```
Network Working Group                                         J. Klensin
Internet-Draft
Updates: 5892, 5894 (if approved)                           P. Faltstrom
Intended status: Standards Track                                  Netnod
Expires: July 10, 2015                                   January 6, 2015
```

# IDNA Update for Unicode 7.0.0
draft-klensin-idna-5892upd-unicode70-03.txt

## Abstract

The current version of the IDNA specifications anticipated that each new version of Unicode would be reviewed to verify that no changes had been introduced that required adjustments to the set of rules and, in particular, whether new exceptions or backward compatibility adjustments were needed. That review was conducted for Unicode 7.0.0 and identified a potentially problematic new code point. This specification discusses that code point and associated issues and updates RFC 5892 accordingly. It also applies an editorial clarification that was the subject of an earlier erratum. In addition, the discussion of the specific issue updates RFC 5894.

## Status of This Memo

## Copyright Notice

Table of Contents

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . 2
2. Problem Description . . . . . . . . . . . . . . . . . . . . . 5
2.1. IDNA assumptions about Unicode normalization . . . . . . . 5
2.2. New code point U+08A1, decomposition, and language dependency . . . . . . . . . . . . . . . . . . . . . . . 6
2.3. Other examples of the same behavior . . . . . . . . . . . 7
2.4. Hamza and Combining Sequences . . . . . . . . . . . . . . 8
3. Proposed/ Alternative Changes to RFC 5892 for new character U+08A1 . . . . . . . . . . . . . . . . . . . . . . . . . . 9
3.1. Disallow This New Code Point . . . . . . . . . . . . . . 9
3.2. Disallow the combining sequences for these characters . . 10
3.3. Do Nothing Other Than Warn . . . . . . . . . . . . . . . 11
3.4. Normalization Form IETF (or DNS) . . . . . . . . . . . . 11
4. Editorial clarification to RFC 5892 . . . . . . . . . . . . . 11
5. Acknowledgements . . . . . . . . . . . . . . . . . . . . . . 12
6. IANA Considerations . . . . . . . . . . . . . . . . . . . . . 12
7. Security Considerations . . . . . . . . . . . . . . . . . . . 12
8. References . . . . . . . . . . . . . . . . . . . . . . . . . 13
8.1. Normative References . . . . . . . . . . . . . . . . . . 13
8.2. Informative References . . . . . . . . . . . . . . . . . 15
Appendix A. Change Log . . . . . . . . . . . . . . . . . . . . . 15
A.1. Changes from version -00 to -01 . . . . . . . . . . . . . 15
A.2. Changes from version -01 to -02 . . . . . . . . . . . . . 15
A.3. Changes from version -02 to -03 . . . . . . . . . . . . . 15
Authors' Addresses . . . . . . . . . . . . . . . . . . . . . . . 15

## 1. Introduction

The current version of the IDNA specifications, known as "IDNA2008" [RFC5890], anticipated that each new version of Unicode would be reviewed to verify that no changes had been introduced that required adjustments to IDNA's rules and, in particular, whether new exceptions or backward compatibility adjustments were needed. When that review was carefully conducted for Unicode 7.0.0 [Unicode7], comparing it to prior versions including the text in Unicode 6.2 [Unicode62], it identified a problematic new code point (U+08A1, ARABIC LETTER BEH WITH HAMZA ABOVE). The specific problem is discussed in detail in Section 2. The behavior of that code point, while non-optimal for IDNA, follows that of a few code points that predate Unicode 7.x and even the IDNA 2008 specifications and Unicode 6.0. Those existing code points make the question of what, if anything, to do about this new one exceedingly problematic because different reasonable criteria yield different decisions, specifically:

- To disallow it as an IDNA exception case creates inconsistencies with how those earlier code points were handled.

- To disallow it and the similar code points as well would necessitate invalidating some potential labels that would have been valid under IDNA2008 until this time. However, there is reason to believe that no such labels exist.

- To permit the new code point to be treated as PVALID creates a situation in which it is possible, within the same script, to compose the same character symbol (glyph) in two different ways that do not compare equal even after normalization. That condition would then apply to it and the earlier code points with the same behavior. That situation contradicts a fundamental assumption of IDNA that is discussed in more detail below.

NOTE IN DRAFT:

This working draft discusses four alternatives, including, for illustration, a radical idea that seems too drastic to be considered now although it would have been appropriate to discuss when the IDNA2008 specifications were being developed. The authors suggest that the community discuss the relevant tradeoffs and make a decision and that the document then be revised to reflect that decision, with the other alternatives discussed as options not chosen. Because there is no ideal choice, the discussion of the issues in Section 2, is probably as or more important than the particular choice of how to handle this code point. In addition to providing information for this document, that section should be considered as an updating addendum to RFC 5894 [RFC5894] and should be incorporated into any future revision of that document.

As the result of this version of the document containing several alternate proposals, some of the text is also a little bit redundant. That will be corrected in future versions.

As anticipated when IDNA2008, and RFC 5892 in particular, were written, exceptions and explicit updates are likely to be needed only if there is disagreement between the Unicode Consortium's view about what is best for the Standard and the IETF's view of what is best for IDNs, the DNS, and IDNA. It was hoped that a situation would never arise in which the the two perspectives would disagree, but the possibility was anticipated and considerable mechanism added to RFC 5890 and 5982 as a result. It is probably important to note that a disagreement in this context does not imply that anyone is "wrong", only that the two different groups have different needs and therefore criteria about what is acceptable. For that reason, the IETF has, in the past, allowed some characters for IDNA that active Unicode Technical Committee members suggested be disallowed to avoid a change in derived tables [RFC6452]. This document describes a case where the IETF should disallow a character or characters that the various properties would otherwise treat as PVALID.

This document provides the "flagging for the IESG" specified by Section 5.1 of RFC 5892. As specified there, the change itself requires IETF review because it alters the rules of Section 2 of that document.

Readers of this document are expected to be familiar with Unicode terminology [Unicode62] and the IETF conventions for representing Unicode code points [RFC5137].

As a convenience to readers of RFC 5892 and to reduce the risks of confusion, this document also formally applies the content of an erratum to the text of the RFC (see Section 4) and so brings that RFC up to date with all agreed changes.

[[RFC Editor: please remove the following comment and note if they get to you.]]

[[IESG: It might not be a bad idea to incorporate some version of the following into the Last Call announcement.]]

NOTE IN DRAFT to IETF Reviewers: The issues in this document, and particularly the choices among options for either adding exception cases to RFC 5892 or ignoring the issue, warning people, and hoping the results do not include serious problems, are fairly esoteric. Understanding them requires that one have at least some understanding of how the Arabic Script works and the reasons the Unicode Standard gives various Arabic Script characters a fairly extended discussion [Unicode62-Arabic]. It also requires understanding of a number of Unicode principles, including the Normalization Stability rules [UAX15-Versioning] as applied to new precomposed characters and guidelines for adding new characters. There is considerable discussion of the issues in Section 2 and references are provided for those who want to pursue them, but potential reviewers should assume that the background needed to understand the reasons for this change is no less deep in the subject matter than would be expected of someone reviewing a proposed change in, e.g., the fundamentals of BGP, TCP congestion control, or some cryptographic algorithm. Put more bluntly, one's ability to read or speak languages other than English, or even one or more languages that use the Arabic script, does not make one an expert in these matters.

## 2. Problem Description

### 2.1. IDNA assumptions about Unicode normalization

IDNA makes several assumptions about Unicode, Unicode "characters", and the effects of normalization. Those assumptions were based on careful reading of the Unicode Standard at the time [Unicode5], guided by advice and commitments by members of the Unicode Technical Committee. Those assumptions, and the associated requirements, are necessitated by three properties of DNS labels that do not apply to blocks of running text:

1. There is no language context for a label. While particular DNS zones may impose restrictions, including language or script restrictions, on what labels can be registered, neither the DNS nor IDNA impose either type of restriction or give the user of a label any indication about the registration or other restrictions that may have been imposed.

2. Labels are often mnemonics rather than words in any language. They may be abbreviations or acronyms or contain embedded digits and have other characteristics that are not typical of words.

3. Labels are, in practice, usually short. Even when they are the maximum length allowed by the DNS and IDNA, they are typically too short to provide significant context. Statements that suggest that languages can almost always be determined from relatively short paragraphs or equivalent bodies of text do not apply to DNS labels because of their typical short length and because, as noted above, they are not required to be formed according to language-based rules.

At the same time, because the DNS is an exact-match system, there must be no ambiguity about whether two labels are equal. Although there have been extensive discussions about "confusingly similar" characters, labels, and strings, such tests between scripts are always somewhat subjective: they are affected by choices of type styles and by what the user expects to see. In spite of the fact that the glyphs that represent many characters in different scripts are identical in appearance (e.g., basic Latin "a" (U+0061) and the identical-appearing Cyrillic character (U+0430), the most important test is that, if two glyphs are the same within a given script, they must represent the same character no matter how they are formed.

Unicode normalization, as explained in [UAX15], is expected to resolve those "same script, same glyph, different formation methods" issues. Within the Latin script, the code point sequence for lower case "o" (U+006F) and combining diaeresis (U+0308) will, when normalized using the "NFC" method required by IDNA, produce the precombined small letter o with diaeresis (U+00F6) and hence the two ways of forming the character will compare equal (and the combining sequence is effectively prohibited from U-labels).

NFC was preferred over other normalization methods for IDNA because it is more compact, more likely to be produced on keyboards on which the relevant characters actually appeared, and because it does not lose substantive information (e.g., some types of compatibility equivalence involves judgment calls as to whether two characters are actually the same -- they may be "the same" in some contexts but not others -- while canonical equivalence is about different ways to produce the glyph for the same abstract character).

IDNA also assumed that the extensive Unicode stability rules would be applied and work as specified when new code points were added. Those rules, as described in The Unicode Standard and the normative annexes identified below, provide that:

1. New code points representing precombined characters that can be formed from combining sequences will not be added to Unicode unless neither the relevant base character nor required combining character are part of the Standard within the relevant script [UAX15-Versioning].

2. If circumstances require that principle be violated, normalization stability requires that the newly-added character decompose (even under NFC) to the previously-available combining sequence [UAX15-Exclusion].

There is no explicit provision in the Standard’s discussion of conditions for adding new code points, nor of normalization stability, for an exception based on different languages using the same script.

## 2.2. New code point U+08A1, decomposition, and language dependency

Unicode 7.0.0 introduces the new code point U+08A1, ARABIC LETTER BEH WITH HAMZA ABOVE. As can be deduced from the name, it is visually identical to the glyph that can be formed from a combining sequence consisting of the code point for ARABIC LETTER BEH (U+0628) and the code point for Combining Hamza Above (U+0654). The two rules summarized above suggest that either the new code point should not be allocated at all or that it should have a decomposition to \u'0628'\u'0654'.

Had the issues outlined in this document been better understood at the time, it probably would have been wise for RFC 5892 to disallow either the precomposed character or the combining sequence of each pair in those cases in which Unicode normalization rules do not cause the right thing to happen, i.e., the combining sequence and precomposed character to be treated as equivalent. Failure to do so at the time places an extra burden on registries to be sure that conflicts (and the potential for confusion and attacks) do not exist. Oddly, had the exclusion been made part of the specification at that time, the preference for precombined forms noted above would probably have dictated excluding the combining sequence, something not otherwise done in IDNA2008 because the NFC requirement serves the same purpose. Today, the only thing that can be excluded without the potential disruption of disallowing a previously-PVALID combining sequence is the to exclude the newly-added code point so whatever is done, or might have been contemplated with hindsight, will be somewhat inconsistent.

## 2.3. Other examples of the same behavior

One of the things that complicates the issue with the new U+08A1 code point is that there are several other Arabic-script code points that behave in the same way for similar language-specific reasons.

In particular, at least three other grapheme clusters that have been present for many version of Unicode can be seen as involving issues similar to those for the newly-added ARABIC LETTER BEH WITH HAMZA ABOVE. ARABIC LETTER HAH WITH HAMZA ABOVE (U+0681) and ARABIC LETTER REH WITH HAMZA ABOVE (U+076C) do not have decomposition forms and are preferred over combining sequences using HAMZA ABOVE (U+0654) [Unicode62-Hamza]. By contrast, ARABIC LETTER ALEF WITH HAMZA ABOVE (U+0623) decomposes into \u'0627'\u'0654' and ARABIC LETTER YEH WITH HAMZA ABOVE (U+0626) decomposes into \u'064A'\u'0654' so the precomposed character and combining sequences compare equal when both are normalized, as this specification prefers.

There are other variations in which a precomposed character involving HAMZA ABOVE has a decomposition to a combining sequence that can form it. For example, ARABIC LETTER U WITH HAMZA ABOVE (U+0677) has a compatibility (???) decomposition into the combining sequence \u'06C7'\u'0674'.

## 2.4. Hamza and Combining Sequences

As the Unicode Standard points out at some length [Unicode62-Arabic], Hamza is a problematic abstract character and the "Hamza Above" construction even more so [Unicode62-Hamza]. Those sections explain a distinction made by Unicode between the use of a Hamza mark to denote a glottal stop and one used as a diacritic mark to denote a separate letter. In the first case, the combining sequence is used. In the second, a precombined character is assigned.

Unlike Unicode generally and because of concerns about identifier spoofing and attacks based on similarities, character distinctions in IDNA are based much more strictly on the appearance of characters; language and pronunciation distinctions within a script are not considered. So, for IDNA, BEH WITH HAMZA ABOVE is not-quite-tautologically the same as BEH WITH HAMZA ABOVE, even if one of them is written as U+08A1 (new to Unicode 7.0.0) and the other as the sequence \u'0628'\u'0654' (feasible with Unicode 7.0.0 but also available in versions of Unicode going back at least to the version [Unicode32] used in the original version of IDNA [RFC3490]. Because the precomposed form and combining sequence are, for IDNA purposes, the same, IDNA expects that normalization (specifically the requirement that all U-labels be in NFC form) will cause them to compare equal.

If Unicode also considered them the same, then the principle would apply that new precomposed ("composition") forms are not added unless one of the code points that could be used to construct it did not exist in an earlier version (and even then is discouraged)[UAX15-Versioning]. When exceptions are made, they are expected to conform to the rules and classes in the "Composition Exclusion Table", with class 2 being relevant to this case [UAX15-Exclusion]. That rule essentially requires that the normalization for the old combining sequence to itself be retained (for stability) but that the newly-added character be treated as canonically decomposable and decompose back to the older sequence even under NFC. That was not done for this particular case, presumably because of the distinction about pronunciation modifiers versus separate letters noted above. Because, for IDNA and the DNS, there is a possibility that the composing sequence \u'0628'\u'0654' already appears in labels, the only choice other than allowing an otherwise-identical, and identically-appearing, label with U+08A1 substituted to identify a different DNS entry is to DISALLOW the new character.

## 3. Proposed/ Alternative Changes to RFC 5892 for new character U+08A1

NOTE IN DRAFT: See the comments in the Introduction, Section 1 and the first paragraph of each Subsection below for the status of the Subsections that follow. Each one, in combination with the material in Section 2 above, also provides information about the reasons why that particular strategy is appropriate.

### 3.1. Disallow This New Code Point

If chosen by the community, this subsection would update the portion of the IDNA2008 specification that identifies rules for what characters are permitted [RFC5892] to disallow that code point.

With the publication of this document, Section 2.6 ("Exceptions (F)") of RFC 5892 [RFC5892] is updated by adding 08A1 to the rule in Category F so that the rule itself reads:

```
   F: cp is in {00B7, 00DF, 0375, 03C2, 05F3, 05F4, 0640, 0660,
                0661, 0662, 0663, 0664, 0665, 0666, 0667, 0668,
                0669, 06F0, 06F1, 06F2, 06F3, 06F4, 06F5, 06F6,
                06F7, 06F8, 06F9, 06FD, 06FE, 07FA, 08A1, 0F0B,
                3007, 302E, 302F, 3031, 3032, 3033, 3034, 3035,
                303B, 30FB}
```

and then add to the subtable designated
"DISALLOWED -- Would otherwise have been PVALID"
after the line that begins "07FA", the additional line:

```
   08A1; DISALLOWED # ARABIC LETTER BEH WITH HAMZA ABOVE
```

This has the effect of making the cited code point DISALLOWED independent of application of the rest of the IDNA rule set to the current version of Unicode. Those wishing to create domain name labels containing Beh with Hamza Above may continue to use the sequence

U+0628, ARABIC LETTER BEH
followed by

U+0654, ARABIC HAMZA ABOVE

which was valid for IDNA purposes in Unicode 5.0 and earlier and which continues to be valid.

In principle, much the same thing could be accomplished by using the IDNA "BackwardCompatible" category (IDNA Category G, RFC 5892 Section 5.3). However, that category is described as applying only when "property values in versions of Unicode after 5.2 have changed in such a way that the derived property value would no longer be PVALID or DISALLOWED". Because U+08A1 is a newly-added code point in Unicode 7.0.0 and no property values of code points in prior versions have changed, category G does not apply. If that section of RFC 5892 were to be replaced in the future, perhaps consideration should be given to adding Normalization Stability and other issues to that description but, at present, it is not relevant.

## 3.2. Disallow the combining sequences for these characters

If chosen by the community, this subsection would update the portion of the IDNA2008 specification that identifies contextual rules [RFC5892] to prohibit (combining) Hamza Above (U+0654) in conjunction with Arabic BEH (U+0628), HAH (U+062D), and REH (U+0631). Note that the choice of this option is consistent with the general preference for precomposed characters discussed above but would ban some labels that are valid today and that might, in principle, be in use.

The required prohibition could be imposed by creating a new contextual rule in RFC 5892 to constrain combining sequences containing Hamza Above.

As the Unicode Standard points out at some length [Unicode62-Arabic], Hamza is a problematic abstract character and the "Hamza Above" construction even more so. IDNA has historically associated characters whose use is reasonable in some contexts but not others with the special derived property "CONTEXTO" and then specified specific, context-dependent, rules about where they may be used. Because Hamza Above is problematic (and spawns edge cases, as discussed in the Unicode Standard section cited above), it was suggested that a contextual rule might be appropriate. There are at least two reasons why a contextual rule would not be suitable for the present situation.

1. As discussed above, the present situation is a normalization stability and predictability problem, not a contextual one. Had the same issues arisen with a newly-added precomposed character that could previously be constructed from non-problematic base and combining characters, it would be even more clearly a normalization issue and, following the principles discussed there and particularly in UAX 15 [UAX15-Exclusion], might not have been assigned at all.

2. The contextual rule sets are designed around restricting the use of code points to a particular script or adjacent to particular characters within that script. Neither of these cases applies to the newly-added character even if one could imagine rules for the

use of Hamza Above (U+0654) that would reflect the considerations of Chapter 8 of Unicode 6.2. Even had the latter been desired, it would be somewhat late now -- Hamza Above has been present as a combining character (U+0654) in many versions of Unicode. While that section of the Unicode Standard describes the issues, it does not provide actionable guidance about what to do about it for cases going forward or when visual identity is important.

## 3.3. Do Nothing Other Than Warn

The recommendation from UTC is to simply warn registries, at all levels of the tree, to be careful with this set of characters, making language distinctions within zones. Because the DNS cannot make or enforce language distinctions, this suggestion is problematic but it would avoid having the IETF either invalidating label strings that are potentially now in use or creating inconsistencies among the characters that combine with Hamza Above but that also have precomposed forms that do not have decompositions. The potential would still exist for registries to respect the warning and deprecate such labels if they existed.

## 3.4. Normalization Form IETF (or DNS)

The most radical possibility would be to decide that none of the Unicode Normalization Forms specified in UAX 15 [UAX15] are adequate for use with the DNS because, contrary to their apparent descriptions, normalization tables are actually determined using language information. However, use of language information is unacceptable for IDNA for reasons described elsewhere in this document. The remedy would be to define an IETF-specific (or DNS-specific) normalization form, building on NFC but adhering strictly to the rule that normalization causes two different forms of the same character (glyph image) within the same script to be treated as equal. In practice such a form would be implemented for IDNA purposes as an additional rule within RFC 5892 (and its successors) that constituted an exception list for the NFC tables. For this set of characters, the special IETF normalization form would be equivalent to the exclusion discussed in Section 3.2 above.

## 4. Editorial clarification to RFC 5892

Verified RFC Editor Erratum 3312 [RFC5892Erratum] provides a clarification to Appendix A and Section A.1 of RFC 5892. This section of this document updates the RFC to apply that clarification.

1. In Appendix A, add a new paragraph after the paragraph that begins "The code point...". The new paragraph should read:

"For the rule to be evaluated to True for the label, it MUST be evaluated separately for every occurrence of the Code point in the label; each of those evaluations must result in True."

2. In Appendix A, Section A.1, replace the "Rule Set" by

```
Rule Set:
  False;
  If Canonical_Combining_Class(Before(cp)) .eq.  Virama Then True;
  If cp .eq. \u200C And
         RegExpMatch((Joining_Type:{L,D})(Joining_Type:T)*cp
    (Joining_Type:T)*(Joining_Type:{R,D})) Then True;
```

## 5. Acknowledgements

The Unicode 7.0.0 changes were extensively discussed within the IAB's Internationalization Program. The authors are grateful for the discussions and feedback there, especially from Andrew Sullivan and David Thaler. Additional information was requested and received from Mark Davis and Ken Whistler and while they probably do not agree with the necessity of excluding this code point or taking even more drastic action as their responsibility is to look at the Unicode Consortium requirements for stability, the decision would not have been possible without their input. Thanks to Bill McQuillan and Ted Hardie for reading versions of the document carefully enough to identify and report some confusing typographical errors. Several experts and reviewers who prefer to remain anonymous also provided helpful input and comments on preliminary versions of this document.

## 6. IANA Considerations

When the IANA registry and tables are updated to reflect Unicode 7.0.0, changes should be made according to the decisions the IETF makes about Section 3.

## 7. Security Considerations

[[CREF1: NOTE IN DRAFT: This section is unchanged in version -01 of this document relative to what appeared in -00. It will need to be rewritten once decisions are made about what path to follow. In particular, if "just warn" is chosen, it will need to contain very strong warnings.]]

This specification excludes a code point for which the Unicode-specified normalization behavior could result in two ways to form a visually-identical character within the same script not comparing equal. That behavior could create a dream case for someone intending to confuse the user by use of a domain name that looked identical to another one, was entirely in the same script, but was still considered different (see, for example, the discussion of false negatives in identifier comparison in Section 2.1 of RFC 6943 [RFC6943]). This exclusion therefore should improve Internet security.

## 8. References

### 8.1. Normative References

[RFC5137] Klensin, J., "ASCII Escaping of Unicode Characters", BCP 137, RFC 5137, February 2008.

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

[RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC5892Erratum] "RFC5892, "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", August 2010, Errata ID: 3312", Errata ID 3312, August 2012, <http://www.rfc-editor.org/errata_search.php?rfc=5892>.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, August 2010.

[RFC6943] Thaler, D., "Issues in Identifier Comparison for Security Purposes", RFC 6943, May 2013.

[UAX15] Davis, M., Ed., "Unicode Standard Annex #15: Unicode Normalization Forms", June 2014, <http://www.unicode.org/reports/tr15/>.

[UAX15-Exclusion] "Unicode Standard Annex #15: ob. cit., Section 5", <http://www.unicode.org/reports/tr15/#Primary_Exclusion_List_Table>.

[UAX15-Versioning] "Unicode Standard Annex #15, ob. cit., Section 3", <http://www.unicode.org/reports/tr15/#Versioning>.

[Unicode5]
The Unicode Consortium, "The Unicode Standard, Version 5.0", ISBN 0-321-48091-0, 2007.

Boston, MA, USA: Addison-Wesley. ISBN 0-321-48091-0. This printed reference has now been updated online to reflect additional code points. For code points, the reference at the time RFC 5890-5894 were published is to Unicode 5.2.

[Unicode62]
The Unicode Consortium, "The Unicode Standard, Version 6.2.0", ISBN 978-1-936213-07-8, 2012, <http://www.unicode.org/versions/Unicode6.2.0/>.

Preferred citation: The Unicode Consortium. The Unicode Standard, Version 6.2.0, (Mountain View, CA: The Unicode Consortium, 2012. ISBN 978-1-936213-07-8)

[Unicode62-Arabic]
"The Unicode Standard, Version 6.2.0, ob.cit., Chapter 8", Chapter 8, 2012, <http://www.unicode.org/versions/Unicode6.2.0/ch08.pdf>.

Subsection titled "Encoding Principles", paragraph numbered 4, starting on page 251.

[Unicode62-Hamza]
"The Unicode Standard, Version 6.2.0, ob.cit., Chapter 8", Chapter 8, 2012, <http://www.unicode.org/versions/Unicode6.2.0/ch08.pdf>.

Subsection titled "Combining Hamza Above" starting on page 263.

[Unicode7]
The Unicode Consortium, "The Unicode Standard, Version 7.0.0", ISBN 978-1-936213-09-2, 2014, <http://www.unicode.org/versions/Unicode7.0.0/>.

Preferred Citation: The Unicode Consortium. The Unicode Standard, Version 7.0.0, (Mountain View, CA: The Unicode Consortium, 2014. ISBN 978-1-936213-09-2)

## 8.2. Informative References

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

[RFC6452] Faltstrom, P. and P. Hoffman, "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA) - Unicode 6.0", RFC 6452, November 2011.

[Unicode32] The Unicode Consortium, "The Unicode Standard, Version 3.2.0", .

The Unicode Standard, Version 3.2.0 is defined by The Unicode Standard, Version 3.0 (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), as amended by the Unicode Standard Annex #27: Unicode 3.1 (http://www.unicode.org/reports/tr27/) and by the Unicode Standard Annex #28: Unicode 3.2 (http://www.unicode.org/reports/tr28/).

## Appendix A. Change Log

RFC Editor: Please remove this appendix before publication.

### A.1. Changes from version -00 to -01

- Version 01 of this document is an extensive rewrite and reorganization, reflecting discussions with UTC members and adding three more options for discussion to the original proposal to simply disallow the new code point.

### A.2. Changes from version -01 to -02

Corrected a typographical error in which Hamza Above was incorrectly listed with the wrong code point.

### A.3. Changes from version -02 to -03

Corrected a typographical error in the Abstract in which RFC 5892 was incorrectly shown as 5982.

## Authors' Addresses

John C Klensin
1770 Massachusetts Ave, Ste 322
Cambridge, MA  02140
USA

Phone: +1 617 245 1457
Email: john-ietf@jck.com

Patrik Faltstrom
Netnod
Franzengatan 5
Stockholm  112 51
Sweden

Phone: +46 70 6059051
Email: paf@netnod.se