mailmaint
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

A. Gulbrandsen
ICANN
J. Yao
CNNIC

7 July 2025

Interoperable Email Addresses for SMTPUTF8
draft-ietf-mailmaint-interoperable-addresses-01

Abstract

This document specifies rules for email addresses that are flexible enough to express the addresses typically used with SMTPUTF8, while avoiding elements that harm human-to-human interoperation.

This is one of a pair of documents: this contains recommendations for what addresses humans should use, such that address provisioning systems can restrain themselves to addresses that email valdidators accept. (This set can also be described in other ways, including "simple to cut-and-paste" and "understandable by some community".) Its companion defines simpler rules, accepts more addresses, and is intended for software like MTAs.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Mail Maintenance Working Group mailing list (mailmaint@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mailmaint/.

Source for this draft and an issue tracker can be found at https://github.com/arnt/mailmaint-smtputf8.

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 https://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 8 January 2026.

Table of Contents

1. Introduction

[RFC6530]-[RFC6533] and [RFC6854]-[RFC6858] extend various aspects of the email system to support non-ASCII both in localparts and domain parts. In addition, some email software supports unicode in domain parts by using encoded domain parts in the SMTP transaction ("RCPT TO:<info@xn--dmi-0na.fo<") and presenting the unicode version (d淡mi.fo in this case) in the user interface.

The email address syntax extension is in [RFC6532], and allows almost all UTF8 strings as localparts. While this certainly allows everything users want to use, it is also flexible enought to allow many things that users and implementers find surprising and sometimes worrying.

The flexibility has caused considerable reluctance to support the full syntax in contexts such as web form address validation.

This document attempts to describe rules that:

1. includes the addresses that users generally want to use for themselves and organizations want to provision for their employees.

2. includes the addresses that have been used significantly, even if not exactly what users wanted.

3. excludes confusable things.

4. excludes things that have been described as security risks.

5. Looks safe at first glance to implementers (including ones with limited knowledge of unicode).

These goals are somewhat aspirational. For example, lower-case L and upper-case i are confusable and cannot realistically be disallowed, the protocol poLIce would arrest us all.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

Script, in this document, refers to the unicode script property (see [UAX24]). Each code point is assigned to one script ("a" is Latin), except that some are assigned to "Common" or a few other special values. Fraktur and /etc/rc.local aren't scripts in this document, but Latin is.

Latin refers those code points that have the script property "Latin" in Unicode. Orl迪ans in France and M端nster in Germany both have Latin names in this document. It also refers to combinations of those code points and combining characters, and to strings that contain no code points from other scripts.

Han, Cyrillic etc. refer to those code points that have the respective script property in Unicode, as well as to strings that contain no code points from other scripts.

ASCII refers to the first 128 code points within unicode, which includes the letters A-Z but not or . It also refers to strings that contain only ASCII code points.

Non-ASCII refers to unicode code points except the first 128, and also to strings that contain at least one such code point.

By way of example, the address info@dmi.fo is latin and non-ASCII, its localpart is latin and ASCII, and its domain part is latin and non-ASCII. 中国 is a Han string in this document, but 阿Q正 is neither a Latin string nor a Han string, because it contains a Latin Q and three Han code points.

ZWJ, "zero-width joiner", is the unicode codepoint U+200D. ZWNJ, "zero-width non-joiner", is the unicode codepoint U+200C.

The terms a-label and u-label are defined in [RFC5890] section 2.3.2.1.

PRECIS is described in [RFC8264], and IdentifierClass in section 4.2 of that document.

4. An oversimplified description of current SMTPUTF8 email addresses

This section is informative.

In the countries the authors have visited, the norm is that some users and organizations want to use their daily script(s), but not others. An Indian student in Japan or an Indian immigrant in an Arabic country will generally have his/her name spelt with either Latin letters or the host country's script, because that's how the university or company works.

While the syntax defined in [RFC6532] technically supports addresses with an Indic localpart and a Japanese or Arabic domain part, organizations such as a university or a company don't want to use that kind of naming: The script that's used for the organization's domain is the one it wants to use when provisioning email addresses.

Note that even when an organization emphatically doesn't want to provision localparts from scripts other than its own, that organization generally wants the ability to correspond worldwide.

There are a few countries in which ASCII localparts are used with non-ASCII domains (reputedly because of email software that supports [RFC3490]/[RFC5891] but not [RFC6532]). So far, the authors have only seen this in countries that use left-to-right scripts such as Cyrillic and Han.

In some countries, ASCII non-letters is used together with non-Latin scripts. Arabic is an example: the Arabic digits 0-9 are used often (more so in some countries than others). Chinese domain names use the ASCII dot instead of the Chinese full-width dot. ASCII punctuation (notably the hyphen) is used with several scripts.

In the authors' experience, left-to-right and right-to-left writing is mixed in only a few cases (that are relevant to email addresses):

* ASCII digits (123) and right-to-left letters

* Arabic Indic digits () and Arabic letters

* Testcase domains/addresses

* single punctuation characters (often neutral in direction)

5. An oversimplified description of IDNs and the domain name system

This section is informative.

The use of non-ASCII in domain names is restricted by several factors: IDNA2008 rules, registry rules and web browsers.

For a domain such as example.com, IDNA2008 restricts both labels (slightly), ICANN policy restricts "com" and the .com registry's policy restricts "example", and the browsers' rules apply to both. For a domain such as beispiel.example.com, IDNA2008 and the browsers' rules apply to "beispiel" as well. Neither ICANN's nor the .com registry's rules apply to "beispiel".

5.1. IDNA2008

Using non-ASCII more or less requires IDNA2008, ie. if a the domain name is to be practically usable, it needs to be usable with IDNA2008, which restricts the set of code points slightly.

5.2. Registry rules

The LGR, Label Generation Rules, are a set of rules largely developed by per-language and per-script committees and collected by ICANN. Most notably, a script or language has to be used by a current community in order to get an LGR. An LGR contains that which is currently used by the community.

Registering a domain name in a public TLD requires adhering to the registry's policy, which is generally either an LGR, a small registry-chosen set of code points, or restricted by rules outside the DNS (such as for .bank, which is effectively restricted by what names banking regulators allow banks to have).

The merged repertoire of all code points in all LGRs is called the Common LGR, and contains more than 30,000 code points at the time of writing.

5.3. Web browser rules

The main browsers have rules for which domains are displayed in a human-readable manner in the address bar. (One author has a runic domain, all of the main browsers display xn--something in the address bar.) Each of the three big browser vendors maintains its own rule set. At the time of writing, all three may be described as broadly similar to the Common LGR and different in detail.

6. Rules for interoperable email addresses

Based on the above descriptions, the following rules are formulated for interoperable email addresses. (See below rationales for each.)

1. An atom in an interoperable address MUST NOT be an a-label.

2. If the address matches the grammar in [RFC5322], then in order to be interoperable it MUST also match the grammar specified by the W3C WHATWG [TYPE_EMAIL] spec.

3. An address MUST contain only code points within the PRECIS IdentifierClass.

4. An address MUST consist entirely of a sequence of composite characters, ZWJ and ZWNJ. ("c" followed by "combining hook below" is an example of a composite character, "d" is another example; see [RFC6365] for the definition.)

5. If an address contains any right-to-left code points, then it MAY contain ASCII digits and MUST NOT contain any other left-to-right code points.

6. If an address contains any non-ASCII code point, then one of the following conditions MUST apply:

   6.1. All code points share one unicode script property, or have the script property Common, or are ASCII digits (or ./@).

   6.2. The localpart consists entirely of ASCII, and the domain part consists of code points that share one unicode script property, or have the script property Common, or are ASCII digits (or ./@).

   6.3. All code points are have the script properties Han or Common, or are ASCII.

7. Examples

The address example@example.com is interoperable, because 1) it does not contain any a-label, 2) it matches the WHATWG [TYPE_EMAIL] spec, 3) it consists entirely of permissible code points, 4) it consists of 19 composite characters, and the last two conditions do not apply.

The address dmi@dmi.fo is interoperable, because 1) it does not contain any a-label, 2) does not apply, 3) it consists entirely of permissible code points, 4) it consists of 12 composite characters, 5) does not apply and 6) it consists entirely of ’Latin’ and ’Common’ code points (and ./@).

The address U+200E ’@’ U+200F ’.’ U+200E is not interoperable, because 4) U+200E and U+200F are not parts of composite characters.

The address U+627 ’1’ ’@’ U+627 ’2’ ’.’ U+627 ’3’ is interoperable. (U+627 is an arabic letter, written right-to-left.) It is interoperable because 1) it does not contain any a-label, 2) does not apply, 3) it consists entirely of permissible code points, 4) it consists of 8 composite characters, 5) the only right-to-left code points used are ASCII digits and 6) all code points are ’Arabic’ or ASCII digits (or @/).

(Note that it’s interoperable even though top-level domain used does not exist, because ’3’ is not permitted by the current top-level domain name rules. Checking domain existence is simple if one assumes that internet access is available and the address is valid at the time, but this document does not assume assume either of those.)

The address info@xn--dmi-0na.fo is not interoperable, because 1) it contains the a-label xn--dmi-0na.

The address 名字@例子.中国 is interoperable, because 1) it does not contain any a-label, 2) does not apply, 3) it consists entirely of permissible code points, 4) it consists of 8 composite characters, 5) does not apply and 6) it consists entirely of ’Han’ code points (and ./@).

The address info@例子.中国 is interoperable, because 1) it does not contain any a-label, 2) does not apply, 3) it consists entirely of permissible code points, 4) it consists of 8 composite characters, 5) does not apply and 6) the localpart is ASCII and the domain part consists entirely of ’Han’ code points (and .).

The address dmi@例子.中国 is not interoperable, because 6) it contains both ’Latin’ and ’Han’ code points and the localpart is non-ASCII.

The address 阿Q@阿Q正@.中国 is interoperable, because 1) it does not contain any a-label, 2) does not apply, 3) it consists entirely of permissible code points, 4) it consists of composite characters, 5) does not apply and 6) the address consists entirely of ASCII and Han code points (and .).

8. Test Suite

Daniel Eggert suggested that this document (and its companion) need a large formal test suite, he suggested that it use JSON. This section is a placeholder.

9. IANA Considerations

This document does not require any actions from the IANA.

10. Security Considerations

When a program renders a unicode string on-screen or audibly and includes a substring supplied by a potentially malevolent source, the included substring can affect the rendering of a surprisingly large part of the overall string.

This document describes rules that make it difficult for an attacker to use email addresses for such an attack. Implementers should be aware of other possible vectors for the same kind of attack, such as subject fields and email address display-names.

If an address is signed using DKIM and (against the rules of this document) mixes left-to-right and right-to-left writing, parts of both the localpart and the domain part can be rendered on the same side of the '@'. This can create the appearance that a different domain signed the message.

The rules in this document permit a number of code points that

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/rfc/rfc5322>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/rfc/rfc5890>.

[RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, DOI 10.17487/RFC6365, September 2011, <https://www.rfc-editor.org/rfc/rfc6365>.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/rfc/rfc6530>.

[RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/rfc/rfc6532>.

[RFC6533] Hansen, T., Ed., Newman, C., and A. Melnikov, "Internationalized Delivery Status and Disposition Notifications", RFC 6533, DOI 10.17487/RFC6533, February 2012, <https://www.rfc-editor.org/rfc/rfc6533>.

[RFC8264] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 8264, DOI 10.17487/RFC8264, October 2017, <https://www.rfc-editor.org/rfc/rfc8264>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

11.2. Informative References

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, DOI 10.17487/RFC3490, March 2003, <https://www.rfc-editor.org/rfc/rfc3490>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/rfc/rfc5891>.

[RFC6854] Leiba, B., "Update to Internet Message Format to Allow Group Syntax in the "From:" and "Sender:" Header Fields", RFC 6854, DOI 10.17487/RFC6854, March 2013, <https://www.rfc-editor.org/rfc/rfc6854>.

[RFC6858] Gulbrandsen, A., "Simplified POP and IMAP Downgrading for Internationalized Email", RFC 6858, DOI 10.17487/RFC6858, March 2013, <https://www.rfc-editor.org/rfc/rfc6858>.

[UAX24] Whistler, K., "Unicode Script Property", n.d., <https://unicode.org/reports/tr24>.

[UMLAUT] "Metal Umlaut", n.d., <https://en.wikipedia.org/wiki/Metal_umlaut>.

[TYPE_EMAIL]
"WHATWG input type=email", n.d.,
<https://html.spec.whatwg.org/multipage/input.html#email-state-(type=email)>.

Appendix A. Acknowledgments

The authors wish to thank John C. Klensin, Jeremy Harris, [your name here, please] [oh wow, the ack section is already outdated]

Dmi.fo and 例子.中国 are reserved by nic.fo and CNNIC for use in examples and documentation.

阿Q正@ is a famous Chinese novella, 阿Q is the main character.

Appendix B. Rationales for each condition

This section is informative. Each of the six conditions has a separate rationale.

1. A-labels are confusing for many readers, and can potentially be used to confuse and attack readers. Being visibly safe is one of the five goals.

2. Many existing address validators use the WHATWG rules; if this specification is exactly compatible with WHATWG [TYPE_EMAIL] for all the addresses that WHATWG covers, then it’s possible to extend a WHATWG-compliant validator without risk of accidentally rejecting formerly accepted addresses.

3. The PRECIS IdentifierClass is one of several similar sets, UAX31 and the Common LGR being others. I’m quite uncertain which is most appropriate, there are arguments in favour of each.

4. Unicode contains many code points that could perhaps be used for attacks. Whether they could be used for attacks is not important, since one of the goals is to be safe at first glance even to implementers with limited knowledge of unicode. By constraining the repertoire to the plainest code points, the specification gains safety at first glance.

5. Mixed-direction text can be confusing, and confusion has been used to attack users before. This rule tries to gain safety at first glance by constraining mixed-direction text in addresses to that which is known to be necessary.

6. This rule permits addresse that are e.g. all-Chinese or all-Thai, and rejects addresses that mix e.g. Thai and Chinese. This restricts the scope for visually confusable code points. Since some communities are known to mix ASCII localparts with IDNs, combining left-to-right text with ASCII is allowed, at least in the way that’s currently used.

Note that metal umlauts ("Mtley Cre") are allowed (see [UMLAUT]). This is an unintentional feature.

Appendix C. Instructions to the RFC editor

Please remove all mentions of the Protocol Police before publication (including this sentence).

Please remove the Open Issues section.

Appendix D. Open issues

1. The use of PRECIS IdentifierClass seems correct, but both UAX31 and the Common LGR are very similar and might work as well.

2. The relationship with RFC 8265 needs to be explained somewhere. A starting point: This document tries to guard against confusing addresses, in the sense that they confuse humans. Computers can also be confused; this document relies on RFC 8265 to make two confusable addresses practically equal. If two addresses look confusable, but test as identical according to 8264/5, then the confusability shouldn’t be a problem.

3. Metal umlauts might be a problem. Accents are used sometimes with non-latin, but very seldom and might be seen as surprising to native users of e.g. cyrillic, even if к в а р и у м exists. https://krebsonsecurity.com/2022/11/disneyland-malware-team-its-a-puny-world-after-all/ is worth considering. Not entirely clear how to subdivide the Common script so it can be used with some scripts, not with others.

4. Test suite.

Authors’ Addresses

Arnt Gulbrandsen
ICANN
6 Rond Point Schumann, Bd. 1
1040 Brussels
Belgium
Email: arnt@gulbrandsen.priv.no

Jiankang Yao
CNNIC
No.4 South 4th Zhongguancun Street
Beijing
100190
China
Email: yaojk@cnnic.cn