negatives, and proceeding to those that have higher computational cost and lower risk of false negatives.

4.1. Simple String Comparison

If two IRIs (when considered as strings of Unicode characters) are identical, then it is safe to conclude that they are equivalent. This type of equivalence test has very low computational cost and is in wide use in a variety of applications, particularly in the domain of parsing. It is also used when a definitive answer to the question of IRI equivalence is needed that is independent of the scheme used

and that can be calculated quickly and without accessing a network. An example of such a case is XML Namespaces ([XMLNamespace]).

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

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

False negatives are caused by the production and use of IRI aliases. Unnecessary aliases can be reduced, regardless of the comparison method, by consistently providing IRI references in a canonical form (after canonicalization is applied).

Protocols and data formats might limit some IRI comparisons to simple string comparison, based on the theory that people and implementations will, in their own best interest, be consistent in providing IRI references, or at least be consistent enough to negate any efficiency that might be obtained from further canonicalization.

4.2. Syntax-Based Equivalence

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

```
example://a/b/c/%7Bfoo%7D/ros&#xE9;  
example://a/./b/./b/%63/%7Bfoo%7D/ros%C3%A9
```

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

4.2.1. Case Equivalence

For all IRIs, the hexadecimal digits within a percent-encoding triplet (e.g., "%3a" versus "%3A") are case-insensitive and therefore should be considered equivalent to forms which 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 treated equivalent to lowercase. For example, the URI "HTTP://www.EXAMPLE.com/" is equivalent to "http://www.example.com/". Case equivalence for non-ASCII characters in IRI components that are IDNs are discussed in Section 4.3. The other generic syntax components are assumed to be case sensitive unless specifically defined otherwise by the scheme.

Creating schemes that allow case-insensitive syntax components containing non-ASCII characters should be avoided. Case equivalence of non-ASCII characters can be culturally dependent and is always a complex operation. The only exception concerns non-ASCII host names for which the character normalization includes a mapping step derived from case folding.

4.2.2. Unicode Character Normalization

The Unicode Standard [UNIV6] defines various equivalences between sequences of characters for various purposes. Unicode Standard Annex #15 [UTR15] defines various Normalization Forms for these equivalences, in particular Normalization Form C (NFC, Canonical Decomposition, followed by Canonical Composition) and Normalization Form KC (NFKC, Compatibility Decomposition, followed by Canonical Composition).

IRIs already in Unicode MUST NOT be normalized before parsing or interpreting. In many non-Unicode character encodings, some text cannot be represented directly. For example, the word "Vietnam" is natively written "Việt Nam" (containing a LATIN SMALL LETTER E WITH CIRCUMFLEX AND DOT BELOW) in NFC, but a direct transcoding from the windows-1258 character encoding leads to "Việt Nam" (containing a LATIN SMALL LETTER E WITH CIRCUMFLEX followed by a COMBINING DOT BELOW). Direct transcoding of other 8-bit encodings of

Vietnamese may lead to other representations.

Equivalence of IRIs MUST rely on the assumption that IRIs are appropriately pre-character-normalized rather than apply character normalization when comparing two IRIs. The exceptions are conversion from a non-digital form, and conversion from a non-UCS-based character encoding to a UCS-based character encoding. In these cases, NFC or a normalizing transcoder using NFC MUST be used for interoperability. To avoid false negatives and problems with transcoding, IRIs SHOULD be created by using NFC. Using NFKC may avoid even more problems; for example, by choosing half-width Latin letters instead of full-width ones, and full-width instead of half-width Katakana.

As an example, "http://www.example.org/résumé.html" (in XML Notation) is in NFC. On the other hand, "http://www.example.org/résuné.html" is not in NFC.

The former uses precombined e-acute characters, and the latter uses "e" characters followed by combining acute accents. Both usages are defined as canonically equivalent in [UNIV6].

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

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

4.2.3. Percent-Encoding Equivalence

The percent-encoding mechanism (Section 2.1 of [RFC3986]) is a frequent source of variance among otherwise identical IRIs. In addition to the case equivalence issue noted above, some IRI producers percent-encode octets that do not require percent-encoding,

resulting in IRIs that are equivalent to their nonencoded counterparts. These IRIs should be compared by first decoding any percent-encoded octet sequence that corresponds to an unreserved character, as described in section 2.3 of [RFC3986].

For actual resolution, differences in percent-encoding (except for the percent-encoding of reserved characters) SHOULD always result in the same resource. For example, "http://example.org/~user", "http://example.org/%7Euser", and "http://example.org/%7Euser", SHOULD resolve to the same resource.

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

4.2.4. Path Segment Equivalence

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

4.3. Scheme-Based Comparison

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

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

In general, an IRI that uses the generic syntax for authority with an empty path should be equivalent to a path of `"/`. Likewise, an explicit `":port"`, for which the port is empty or the default for the scheme, is equivalent to one where the port and its `":"` delimiter are elided.

Another case where equivalence 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.

The presence of a missing component vs. one with an empty string component in an IRI SHOULD NOT be treated as equivalent unless explicitly defined as such by the scheme definition. For example, the IRI `"http://example.com/?"` cannot be assumed to be equivalent to any of the examples above; an empty query component is NOT equivalent to a missing one. 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 equivalence; thus, two IRIs that differ only by the suffix `"#"` are considered different regardless of the scheme.

Some IRI schemes allow the usage of Internationalized Domain Names (IDN) [RFC5890] either in their ireg-name part or elsewhere. When in use in IRIs, those names SHOULD conform to the definition of U-Label in [RFC5890]. An IRI containing an invalid IDN cannot successfully be resolved. For legibility purposes, they SHOULD NOT be converted into ASCII Compatible Encoding (ACE).

Scheme-based comparison 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 equivalence rules are possible.

4.4. Protocol-Based Comparison

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

`http://example.com/data`

redirects to an IRI differing only in the trailing slash
`http://example.com/data/`

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

5. Security Considerations

The primary security difficulty comes from applications choosing the wrong equivalence relationship, or two different parties disagreeing on equivalence. This is especially a problem when IRIs are used in security protocols.

Besides the large character repertoire of Unicode, reasons for confusion include different forms of normalization and different normalization expectations, use of percent-encoding with various legacy encodings, and bidirectionality issues. See also [UTR36].

6. Acknowledgements

This document was originally derived from [RFC3986] and [RFC3987], based on text contributed by Tim Bray.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.
- [RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

- [RFC3987bis] Duerst, M., Masinter, L., and M. Suignard, "Internationalized Resource Identifiers (IRIs)", 2012, <<http://tools.ietf.org/id/draft-ietf-iri-3987bis>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.
- [UNIV6] The Unicode Consortium, "The Unicode Standard, Version 6.0.0 (Mountain View, CA, The Unicode Consortium, 2011, ISBN 978-1-936213-01-6)", October 2010.
- [UTR15] Davis, M. and M. Duerst, "Unicode Normalization Forms", Unicode Standard Annex #15, March 2008, <<http://www.unicode.org/unicode/reports/tr15/tr15-23.html>>.

7.2. Informative References

- [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/html401/appendix/notes.html#h-B.2>>.
- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.
- [UTR36] Davis, M. and M. Suignard, "Unicode Security Considerations", Unicode Technical Report #36, August 2010, <<http://unicode.org/reports/tr36/>>.
- [XML1] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Forth Edition)", World Wide Web Consortium Recommendation, August 2006, <<http://www.w3.org/TR/REC-xml>>.
- [XMLNamespace] Bray, T., Hollander, D., Layman, A., and R. Tobin, "Namespaces in XML (Second Edition)", World Wide Web

Consortium Recommendation, August 2006,
<<http://www.w3.org/TR/REC-xml-names>>.

Authors' Addresses

Larry Masinter
Adobe
345 Park Ave
San Jose, CA 95110
U.S.A.

Phone: +1-408-536-3024
Email: masinter@adobe.com
URI: <http://larry.masinter.net>

Martin Duerst
Aoyama Gakuin University
5-10-1 Fuchinobe
Sagamihara, Kanagawa 229-8558
Japan

Phone: +81 42 759 6329
Fax: +81 42 759 6495
Email: duerst@it.aoyama.ac.jp
URI: <http://www.sw.it.aoyama.ac.jp/D%C3%BCrst/>

