PRI 185: Extension of UBA for improved display of URL/IRIs

Contents

Proposal

This document discusses proposals for an extension to the Unicode Bidirectional Algorithm (UBA) to better handle URLs and similar common structured text. The Unicode Technical Committee would appreciate feedback on the advantages and disadvantages of introducing such an extension, and the best choices among the options for doing so.

Problem

The Unicode Bidirectional Algorithm (UBA) was designed for handling ordinary text, and predated the rise of the web. Unfortunately, IRI/URLs* are not ordinary text; they are syntactically complex in ways that don't work well with the UBA. That causes IRIs that contain right-to-left text (such as Arabic or Hebrew) to appear jumbled, to the point where the IRIs are either uninterpretable, misleading, or ambiguous. In particular the ambiguous displays could cause security problems.

> *Formally speaking, what we are talking about are IRIs, although most end-users know them as URLs. For more information, see idn-and-iri.

For example, consider the IRIs in Table 1. These sample IRIs are shown in "memory" order first; uppercase bold represents right-to-left characters.

Table 1. Sample IRIs, Memory order

IRIs	**Comments on fields**
http://ab.cd.com/mn/op	All LTR characters

http://ab.cd.**EF.GH.**com/**IJ/KL/**mn/op	Mixture of RTL and LTR fields
http://**EF.GH/IJ/KL**	All RTL characters (except the scheme "http")

They would be displayed as in Table 2 by any Unicode-compliant implementation.

Table 2. Sample IRIs, Display order

Environment	**Display**	**Details**
LTR	http://ab.cd.com/mn/op http://ab.cd.**HG.FE.**com/**LK/JI/**mn/op http://**LK/JI/HG.FE**	http://goo.gl/yvNFq
RTL	http://ab.cd.com/mn/op mn/op/**LK/JI/**com.**HG.FE**.http://ab.cd **LK/JI/HG.FE**//:http	http://goo.gl/DuELB

People have been looking for an extension to the Unicode Bidirectional Algorithm (UBA) that handles IRIs in a more consistent way for bidi users. The general goal of such an extension would be for the "fields" of an IRI to flow in a consistent direction.

While the UBA permits higher-level protocols to modify the display of text, serious interoperability problems result if different applications use different conventions for IRIs. A number of major companies either are in the process of extending, or intend to extend the UBA for display of IRIs, in response to pressure from users. To avoid interoperability problems, a consistent set of conventions needs to be adopted across a wide range of applications for any such extension for handling IRIs and other structured text. For example, when someone copies the contents of an address bar into an email, we don't want all the fields in the IRI to switch around. Attaining such consistency would require a general extension to the UBA to indicate how IRIs should be displayed, both in the limited context of an address bar, and in plain text.

While a major focus for such extensions is the display of IRIs, it is important that any solution also apply to email addresses and filenames.

There are challenges for developing such an extension.

1. There will be a long migration period as implementations are updated from the current version of UBA to a revised version. During that period, people will be using a mixture of old and new applications. It is crucial to minimize the negative effects for interoperability of different displays of IRIs during this transition period.
2. It is necessary to have a simple, comprehensible way to recognize IRIs in plain text to

support correct display in text, and to avoid jumbled, ambiguous IRIs

The Unicode Technical Committee approaches changes to the UBA very carefully. Any changes must be carefully vetted, because the exact behavior of the UBA is crucial to the appearance of millions of already-published documents and web pages. Thus an extended period of comment and testing is required. Any specification for an extension may be initially characterized as *experimental*.

Recognizing IRIs in plain text

There are two problems:

1. A formal reading of the IRI spec allows almost anything in fields, so it is hard to test for termination.
2. The scheme for an IRI (such as "http://") is commonly omitted when IRIs are represented in text. For example, the string "adobe.com" should be recognized as being an IRI when it occurs in the body of an email message.

The first issue can be addressed with the following approach. While in theory, almost any Unicode character can occur in fields in an IRI, in practice many characters have very restricted usage in IRIs. One can take advantage of this pattern by defining a restricted and simplified syntax for IRIs, which captures the majority of actual practice. This syntax can then be used to define display of IRIs for UBA. IRIs that need to use characters outside of this restricted syntax can still be appropriately displayed by the UBA by representing those characters with % escapes.

For the second issue, a common technique can be applied. The technique is to recognized a list of TLDs in context. For example, microsoft.com and google.de can both be recognized. (See http://www.iana.org/domains/root/db/).

Here is a proposed BNF for recognizing IRIs and other structured text within plain text. This BNF uses a Perl-style syntax:

[Review Note: this BNF is simplified so as to not interfere with understanding the general proposal. It would be completed in a final proposal.]

```
bidi_structure := bidi_IRI | bidi_filename | bidi_email

bidi_IRI := ((scheme "://" domain) | domainWithTLD)
                    ("/" path)?
                    ("?" query)?
                    ("#" fragment)?
```

```
domain :=                     label (IDNSep label)* IDNSep?
domainWithTLD :=              label (IDNSep label)* IDNSep TLD IDNSep?
label :=                      UTS46Chars +
IDNSep := [\u002E \uFF0E \u3002\uFF61] // see http://unicode.org/reports/tr46/#Notation

TLD := <list on http://www.iana.org/domains/root/db/>

path := (char - "?" - "#")*
query := (char - "#")*
fragment := char*

char := percentEncodedUTF8
     | [[:L:][:N:][:M:][:S:][:Pd:][:Pc:][:Cf:] inclusionChar - exclusionChar]

inclusionChar :=
        U+0021 ( ! ) EXCLAMATION MARK
        U+0022 ( " ) QUOTATION MARK
        U+0023 ( # ) NUMBER SIGN
        U+0025 ( % ) PERCENT SIGN
        U+0026 ( & ) AMPERSAND
        U+0027 ( ' ) APOSTROPHE
        U+002A ( * ) ASTERISK
        U+002C ( , ) COMMA
        U+002E ( . ) FULL STOP
        U+002F ( / ) SOLIDUS
        U+003A ( : ) COLON
        U+003B ( ; ) SEMICOLON
        U+003F ( ? ) QUESTION MARK
        U+0040 ( @ ) COMMERCIAL AT
        U+005C ( \ ) REVERSE SOLIDUS
        U+00A1 ( ¡ ) INVERTED EXCLAMATION MARK
        U+00B7 ( · ) MIDDLE DOT
        U+00BF ( ¿ ) INVERTED QUESTION MARK

exclusionChar :=
        U+003C ( < ) LESS-THAN SIGN
        U+003E ( > ) GREATER-THAN SIGN

bidi_filename := path_sep? char+ (path_sep char+)* path_sep?
path_sep :=
        U+002F ( / ) SOLIDUS
        U+005C ( \ ) REVERSE SOLIDUS

bidi_email := char* '@' domainWithTLD
```

There is one extra condition:

> When parsing a bidi_structure, an inclusionChar (such as “!”) is treated specially. If it is followed by a char, then it is included in the BNF. If not, it is excluded. For example, “abc.com/foo.bar” would be parsed completely as a bidi_IRI, but “abc.com/foo. other text” would stop before the period.

It is possible to capture this extra condition in the BNF, but it would make the formulation far less readable.

Open issues

- Are there other characters (or categories of characters) to include or exclude in this BNF?
- The inclusionChars are currently from ASCII/Latin1. Should any of the ASCII/Latin1 exceptions be changed to terminating characters instead?

Please provide feedback on these open issues.

To compare this simplified UBA syntax for a bidi_IRI with the full standard IRI syntax, see:
http://tools.ietf.org/html/rfc3987#section-2.2

Termination and Continuation

In summary, encountering any of the following classes of characters would cause parsing of a bidi_structure in plain text to continue:

1. Letters, Marks, Numbers
2. Dash and connector punctuation
3. Symbols

Encountering any of the following classes of characters would cause parsing of a bidi_IRI in plain text to terminate:

1. Unassigned, surrogates, private-use, control codes
2. Whitespace
3. Opening, closing or most ‘other’ punctuation, plus special cases < and >.

Many of these characters can be included in an IRI, but they would need to be % encoded for the specialized bidi display to kick in.

For ASCII and Latin1, the list of terminating punctuation consists of:

U+003C < LESS-THAN SIGN
U+003E > GREATER-THAN SIGN
U+0028 (LEFT PARENTHESIS

U+0029) RIGHT PARENTHESIS
U+005B [LEFT SQUARE BRACKET
U+005D] RIGHT SQUARE BRACKET
U+007B { LEFT CURLY BRACKET
U+007D } RIGHT CURLY BRACKET
U+00AB « LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
U+00BB » RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK

Proposed extension of UBA for bidi_IRIs

The proposed UBA extension would display any recognized bidi_IRI in the following way. The goal of this display is to ensure that the fields of bidi_IRI occur in a predictable, uniform order.

Characters are escaped in IRIs using a sequence like '%61' for 'a'. In each of the following options, any such sequence, that is, (%[0-9A-Fa-f]{2})+ is handled specially. It is displayed as though it were embedded in LRO..PDF.

For a bidi_IRI:

A *separator* is defined as any instance of the quoted strings in the bidi_IRI BNF:

- right after a scheme: "://"
- in a domain: IDNSep
- right after a domain: "/" , "?", "#"
- in a path: "/"
- right after a path: "?", "#"
- in a query: "=" or "&"
- right after a query: "#"

A *field* is defined as any text between separators, or at the front or end.

For a bidi_filename, a separator is defined as any path_sep.

For a bidi_email, the '@' is a separator, as is any IDNSep in the domain.

Option 1: Constant Order

Each bidi_structure is displayed with fields from left to right. Thus the examples shown above in Table 2 will instead appear with a predicatible order of their fields. The order is the same whether the bidi_structure is in a RTL or in a LTR environment (paragraph).

Table 3. Constant order

Environment	**Display**
LTR, RTL	http://ab.cd.com/mn/op http://ab.cd.**FE.HG.**com/**JI/LK/**mn/op http://**FE.HG/JI/LK**

Note that just the ordering of the fields is changed. The ordering of the characters *within* each field is unaffected; the rules for bidirectional display specified in the UBA still apply. For example, the field with memory order **EF** still displays from right to left, as **FE**.

An implementation of the UBA extension can accomplish the display of Table 3 by behaving *as if*:

- the entire bidi_IRI is embedded in an <LRE>...<PDF> sequence, and
- each field in that bidi_IRI is surrounded by LRMs.

If the field ordering of IRIs were consistently "big-endian", it would be useful to have their display ordering depend on the direction of the paragraph. However, IRIs are not consistently big-endian; the most important part, the domain, has its fields organized in little-endian order. For example, http://www12.sap.com/uk/about would be http://com.sap.www12/uk/about if it were in big-endian order.

Because the ordering of fields in an IRI is already inconsistent, this proposal is to have a consistent ordering always, no matter what the bidi environment is (RTL vs LTR).

The UTC is leaning towards this option, but would like feedback on this choice, whether pro or con.

Option 2. Environment Order

Alternatively, the ordering of fields could be subject to the environment (whether the current embedding level is RTL or LTR). Thus the examples shown above in Table 2 will instead appear as in Table 4:

Table 4. Environment order

Environment	**Display**
LTR	http://ab.cd.com/mn/op http://ab.cd.**FE.HG.**com/**JI/LK/**mn/op http://**FE.HG/JI/LK**

RTL	op/mn/com.cd.ab//:http
	op/mn**/LK/JI**/com**.HG.FE**.cd.ab//:http
	LK/JI/HG.FE//:http

An advantage of this option is that for IRIs that have only RTL letters, the appearance would more closely match what happens currently. A disadvantage is that the appearance of an IRI may change radically depending on the environment.

Option 3. Content Order

A third option would be to have the ordering not depend only on the environment, but instead depend on whether there were any RTL characters in the bidi_structure. Thus the examples shown above in Table 2 will instead appear as in Table 5:

Table 5. Content order

Environment	**Display**
LTR, RTL	http://ab.cd.com/mn/op
	op/mn**/LK/JI**/com**.HG.FE**.cd.ab//:http
	LK/JI/HG.FE//:http

Having the order of fields depend on whether there is any RTL character in the IRI, is problematic. Having or not having a RTL character in the path, query or fragment part would turn around the whole bidi_structure display, including any domain part. This is likely to be very confusing for users.

Option 4. TLD Order

A further suggested option is to use the direction of the TLD to determine the order. So if, say, the TLD had no LTR characters, the overall direction would be RTL. The appearance would be the same as in Table 4.

A significant advantage of this is that the order would be determinant, like that of Constant Order. A different strategy would need to be applied for file names, however.