Universal Acceptance

Email Address Internationalization (EAI): A Technical Overview

UASG012

2018-05-19

TABLE OF CONTENTS

This introduction to Email Address Internationalization (EAI) is intended for technically capable email application developers and systems administrators who want to add EAI support to mail software. For a high-level view of EAI, see UASG014: Quick Guide to EAI. For background information on mail and related technologies, see Other Sources of Advice at the end of this document.

Why do we need EAI?

When modern Internet mail was defined in the 1980s, the net's working language was English and all mail was written using the ASCII character set, which only has unaccented upper and lower case Roman characters.

But nearly every other written language requires characters outside the ASCII set. On the modern Internet, mail users live in every country in the world and write in a vast array of languages, and email has been slowly evolving to handle all of them. Unicode can represent all those characters. EAI allows Unicode characters, specifically Unicode characters encoded as UTF-8,[1] in email addresses.

Standards and interoperation

The Internet works because the software on the computers connected to it is designed to *interoperate*. The software is written so that, for example, any computer can send mail to any other computer running mail software. The software all implements standards that describe how to do all of the things the Internet does, from the lowest level packet switching to the codes for formatted HTML text pages and file attachments.

For software to interoperate, everyone has to implement the standards in the same way. In this document, we emphasize how to do that, and how to avoid incompatibilities that could prevent interoperating. In particular, even though there are often changes that might seem "better," if they don't match the standards, they won't interoperate with other systems, which is of course not better at all.

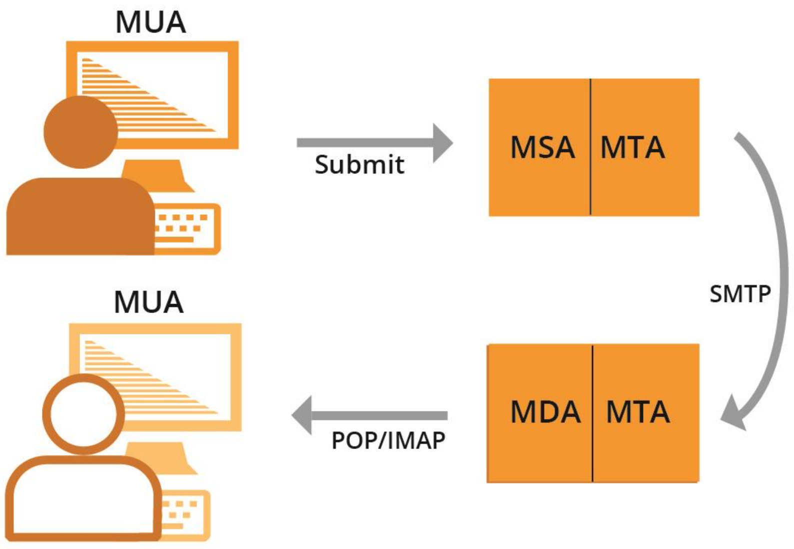
Parts of the mail ecosystem

The path that an email message takes from the sender to the recipient can be rather complex. To make it easier to understand, the IETF mail community has adopted a mail architecture that uses standard names for the various parts of the mail system and the connections among them. The architecture is described in detail in RFC 5598, *Internet Mail Architecture*.

Here we briefly look at the common parts of the architecture and how they fit together as a message travels from author to recipient.

[1] UTF-8 is a standard way of storing Unicode code points (characters and modifiers) in 8-bit bytes. For historic reasons, some software uses the name UTF8 without the hyphen.

The Mail Software Ecosystem



The Internet mail architecture has agreed-on names for the software that handles mail, mostly referred to by three- or four-letter abbreviations.

Mail User Agent (MUA)

Human mail users read and send mail with a *Mail User Agent* (MUA), the only program in the mail ecosystem with which users directly interact.

Popular MUAs include Outlook, Apple Mail, and the mail programs that people use on smartphones. The user composes the message, perhaps using formatting tools, optionally attaches other media, and then sends the message off using the SUBMIT protocol.

For incoming mail, the MUA retrieves mail from mail server(s) using POP or IMAP and displays it to the user.

In both cases, the format of the message that the user sees usually differs from the form in which it is submitted or received. Some parts of the message are reformatted, e.g., the `Date:` header information is often shown in the user's local format and time zone. Message contents are often decoded and displayed, e.g., a JPEG attachment will be displayed as an image.

MUAs usually let the user use address books, where the user can store email address(es) and other information about frequent correspondents. In some cases, the address book is stored on the user's computer, while other address books can be shared and accessed over the network. Address book entries can be quite complex, with multiple email addresses and other contact information for each correspondent.

Webmail

Webmail creates a mail client using a combination of the user's web browser and web servers that the browser connects to. In some versions, the web server uses standard mail protocols to send mail to and retrieve mail from separate mail servers. In others, the web server performs the functions of the MUA or other parts of the mail system. In either case, the webmail provides features similar to those that a separate MUA would perform.

Mail Submission Agent (MSA)

Once users have created an email message, they tell the MUA to submit it for delivery. The MUA connects to the *Mail Submission Agent* (MSA). Then the MUA authenticates the message with a password (or implicitly authenticates it with an IP address), and sends the envelope and the message. As the MSA receives the message, it checks for and fixes minor errors in the message format such as a missing `Date:` or `Message-ID:` header. Depending on local policy it may force the `From:` header to match the authenticated user. Many MSAs also add cryptographic authentication signatures at this stage. (See the discussion of DKIM below.)

Once the message is prepared, the MSA sends it to the *Mail Transmission Agent* (MTA). Depending on the design of the mail system, the MSA and MTA may be separate programs, or merely different functions in the same program.

The protocol between the MUA and MSA is similar to SMTP (Simple Mail Transport Protocol), which is used between MTAs, but has some significant differences. It's sometimes called SUBMIT to distinguish it from SMTP.

Mail Transmission Agent (MTA)

A *Mail Transmission Agent* (MTA) is what is most often meant by the phrase "mail server." It receives a completed message from an MSA, and then, for each recipient of the message, determines how and where to deliver the message. In general, for each recipient address, the MTA first finds the domain in the address to see if it's one that the MTA handles locally. If not, it does a DNS lookup for MX (mail exchanger) records to find the host computers that do handle it, makes an SMTP connection to one of those computers, and sends the message.

In the simplest mail systems, the host listed in the MX handles mail delivery itself, while in more complex systems a message may be passed several times from one computer to the next.

The SMTP transaction is a sequence of commands from the sending system and responses from the recipient system. The first command and response sets up the connection and lists the SMTP extensions that the recipient system supports, such as the `SMTPUTF8` extension that enables EAI mail. Subsequent commands send the envelope information (the return address and recipient addresses) and then send the message itself.

For more details on SMTP, see Appendix C: How does SMTP Work?.

Mail Delivery Agent (MDA)

Once a message has made its way through the recipient's MTA, it is sent to the recipient's Mail Delivery Agent (MDA) for delivery. In the simplest case, the MDA simply stores the message into a file for later pickup, but in most cases, there is some delivery-time processing.

The IETF has defined a language called SIEVE that enables mail users to write scripts that specify delivery-time processing procedures like sorting mail into separate mailboxes or redirecting mail to other addresses. There are many other delivery programs, such as the widely used procmail. Most Unix and Linux systems allow users to specify arbitrary programs to handle incoming mail.

Webmail systems provide various sorts of delivery-time processing. Most enable users to sort mail based on text found in the message (e.g., if the subject includes “pickle-list”, it is saved to the pickle-list folder) while others group mail into categories such as Updates or Forums.

Mail storage

Originally, an email mailbox was analogous to a physical mailbox, a place to store incoming mail messages temporarily until they were picked up and processed. As computer storage has gotten faster and cheaper, mailboxes have grown to the point where it is now common to store many years of mail. Mail systems generally have multiple folders for each mailbox, with each folder named hierarchically, as disk files are. Messages can be flagged as “unread,” “read,” “important,” or “answered.” Mail programs generally provide ways to organize mail, and to search through messages in various ways, looking for keywords or addresses, or limiting the search by date range.

Mail systems use many underlying storage schemes. In the simplest, all of the mail in a mailbox is stored in a single file. Others store each message in a separate file. Mail may also be stored in a database, perhaps with header and body information handled differently. Regardless of the storage scheme, MUAs see the mail as a list of messages in each folder.

POP/IMAP mail pickup

On early mail systems, users stored their email and ran their mail programs on the same computer, and the mail program read the mail directly from local files. These days, users invariably read their mail on a different computer, using POP and IMAP to retrieve it.

POP (Post Office Protocol), the older and simpler protocol, lets the MUA retrieve mail from the mail server and then optionally delete it from the server. The MUA then manages the mail on the user’s own computer.

IMAP (Internet Message Access Protocol) lets the user’s mail program manage mailboxes on the server, copying mail to and from the server as desired. Since the mail generally stays on the server, IMAP makes it possible to have mail programs on different computers, such as laptops and phones, show the same view of the mail. Since the mail program doesn’t have to download all the mail, vast amounts of mail can be stored while running mail programs on small computers such as tablets and phones.

Another important difference between POP and IMAP is that POP can only see the main inbox for a mail account, while IMAP can manage all of an account’s folders. The expectation was that POP users would copy all the mail to their own computer and manage it there.

POP and IMAP both require that a user authenticate before picking up mail, with a username and password. In most cases the POP/IMAP user name is the user’s email address.

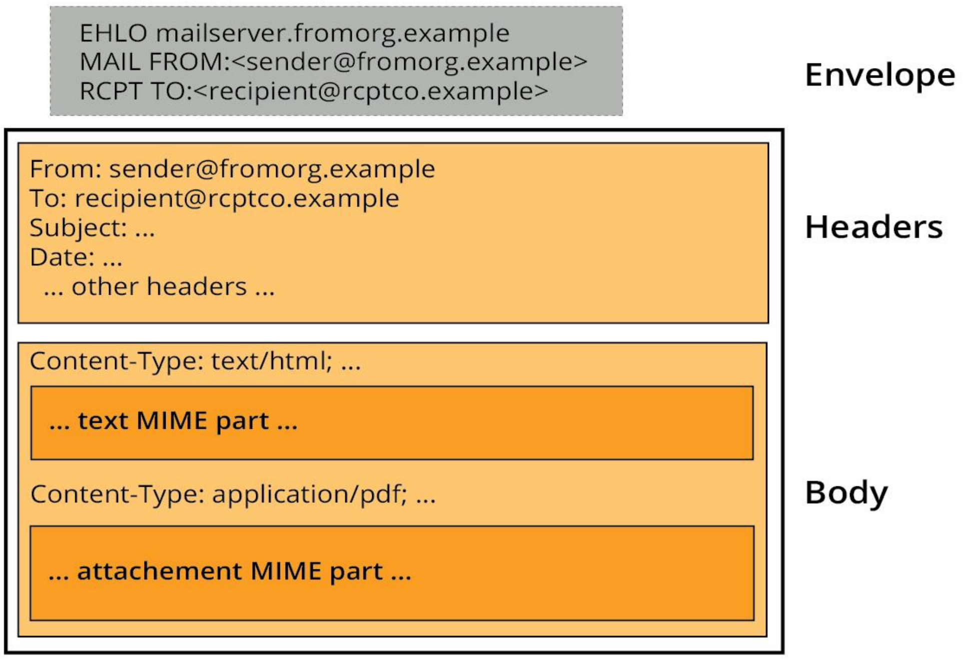
Mail message formats

The basic Internet message format was initially defined by RFC 724 in 1977 and has changed very little over the subsequent 40 years as the specification has been updated several times, most recently by [RFC 5822](). That format has proven to be flexible enough to support the changing needs of mail users over the years.

Each legacy message, i.e., a pre-EAI message, is a sequence of lines of ASCII text ending with the ASCII Carriage Return and Line Feed (CR and LF) characters. The first part of the message, usually called the header, is a series of structured header fields followed by a blank line and ending with an unstructured message body. Each header field starts with a header name such as `From:` or `Subject:` and a colon, followed by the contents of the header. Some header field contents are free-format, such as the `Subject:` header, while others are entirely fixed-format, such as the `Date:` and `Message-ID:` headers, while others are a combination of fixed- and free-format, such as the `To:`, `From:` and `Cc:` headers, which combine fixed-format addresses with free-format comment text.

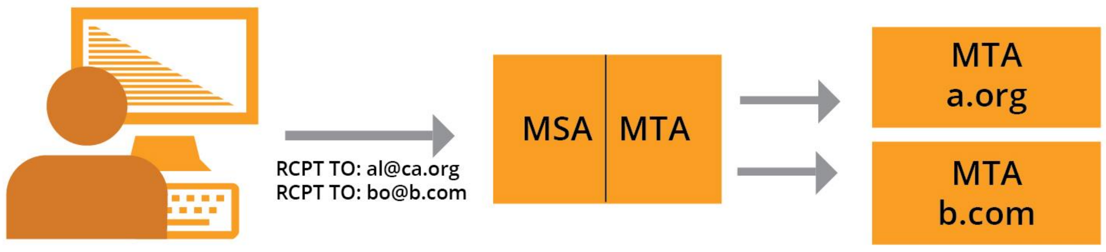
Messages are invariably accompanied by metadata, information related to the message but not part of it. The envelope, described in the next section, is an important part of the metadata. Other parts of the metadata may include the date a message was received, whether the recipient has read it, and whether the recipient has marked it as important or as junk.

Messages and Envelopes


EHLO mailserver.fromorg.example
MAIL FROM:<sender@fromorg.example>
RCPT TO:<recipient@rcptco.example>
Envelope
From: sender@fromorg.example
To: recipient@rcptco.example
Subject: ...
Date: ...
... other headers ...
Headers
Content-Type: text/html; ...
... text MIME part ...
Content-Type: application/pdf; ...
... attachement MIME part ...
Body

Email messages in transit, from submission to delivery, are accompanied by an *envelope*. The envelope contains the *return address*, to which errors should be reported (often called the “bounce address”), the *recipient addresses*, to which the message should be delivered, and sometimes other information about the delivery process. The envelope information is sent along with—but separately from—the message. When a message is transmitted via SMTP (described in more detail below), one SMTP command sends the return address, additional SMTP commands send each recipient address, then another SMTP command sends the message.

When a message starts out, the return address in the envelope usually matches the `From:` address in the message, and the recipient addresses in the envelope match the `To:`, `Cc:`, and `Bcc:` addresses. But there is no requirement that the envelope addresses match the ones in the message, and there are many common situations where they don't. For example, if a message is sent to two addresses, Alice and Bob, which are on different mail systems, the copy sent to Alice will only have recipient address Alice, and the copy sent to Bob will only have recipient address Bob. When mail is sent to mail discussion lists, the `To:` line address is usually the address of the list, but the list software will re-mail the message to all of the list's subscribers, which means that all of those subscribers will be in the message's envelope recipients.



In addition to addresses, the envelope can include other parameters, such as the size of the message or the identity of the sender of a newly submitted message. As we will see below, in EAI mail, the envelope indicates whether a message is an EAI message or a legacy message.

Mail addresses

Each legacy email address is of the form *local-part@domainname*. The local-part is arbitrary ASCII text, and the domain name is a DNS hostname. A legacy hostname is a sequence of labels separated by dots. Each label starts and ends with an ASCII letter or digit and consist of only ASCII letters, digits and hyphens, known as the LDH rule.

In principle, legacy local-parts can contain any printable ASCII characters, including spaces and punctuation. In practice names with exotic punctuation are hard to to type and often don't work well when entered into web forms.

In mail headers, addresses are usually enclosed in angle brackets to distinguish them from comment text, e.g.:

```
To: Jim Smith <james@example.com>, Mary Jones <mzjones@example.net>
```

An obsolete but still valid syntax puts the comments in parentheses, e.g.

```
To: james@example.com (Jim Smith), mzjones@example.net (Mary Jones)
```

Internationalized Domain Names

Just as Internet users want to use their own scripts in the body of their emails, they also want to use their own scripts in the domain names. Although the Domain Name System (DNS) internally uses eight-bit bytes and could, in principle, have used literal UTF-8 names, a vast amount of DNS support software can only handle ASCII. In addition, Unicode often allows several different ways to produce the same character, e.g., writing an accented "é" either with two code points, a plain "e" and a separate accent character "´", or with a single, precomposed code point for the accented letter. DNS lookups only do exact matches, so if a name were stored in one form but looked up in another, the lookup would fail.

The DNS community has addressed this problem with a design known as Internationalized Domain Names in Applications (IDNA.) It uses several techniques to make UTF-8 names work reasonably well. EAI uses the most recent version of IDNA, IDNA2008.

Normalization

The first is normalization, using a consistent representation of characters that can be represented in UTF-8 in multiple ways, e.g., a single "é" code point or an "e" and an accent "´". The Unicode consortium has defined several normalization forms. IDNs use Normalization Form C (NFC), where C stands for Composed. Roughly speaking, it uses precomposed characters that include accents and other modifiers rather than separate letter and modifier code points.

Inclusion

The second technique is *inclusion*, the principle that specific characters are deliberately included in the allowed set, rather than starting with everything and trying to delete the problematic ones. The IDNA specifications identify the code points that are allowed in domain names, which are generally intended to include letters and digits used in a wide variety of languages, as well as—in some cases—additional characters needed to combine or format them. They deliberately exclude all the other Unicode code points such as punctuation and emoji. A UTF-8 string that complies with all of the IDNA rules is called a U-Label.

ACE Encoding

The third technique is ASCII Compatible Encoding (ACE). Since a lot of existing software (not just mail software) only handles ASCII, IDNA created an alternate ASCII encoding called an A-Label. Every valid U-Label corresponds to a unique A-Label and vice versa. The A-Label is stored in the DNS, while the U-Label is typically entered by users and displayed to them.

An A-Label is in the form `xn--`*`text`*`,` where `xn--` is a prefix indicating that a label is an A-Label, and the text is an ASCII-encoded version of the U-Label, using a scheme known as punycode. IDNA is widely supported in web browsers, which will turn UTF-8 names in their address bars into A-labels before looking them up.

Scripts and LGR

The fourth technique is comprised of scripts and *Label Generation Rules* (LGR.) A script is a collection of characters used to write a language. The relationship between scripts and languages can be complex—some scripts such as Cyrillic are used for several languages, while some languages such as Japanese are written in multiple scripts. Label Generation Rules define the set of Unicode characters valid for a particular language. The main point of LGRs is to avoid mixed script names that are confusing for users (e.g., mixtures of Latin and Cyrillic), and to identify variant characters that look different but have the same meaning.

LGRs are not formally a part of IDNA2008, but IETF standards recommend using them. Most top-level domain registries have LGRs for each of the languages in which they accept registrations. IANA keeps a repository of them at https://www.iana.org/domains/idn-tables.

UTS#39, Unicode Security Mechanisms, offers advice on restricting mixing of scripts. Its Highly Restrictive restriction level is a conservative and widely used rule for identifiers.

The part of an email address after the @ is a domain name, so A-labels are valid there. But the part of the email address before the @, the local-part, is not a domain name, and for a variety of reasons, a UTF-8 local-part cannot be encoded as punycode or an A-label.[2] To have proper internationalized mail, we need an extension to the mail system to handle UTF-8 in the local-part part of the address, as well as in the other nooks and crannies that MIME doesn't handle.

MIME mail

MIME (Multipurpose Internet Mail Extensions), the first major extension to mail, appeared in 1992. Before MIME, the body of a mail message was just a block of unstructured ASCII text. MIME provided a way to treat the mail body as a group of structured blocks of data, some of which might be text, while others might be pictures, documents, or any other sort of data that could be stored in a file. MIME provides standard ways to encode any kind of data. When pictures or other files are attached to a mail message, it's done with MIME.

MIME overlays a structure of message parts on the block of unstructured text. A message can be one part or several, the parts can be related in different ways, and each part can have a type. Typical types are `text/plain` and `text/html` for plain or HTML-formatted text, or `application/pdf` for an attached PDF file. A message with more than one part can be `multipart/alternative`, different versions of the same thing (typically unformatted and formatted text), or `multipart/related`, typically text and attachments.

Every MIME text part can specify a character set such as UTF-8, which allows UTF-8 message bodies even in legacy mail.

MIME also added a feature called 8BITMIME, which allows the transfer of messages that include 8-bit characters in message bodies so long as the body is still lines of text with CR and LF at the end. This allows the use of UTF-8 text since all ASCII characters, including CR and LF, are also UTF-8 characters. In principle, mail server support of 8BITMIME is optional, but in practice, all current mail servers support it. Messages can also use *quoted-printable* or *base64* encoding to represent any data including UTF-8 in legacy messages.

MIME also provides a way to specify encoded character sets in the unformatted parts of message headers such as the `Subject:` line, using the same character sets and encodings that are allowed in text bodies. These are called encoded-words.

[2] Since mail programs accept and display local parts as-is, users would see the meaningless punycode or A-label rather than the intended UTF-8 address. Even worse, users would have to enter the characters in the punycode or A-label when typing an address.

For example, this subject line is an encoded-word in UTF-8. This example includes a UTF-8 character hex C2 B0, which is a raised small letter "o".

```
Subject: =?utf-8?Q?Your_reservation_N=C2=B0_F39-04XS?=
```

Subject: Your reservation Nº F39-04XS

MIME encoded-words can be used in most mail headers visible to users, including the `Subject:` and the comment parts of address headers such as `To:` and `From:`, but it can't be used in email addresses in legacy mail, which still have to be ASCII.

It's worth emphasizing this last point: *other than addresses, almost everything in a mail message can already use UTF-8 via MIME*.

A first look at EAI mail

The technical differences between legacy and EAI mail are quite small. The most important difference is that email addresses (the local-part and/or domain name) can contain UTF-8 as well as ASCII. The local-part part of an address can contain any printable UTF-8 characters. The characters in a domain name are limited to those allowed by IDNA2008.

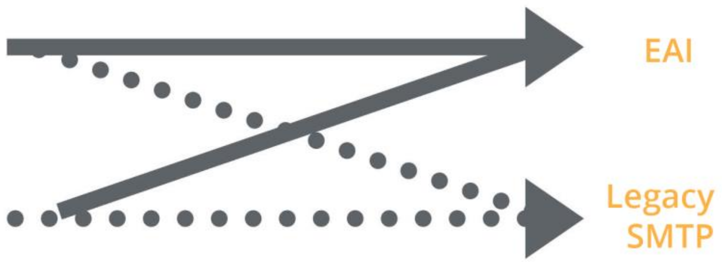
Another difference is that most parts of a message can be UTF-8 rather than ASCII. The contents of header fields such as `To:`, `From:`, and `Subject:` can be UTF-8, although the field names are still restricted to ASCII. The contents of a message can be UTF-8 without needing any special encoding.

There are a few other differences, but these are the most important ones. Since there is no way for a legacy mail system to handle UTF-8 addresses in a message envelope, or UTF-8 addresses in mail headers, EAI had to modify both SMTP and message formats to handle the new address forms.

Transmitting EAI mail

EAI creates a conceptually parallel mail stream for EAI mail. When one computer transmits an EAI message to another, such as when an MUA submits a message to an MSA, or an MTA transmits a message to another MTA, it needs to be sure that the receiving system can deal with it. (We say conceptually parallel because any EAI mail server can support legacy mail.)

The standard MUA→MTA submission and MTA→MTA SMTP have defined an extension feature called `SMTPUTF8`. If a mail server supports `SMTPUTF8` it can handle EAI mail; if not, it can't. When the sending computer first connects, the receiving computer sends it a list of supported extensions (see EHLO in Appendix C below) and the sending computer checks to see if `SMTPUTF8` is in the list.



This picture shows the two conceptually separate mail streams, EAI and legacy SMTP. The solid diagonal line indicates that any EAI mail server is also a legacy mail server; that is, any legacy sender can send to a legacy address on an EAI server. The dotted diagonal line indicates that in some cases (not many), EAI mail can be turned back into legacy mail.

Recognizing EAI messages

Since there are likely to be parallel EAI and legacy mail streams for a long time to come, it will sometimes be necessary to distinguish between EAI and legacy messages. There are two ways to do this.

One is to look at the message you need to identify as EAI. If there are UTF-8 characters in any of the envelope addresses, or UTF-8 characters in any of the headers in the message, it's an EAI message.[3] In addition, the end of the header is marked by an empty line, coded as the four characters CR LF CR LF. This identification process could potentially be slow if a message's headers were large, or if it would require reading the message from a disk file.

The other way of distinguishing an EAI message is to keep its EAI status as part of the message's metadata. The metadata already includes the envelope addresses, and—in many systems—other information such as receipt time, whether it's been read or replied to, and whether it's considered junk. In some cases, one more indicator for EAI status would be easy to include.

[3] This is easy to do in software because all UTF-8 character codes are greater than 0x80, whereas ASCII codes are less than 0x80.

Technical topics for EAI mail

Most changes are in the SMTP session and in message headers, but there are a few changes in user mail programs, and some new options for the message bodies, too.

Assigning and interpreting EAI mail addresses

Many of the conventions from ASCII addresses carry over to EAI mail addresses: EAI local parts can contain any printable UTF-8 characters, while the domain parts are still domain names but IDN names can be written as UTF-8 U-labels.

Any address with UTF-8 in the local-part, the domain, or both is an EAI address. While the email standards say that a local-part can contain any sequence of UTF-8 characters up to 64 bytes long, some local-parts are a lot easier for users to remember and type than others.

The IETF PRECIS[4] (Preparation and Comparison of Internationalized Strings) working group created standard ways to handle string input used in applications. It defines classes of strings for different applications. The *Identifier* Class is intended, as its name suggests, for strings used as identifiers. The class has a profile *UsernameCaseMapped* which matches mailbox names fairly well, including case-insensitive letters and digits but not spaces or punctuation. PRECIS also defined preparation rules to preprocess user input, in order to deal with multiple versions of characters such as upper and lower case or full- and half-width letters.

An additional advantage of using a Username profile is that internationalized usernames for POP and IMAP servers (described below) have to be PRECIS usernames, and this allows the user's mailbox to be used as the POP/IMAP username. See the list of RFCs in Appendix D for links to detailed PRECIS specifications.

The Unicode consortium offers similar advice in UTS#39 Unicode Security Mechanisms, which offers advice for email addresses in section 3.3.

In some cases it may be useful to assign both an EAI and a legacy address for a mailbox. (See Downgrading, below.) In some cases there may be a straightforward transliteration, such as борис@domain to boris@domain or 李伟@domain to liwei@domain. In other cases, there may be no natural way to transliterate, and the two names may have no obvious connection.

Mail standards require that systems treat addresses of mailboxes on other systems as opaque identifiers; that is, to pass them along as is without trying to interpret or modify them. This means that systems must not even attempt to normalize text to NFC, since there is no promise that the recipient system uses that normalization, or any normalization.

But the standards allow the delivering system a great deal of flexibility in handling mail addressed to its own users. For legacy addresses, upper- and lower-case ASCII are usually treated as the same, and systems often accept minor variants of an address, such as

[4] Pronounced as the French *précis*, roughly like pray-see.

addresses with or without periods, apostrophes, or other punctuation (e.g., J.O’Brian@ and J.obrian@ and jobrian@ can deliver to the same mailbox).

EAI mail systems can allow similar flexibility in UTF-8 addresses. See the PRECIS documents for advice on preparing and normalizing string input.

Storing EAI mail

The traditional folder structure used for legacy mail doesn’t change with EAI, but the folder names can now be UTF-8, and metadata associated with messages such as `To:` and `From:`addresses and `Subject:`headers can be UTF-8 as well.

The PRECIS *FreeFormText* class, defined in RFC 8264, is intended for free-form strings such as “human-friendly nicknames” that can be useful for folder names. This allows a wider range of text than the Identifier class, including spaces and some punctuation.

MUAs and EAI mail

The changes to MUAs to handle EAI mail are not large.

In all the places where a user can enter a legacy mail address, such as `To:`and `Cc:` headers and address book entries, any validation of addresses at the time of entry must allow EAI addresses.

EAI addresses may contain *Left-to-right (LTR)*, *Right-to-Left (RTL)* or *bidirectional (bidi)* text, when for example, an address has an RTL local part and an ASCII domain name. MUAs must be prepared to display such addresses when they occur in header fields or in the message body. A discussion of the best ways to display bidi text is beyond the scope of this document. See Appendix D for references.

When user-entered text in messages is linkified by the MUA, the linkification process should treat all domain names and all email addresses the same.

Address books

Address books might provide a way to mark whether or not an address is able to receive EAI mail. While any EAI address can receive EAI mail, legacy addresses may or may not. If a user receives an EAI message, the sender of the message can be marked as EAI-ready.

If a contact in an address book has both an EAI and a legacy address, the EAI address is likely to be preferred for EAI mail, although the user should be able to override the choice.

Mail addresses on the web

Many websites accept email addresses through web forms, to be used for sending confirmations, adding to mailing lists, and other purposes. The addresses are generally validated either by javascript in the web page, or by code on the server to which the form is sent. Javascript typically uses regular expressions to check the address syntax. A fairly complete set of address expressions are in the IETF Internet draft draft-seantek-mail-regexen.

When a server receives an address, it should do some checks beyond what the javascript can do. If the domain part of the address is in UTF-8, it should normalize it following IDNA2008 rules and then verify that the domains are valid U-labels by converting them to A-labels. If the conversion fails, the address is invalid and the user needs to correct it. Once the server has the A-label version of the domain, it can look up the domain in the DNS to be sure it exists and that it has MX, A, or AAAA DNS records. If the domain doesn't exist, or doesn't have MX, A, or AAAA records, it's not valid for mail.

There is no way to validate the local part of an address other than sending mail to the address and seeing if the user receives the message. (In particular, doing a partial SMTP session to see if a `RCPT TO:` command succeeds is not a reliable check.) Sending a confirmation message to check that the address is valid and the person at that address is the one who provided the address is an email best practice whether or not the address is EAI.[5]

Email addresses can appear in mailto: URIs in formatted text. If an EAI address appears in a mailto: URI, the URI specifications[6] require that any non-ASCII characters must be percent-encoded in the URI. If an address were 猫王@普遍接受-测试.世界, the mailto: URI would be:

```
mailto:%e7%8c%ab%e7%8e%8b@%e6%99%ae%e9%81%8d%e6%8e%a5%e5%8f%97%2d%e6%b5%8b
%e8%af%95.%e4%b8%96%e7%95%8c
```

The Chinese characters are percent-encoded but the ASCII dot and at-sign are not. The domain name can be a U-label or an equivalent A-label, e.g.:

mailto:%e7%8c%ab%e7%8e%8b@xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g

Sending, delivering and picking up EAI mail

The following discussion assumes the reader is already familiar with the way that SMTP and SUBMIT work. See Appendix C for an overview of SMTP.

When an MTA uses SMTP to send a message to another MTA, or an MUA SUBMITs a message to an MSA, there a a few changes to the SMTP and SUBMIT protocols. EAI makes the same changes to SMTP and to SUBMIT, so we describe the changes together here.

The first difference is that the receiving computer includes the keyword `SMTPUTF8` in the list of extension keywords to say that it can handle EAI mail:

```
S: <connect>
R: 220 receive.net ESMTP
S: EHLO sender.org
R: 250-8BITMIME
R: 250-SMTPUTF8
R: 250 PIPELINING
```

The order of the keywords doesn't matter, but `SMTPUTF8` has to be in the list. (Every EAI mail server must also support 8BITMIME, but all modern servers do anyway.)

[5] See M³AAWG Sender Best Common Practices Version 3.0.

[6] Defined in RFC 3987.

Next, when sending the return address, the `MAIL FROM:` command includes an `SMTPUTF8` keyword to indicate that this message is an EAI message. UTF-8 addresses are sent as-is.

S: `MAIL FROM:<猫王@普遍接受-测试.世界> SMTPUTF8`
R: `250 Sender accepted`

If the body of the message contains 8-bit data such as UTF-8, the sender may indicate it in the `MAIL FROM`, although it’s not required.

S: `MAIL FROM:<猫王@普遍接受-测试.世界> SMTPUTF8 BODY=8BITMIME`
R: `250 Sender accepted`

The rest of the mail session is unchanged from legacy mail, other than allowing UTF-8 in the envelope recipient addresses and the message headers.

If a sending computer is trying to send an EAI message, and the `EHLO` response from the server doesn’t contain `SMTPUTF8`, that server can’t handle EAI mail and the delivery fails. Similarly, if the receiving computer rejects the `MAIL FROM` that includes a `SMTPUTF8` keyword, the delivery fails.

MTAs and incoming A-label and U-label addresses

An MTA usually handles mail for multiple specific domains.[7] An EAI mail server should accept mail both for the A-label and U-label versions of the domains whose mail it accepts. While EAI senders should use the U-label, legacy senders will use the A-label, and it’s also possible that poorly configured EAI senders will, too. Once a message is received, the address should be treated the same whether the address was an A-label or a U-label. An easy way to do that is to normalize incoming addresses so they’re all A-labels or all U-labels before subsequent processing.

POP and IMAP with EAI mail

POP and IMAP servers that support EAI need two kinds of changes.

The first change allows UTF-8 in usernames and passwords as well as IMAP folder names and search strings. In each case, the server needs to be able to tell clients that it supports the new features.

The other change allows POP clients to fetch EAI messages, and IMAP clients to fetch messages and store them back to the server. This requires that the client tell the server that it can handle EAI messages.

[7] Some servers are both an MTA and an MSA. If a sender authenticates before sending mail, the server is being used as an MSA, so it will accept and relay mail to addresses in any domains. An MTA that accepts and relays mail for any domain without needing authentication is known as an *open relay,* and is likely to be abused by spammers and blacklisted.

Both POP and IMAP support a variety of ways to send login credentials. Each has a command to send a plain text username and password as well as a more complex authentication command that can use a variety of security schemes known collectively as SASL, for Simple Authentication and Security Layer.[8]

POP3 support for EAI

POP3 allows client MUAs to fetch mail from mail servers, and is intended to be easy to implement on both servers and clients.

POP3 uses a sequence of four-letter commands and simple responses. Clients use the `CAPA` command to get a list of available capabilities and—for some capabilities—options.

The basic UTF8 capability indicates that the server can handle EAI mail.

The "`UTF8 user`" capability (that is, UTF8 with the "user" option) means that it can also handle UTF-8 login usernames and passwords.

Any UTF-8 username and password must be prepared using the PRECIS username profile (or its predecessor, SASLprep) and be sent with an `AUTH` command that uses a SASL mechanism.

Once the client has logged in, it sends a `UTF8` command to tell the server that it (the client) can handle EAI messages. After that, the session is the same as for legacy clients. If a client does not send a `UTF8` command but the server has EAI messages, the server either omits EAI messages from the list of available messages, or downgrades them. (See the Backward Compatibility section below for more about downgrading.) POP3 only handles a single mailbox per connection, so there are no folder names to deal with.

IMAP support for EAI

IMAP has a complex design that allows client MUAs to manage multiple mailboxes on remote servers. The client can use complex commands to manage large mailboxes on the server such as `SEARCH` to look for messages that match various criteria, and `SORT` and `THREAD` to control the order messages are returned from the server to the client. Over the years there have been several attempts to add internationalization features to IMAP; these coexist uneasily.

Like POP3, IMAP uses capabilities to indicate which optional features a server supports. An IMAP server sends a list of initial capabilities in the greeting message when the client first connects, and another list after the client logs in. (Many capabilities are only available after a client logs in.) The client then sends an `ENABLE` command with a feature name to tell the server that it wants to use that feature.

[8] SASL is defined in RFC 4422, and the various mechanisms are listed in the IANA SASL Mechanisms registry.

For a long time IMAP has had features to allow non-ASCII text in some contexts and different collating sequences for `SORT` and `THREAD`, but they are clumsy to use and often slow, so clients rarely use them.

In 2013 IMAP added nearly full support for UTF-8 with features `UTF-8=ACCEPT` and `UTF-8=ONLY`. If a server offers `UTF-8=ACCEPT` and the client enables it, the client and server can send each other UTF-8 in most places that required ASCII before. That includes mailbox names and mail messages sent from the server to the client, and mailbox names, search strings, and mail messages sent from the client to the server.

Since the client cannot enable `UTF-8=ACCEPT` until after it logs in, the IMAP `LOGIN:` command is still ASCII-only. The IMAP `AUTHENTICATE` command allows SASL authentication methods that handle UTF-8, so an EAI client can use `AUTHENTICATE:` to log in, then enable `UTF-8=ACCEPT`.

If the server offers `UTF-8=ONLY`, it is saying that it *only* provides EAI-compatible service. The client must enable `UTF-8=ACCEPT` or disconnect. Since ASCII is a subset of UTF-8, the client can access both ASCII and EAI folders and messages.

If a server offers `UTF-8=ACCEPT` but the client does not enable it, the server has the option to downgrade UTF-8 mailbox names and EAI messages. See Backward Compatibility, below.

EAI message headers

Changes to message headers in EAI mail range from none to very extensive, depending on the header. See Appendix A for advice on individual headers.

Headers with addresses

Headers with addresses include `From:`, `Sender:`, `To:`, `Cc:`, `Bcc:`, `Reply-To:`, `Resent-From:`, `Resent-To:`, `Resent-Cc:`, and `Resent-Bcc:`. All have the same syntax, i.e., a comma-separated list of addresses and optional comments. The addresses in EAI messages can have any printable UTF-8 text in the local part, and U-labels as well as A-labels and ASCII hostnames in the domain part. U-labels are preferred over A-labels, since human users can read them.

UTF-8 text in the comments can be MIME encoded as in legacy mail, but does not have to be. Since the unencoded UTF-8 is shorter, it is preferred.

Header fields that users see

Some header fields are free text intended for users to read, such as `Subject:` and `Organization:`. They can contain any printable UTF-8 text. The text should be unencoded UTF-8, but (as in legacy messages) can also be MIME encoded-words.

Header fields with contents for machines

Some header fields, such as `Date:` and `Message-ID:`, are intended for computers, not for people. In general, there is no advantage to using non-ASCII characters in those headers, and doing so makes it harder to send mail to legacy systems.

Even though content of the `Date:` header has a form familiar to English speakers, it is in a rigid fixed format defined in RFC 5322, which must be followed exactly. MUAs often parse the headers and display the dates in a convenient localized form, translating from the fixed form in the message. Other headers that contain dates such as Resent-Date are treated the same way.

The `Message-ID:` header contains a unique string that identifies the message. Mail software can check the `Message-ID:` in two messages to see if they are duplicates. Reply messages put the original `Message-ID:` into `References:` and `In-Reply-To:` headers that let MUAs group related messages together. The syntax of a `Message-ID:` is similar to an email address in brackets, like `<text@text>`. Most mail systems create Message-IDs with their own domain name to the right of the @ sign and pseudo-random text to the left. Although it is valid to use UTF-8 characters in Message-IDs, it's not a good idea, since it makes it impossible for a legacy message to refer to the EAI message in an `In-Reply-To:` or `References:` header. IDN domain names in Message-IDs should be represented as ASCII A-labels rather than U-labels.

Note: Even though Message-IDs resemble email addresses, they are actually just strings with no address meaning. It is not possible to downgrade a Message-ID; if you change U-labels in a Message-ID into A-labels, that is a different Message-ID unrelated to the original.

Received headers

Every time a message is transferred from one computer to another by SMTP, the receiving computer puts a `Received:` header at the top of the message. The header identifies the sending and receiving computers ("from" and "by"), the transmission protocol ("with"), a timestamp, and often other information. Here's a typical example for a message sent via SMTP using TLS encryption:

```
Received: from mail.mit.edu ([128.31.0.31])
 by mail1.iecc.com ([64.57.183.56]) with ESMTPS
 via TCP port 40060/25 id 586470169; 15 Feb 2018 02:28:32 -0000
```

The "with" clause indicates which transport was used to transport the message. Currently the most common are those listed below. When a message uses EAI transport—that is, with an `SMTPUTF8` tag on the `MAIL FROM:` that introduces the message— the "with" value changes to note the EAI transport.

Legacy transport	EAI transport	Meaning
ESMTP	UTF8SMTP	SMTP session starting with EHLO
ESMTPS	UTF8SMTPS	ESMTP with TLS encryption
ESMTPA	UTF8SMTPA	ESMTP from an authenticated sender (typically for SUBMIT)
ESMTPSA	UTF8SMTPSA	ESMTPA with TLS encryption

So, for example, a message sent using EAI SMTP and TLS would have this different `Received:` header.

```
Received: from mail.mit.edu ([128.31.0.31])
 by mail1.iecc.com ([64.57.183.56]) with UTF8SMTPS
 via TCP port 40060/25 id 586470169; 15 Feb 2018 02:28:32 -0000
```

While users don't usually see the Received headers, they are invaluable in tracking transport and delivery issues. The "with" values make it easy to tell which transports used EAI and which didn't. If it seems odd that the SMTP extension keyword is `SMTPUTF8` but the Received "with" tag defined in the same document is `UTF8SMTP`, yes, it is.

MIME headers

Most messages have more than one part, such as alternative plain text and HTML versions or attached files. The parts in the message are separated by a boundary string which is defined in a `Content-Type:` header. For example, these lines might appear in the message header:

```
MIME-Version: 1.0
Content-Type: multipart/alternative;
 boundary="b1_d9ef97d65ef88703f84b0a36e52aa4a2"
```

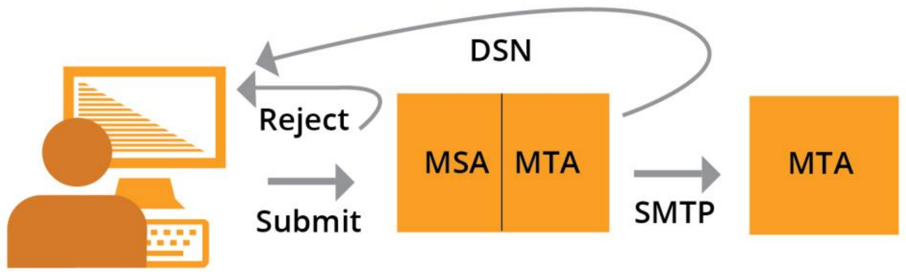
And these in the message body:

```
--b1_d9ef97d65ef88703f84b0a36e52aa4a2
Content-Type: text/plain; charset=utf-8

… text version of the message ...

--b1_d9ef97d65ef88703f84b0a36e52aa4a2
Content-Type: text/html; charset=utf-8

… HTML version of the message ...

--b1_d9ef97d65ef88703f84b0a36e52aa4a2--
```

Each part starts with the boundary string, and then a `Content-Type:` header with the type of the material in the part, the encoding, and sometimes other information such as the suggested filename for an attached file. An optional `Content-Description:` header contains free text about the MIME part, and `Content-ID:` contains an ID with the same syntax as a `Message-ID:` that allows one MIME part to refer to another.

EAI makes almost no changes to MIME headers, and EAI messages should continue to use ASCII characters as boundary strings. UTF-8 may appear in optional `Content-Type:` fields such as the suggested filename for an attached file, and in the text in `Content-Description`.

Handling and reporting delivery failure



Whenever one computer sends a message to another using SUBMIT or SMTP, things can fail. As the sending computer sends commands, the receiving computer sends responses saying whether each command succeeded or failed. In most cases, if any of the commands fail, sending the message fails.[9]

If a message is rejected inline, during an SMTP or SUBMIT session, specific error codes indicate why it was rejected. EAI adds some new rejection codes to indicate that it was rejected because a recipient does not accept an EAI message. The codes are different depending on which SMTP command provoked the rejection:

- 550 cannot accept EAI sender 5.6.7 (in response to `MAIL FROM`)
- 553 cannot accept EAI recipient 5.6.7 (in response to `RCPT TO`)
- 554 some recipients cannot accept EAI 5.6.9 (after the end of `DATA`)

After the receiving computer accepts a message, it either has to deliver the message, or send back a Delivery Status Notice (DSN) to the address in the `MAIL FROM:` command. Although mail systems try to reject undeliverable mail during the SMTP or SUBMIT session, some DSNs are unavoidable.

Assume a user uses an MUA to create an EAI message and submits it to the user's own mail system. That system's MTA uses SMTP to relay it to the recipient's system. If the recipient's system turns out not to accept EAI mail, the SMTP session will fail, and then the user's system has to send a DSN back to the user's MUA.

The format of DSNs for legacy messages is defined by RFC 3464, and extended to EAI messages by RFC 6533. A DSN is a mail message in a specific format, containing a MIME multipart/report structure with three parts: a text explanation for a human recipient; a fixed-format, machine-readable section describing what went wrong; and a copy of the failing message, or at least its headers.

[9] If a message has several recipients, the sender can send the message even if some of the recipient addresses are rejected, so long as at least one is accepted. It's a quality of implementation decision whether to send anyway or stop and go back to the user to fix the addresses.

A DSN might be returned a few seconds after the initial message was sent, or it might take hours or occasionally days if a recipient system is unavailable and an intermediate system retries the delivery several times before giving up.

Backward compatibility

For the foreseeable future EAI mail will have to coexist with legacy mail. While an EAI mail system can accept any mail from legacy mail systems, legacy systems can only accept legacy mail.

EAI and legacy recipients

When a user sends a message to a correspondent with an EAI address or to several EAI correspondents, they can obviously all accept EAI mail. If the correspondent has an ASCII address, and the address is in the user's address book, it might be marked as EAI-compatible or not. In the absence of that information, one possibility is to be conservative and send a legacy message, while the other is to send an EAI message, hope that it works, and report the failure if it doesn't.

As discussed in the previous section, if a message cannot be delivered, the sending system may get an immediate failure code, or the failure report may come back later in a DSN, depending on the architecture of the system.

If a message has more than one recipient, some recipients may accept EAI messages while others do not. In that case the sending system may try to create a downgraded message and send that to the non-EAI recipients. Note that SMTP makes no distinction among `To:`, `Cc:`, or `Bcc:` recipients on a message. The process of downgrading is the same regardless of where—if anywhere—the message header mentions the non-EAI recipient. Hence the process of downgrading and resending depends only on whether there are non-EAI recipients.

Whether to downgrade

The downgrading advice in the following sections is completely optional. A compliant EAI system need not do any downgrading at all.

While the EAI standards allow senders to downgrade messages, they neither encourage nor discourage it. A reason to downgrade is that legacy recipients can get approximate copies of messages they couldn't get otherwise. But a reason *not* to downgrade is that downgrading messages loses some of the information, and recipients of downgraded messages may find them confusing or frustrating. It is also not possible to tell reliably whether a recipient with an ASCII address can receive EAI mail, so a system that downgrades is likely to guess wrong some of the time and send downgraded messages to people who could have accepted the EAI original.

Another problem is that downgrading removes EAI addresses from message headers. In some cases it is possible to replace them with substitute addresses, but more often an address is just gone. Without a substitute address, recipients cannot reply to the message, put the person in their address books, or do any of the other things they do with addresses in incoming mail. Downgraded messages may be handled poorly by mail systems that do not expect `From:` header fields that don't contain an address. If messages are downgraded automatically, the sender is likely to be unaware that some or all of the recipients of the message have gotten a downgraded version, and users may find it confusing that different recipients got different versions of the message.

Where mail can be downgraded

When an MTA attempts to send a message to a recipient, the attempt may fail because the recipient MTA doesn't offer SMTPUTF8, or rejects the message with a status code that indicates that the recipient doesn't accept EAI mail. At that point the MTA might attempt to create a downgraded version of the message to send to the recipient. In theory any MTA in the message's path might do this, but the result is more likely to be useful if the MTA can find substitute addresses for the EAI addresses in the downgraded message. The most likely situation for a successful downgrade is a webmail system that integrates the MUA, MSA, and outgoing MTA into a single package.

Another place that messages might plausibly be downgraded is in mailing list manager software. If some subscribers to a list can accept EAI mail and some can't, the list software might create a downgraded version for legacy recipients.

General downgrade techniques

The EAI standards describe two methods of downgrading received EAI messages on POP or IMAP servers. These two approaches are also, with minor changes, suitable for downgrading messages by senders with messages for non-EAI recipients.

RFC 6857, *Post-Delivery Message Downgrading for Internationalized Email Messages*, describes a complex scheme that creates headers such as `Downgraded-To:` and `Downgraded-From:` that are intended to include everything that was in the message in a way that could be reversed later. RFC 6858, *Simplified POP and IMAP Downgrading for Internationalized Email*, describes a much simpler scheme that preserves EAI information where convenient, but does not try to preserve every detail. We recommend the simpler RFC 6858 approach.

Headers and bodies

In most cases, only the message's headers need to be downgraded to be acceptable to non-EAI mail systems. In theory there could be systems that only accept 7-bit ASCII mail, but in practice all mail systems now support 8BITMIME, so they can accept UTF-8 in message bodies.

Downgrading headers with addresses

Headers with addresses such as `From:`, `To:`, and `Resent-Cc:` contain addresses and optional comments. Since the comments can be encoded words that are valid in legacy messages, only the addresses need special handling when downgraded. Downgrading a UTF-8 address has two steps: determine whether there is a substitute ASCII address, and then rewrite the header appropriately.

If the address is one managed by the local system (generally the case for addresses in the `From:` header), there may be a list of locally equivalent EAI and ASCII addresses. If so, find the equivalent ASCII address in that list. Also, if the user has an address book that is available at the time of the downgrade, the address book may have a substitute address for the EAI address.

In addition, there is one uncommon case where it is possible to downgrade an address mechanically to a substitute ASCII address. If an address is of the form `<ascii@U-labels>`— that is, an ASCII local-part and a UTF-8 domain-part—it is permissible to replace the U-labels in the domain with A-labels, producing an ASCII address. This is the *only* mechanical downgrade possible. In particular, there is no mechanical downgrade possible from a UTF-8 local-part.

If there is a substitute address, downgrade the address by replacing the original address with the ASCII substitute.

If there is no substitute, downgrading the address removes the original address from the header and replaces it with an empty address group, written as a colon followed by a semicolon `:;`.[10] It is possible to copy the original EAI address into an address comment to show the recipient what the address was, but the legacy recipient cannot reply to it.

Consider this example, where capital letters indicate UTF-8 text, and underlined text means a MIME encoded-word.

Before:	`From: “CCCCC” <MMMM@DDDD.DDD>`
After:	`From: “CCCCC MMMM@DDDD.DDD” :;`

The comment (CCCCC) and the EAI address (MMMM@DDDD.DDD) are moved into an encoded-word comment, and the address is replaced by an empty group to keep the header syntactically valid. The exact details of the encoded-word comment don’t matter because it’s a comment. Note that the rewritten form no longer contains an address that the recipient can reply to.

This example is a `From:` address but the same modifications work on all address headers. If there is more than one address in an header, handle each one separately, converting EAI addresses and leaving legacy addresses alone.

Downgrading other headers that users see

Headers displayed to users can use MIME encoded-words to represent UTF-8 text. For any UTF-8 text in a message header, turn any UTF-8 into encoded-words. This includes text in `Subject:`, `Comment:`, and `Keyword:` headers, and many semi-standard headers such as `Organization:`.

Downgrading headers that users do not see

As mentioned above, `Message-ID:` headers should not have non-ASCII characters. If the Message-ID nonetheless does have non-ASCII characters, delete it and create a new one. (Since the ID is an opaque string, there’s no benefit in trying to create a downgraded ID that looks like the original one.) Message-IDs also appear in `References:` and `In-Reply-To:`

[10] Mail standards originally limited empty groups to specific headers, but RFC 6854 relaxed the rules to allow them anywhere an address can appear.

headers. If those headers contain non-ASCII IDs, delete the ID, and if there are no other IDs left in the header, delete the header.

If any other headers not shown to users have UTF-8 characters, delete them, since there’s no way to downgrade them and they won’t affect what the user sees.

If a MIME Content-Type header, either in the message headers or in the header of a MIME body part, contains a field with non-ASCII text, delete that field. All required fields in these headers are ASCII. If the header has no fields left, delete the header.

Downgrading in POP and IMAP

If a POP or IMAP mailbox contains EAI messages but the MUA client does not support EAI, the POP or IMAP server has the option of downgrading the EAI messages so the user can see at least an approximation of what was sent. The downgrade process is the same as that described above, except that POP and IMAP servers are unlikely to have substitute addresses. This is addressed in more detail in RFC 6858.

Downgrade notes

If a downgraded message has a DKIM signature, the downgrade will invalidate the signature, so delete the signature. In some cases it's possible to re-sign. See the next section.

Rather than trying every EAI message sent to a non-EAI address to see if it works, when a message fails, if the recipient address is in an address book, tag the entry as legacy only. (Also provide some way to remove the tag when the recipient system upgrades.)

Out-of-office, read receipts, and other autoresponders

Mail systems often allow automatic responses for out-of-office and similar notifications.

From an EAI point of view, autoresponder mail is like any other mail. If the recipient of the response (that is, the sender of the original message) has an EAI address, or is otherwise known to accept EAI mail, the autoresponse can be an EAI message. Otherwise it should be a legacy message, either created as a legacy message or downgraded from an EAI message.

Mail authentication and spam filters

As spam becomes an ever-worsening problem, mail systems have added a wide variety of anti-spam features. Among the most effective of those is authentication, showing that the message was in fact sent by its purported sender.

For details on EAI and mail authentication beyond the discussion below, see the Internet Draft https://datatracker.ietf.org/doc/draft-levine-appsarea-eaiauth/.

SPF authentication

SPF (Sender Policy Framework) checks the IP address from which a message was sent against a list of authorized IPs for a domain. The domain checked is the domain in the `MAIL FROM:` bounce address, or if that address is empty, in the `EHLO/HELO` argument.

A domain publishes its authorized addresses in a TXT record with a fairly complex syntax. The SPF record for icann.org is typical:

```
icann.org IN TXT “v=spf1 ip4:192.0.32.0/20 ip4:199.91.192.0/21
 ip4:64.78.40.0/27 ip4:162.216.194.0/27 ip6:2620:0:2d0::0/48
 ip6:2620:0:2830::0/48 ip6:2620:0:2ed0::0/48
 include:salesforce.icann.org -all”
```

This lists several IPv4 and IPv6 address blocks, and includes all the blocks published at salesforce.icann.org.

Using SPF in EAI mail is straightforward. Since the information is published in the DNS, a validator performs the usual IDNA domain lookup, turning U-labels into A-labels and then doing a DNS query. The SPF records themselves don't contain any UTF-8 so the processing to see whether an IP address matches an SPF record is unchanged.

DKIM authentication

Sending and relaying MTAs use DKIM to add cryptographic signatures to mail messages. A receiving system can check the signature. If it's valid, it knows that the message it checked is the same one the signer signed. The signature is placed in a DKIM-Signature header, such as this one on a message from Microsoft:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=microsoft.com;
 s=rm1; h=From:Date:Subject:Message-ID:Content-Type:MIME-Version;
 i=billg@microsoft.com; bh=SwUgUaGo9gVX7cc6F206JZvUfMea/5YYkAej9uuVF6Q=;
 b=EdBiodGIf4bCmbGEZBQFUa+MCfoPhm138ox5x6iq4FCmITu522v1Q/9t/A+E=
```

The signature is a list of `tag=value` items. The tag is the domain that applied the signature, and the `s= tag` is a "selector" used to find the validation key in the DNS. The optional `i= tag` is an identity meaningful to the signer.[11] The `bh= tag` is a hash of the message's body, and the `b= tag` is a signed hash of selected message headers along with the DKIM-Signature itself. A receiving system checks the hashes against the received message and the validation key in the DNS to see if the signature is valid.

The validation key is stored in the DNS at `<`*`selector`*`>._domainkey.<`*`domain`*`>`, so in this case, since the `s= selector` is `rm1` and the `d= domain` is `microsoft.com`, it would be (omitting some uninteresting details):

```
rm1._domainkey.microsoft.com. TXT "v=DKIM1; k=rsa;
 p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCkHq3ztGIm1R8alD+7oZiaG5mT;"
```

EAI makes some small changes to DKIM signatures. The `s= selector` and the domains in the `d=` and `i=` tags can be U-labels, and the `i=` string can also be UTF-8.[12] This should have no effect when creating DKIM signatures, since the characters in the domains are handled like any other characters in the message. When verifying a DKIM signature, any U-labels in the `d=` domain or `s=` selector must be converted to A-labels for the DNS lookup.

If a system downgrades a message, the changes from downgrading will change the message enough that existing DKIM signatures become invalid. If the downgrading happens in an MUA, the message will generally be re-signed as it passes through the MSA. If a message is downgraded in an MSA or MTA, the message should be re-signed if the MSA or MTA has the signing key available.

[11] Although the i= tag has the same syntax as an email address, it need not be the address of the person who sent the message, or of anyone else.

[12] RFC 6376 says that the domains in signatures should all be A-labels, but this is likely to change.

DMARC authentication

DMARC is an anti-phishing scheme that uses the authentication results of SPF and DKIM to validate the domain in the `From:` header. If a domain publishes a DMARC policy in the DNS, receivers check to see if the message has a valid SPF result or DKIM signature for that domain. If so, the message passes. This doesn't mean it's not spam, but it does mean it really is from the purported sender.

For example, assume the `From:` header says:

```
From: Bob Smith <bob@example.com>
```

The DMARC validator checks to see if the SPF check is from example.com (that is, the `MAIL FROM:` address was at example.com) and succeeded, or that the message has a valid DKIM signature with `d=example.com`. If either is so, the DMARC check succeeds.

Once an EAI system has implemented SPF and DKIM, there is little additional work for DMARC. When checking the domain in the `From:` header, if the domain is an IDN, either the A-label or U-label version of the IDN can match the SPF domain or the DKIM `d= domain`.

Spam filtering

EAI makes little difference to spam filtering techniques. Since everything in the message other than the `To:/From:/Cc:`[13] header addresses could already be UTF-8, filters already deal with internationalized text. If a filtering system provides whitelisted and blacklisted sending addresses, those lists should handle EAI addresses, and when checking against incoming mail, should handle both U-label and A-label versions of the domains.

One possible source of trouble is empty address groups in downgraded messages. Groups have been allowed in `To:` and `Cc:` headers since the 1970s, but have only been allowed in `From:` headers since RFC 6854 in 2013. There are still some mail systems that consider `From:` groups to be invalid, in some cases displaying them wrong and in a few cases treating the whole message as spam.

[13] Incoming messages do not normally have Bcc headers because they are supposed to be stripped out as a message is sent.

Appendix A: Handling message header fields

In this table “Substitute or delete EAI address” means to replace the address with a legacy address if possible, otherwise delete the address as described in “Downgrading headers with addresses” above. “Allow UTF-8 text” means literal text in preference to MIME encoded-words.

Header Field Name	EAI changes	Downgrade actions
Authentication-Results	Allow IDN domains where domains appear	Delete if contain UTF-8
Bcc	Allow EAI addresses	Substitute or delete EAI addresses, delete header if no addresses remain
Cc	Allow EAI addresses	Substitute or delete EAI addresses
Comments	Allow UTF-8 text	Use MIME encoded-words for UTF-8 text
Content-Description	Allow UTF-8 text	Use MIME encoded-words for UTF-8 text
Content-ID	None	Delete if contains UTF-8
Content-Transfer-Encoding	None	None

Content-Type	Usually none, possible UTF-8 in optional filename tags	Delete any tags with UTF-8 values
Date	None	None
DKIM-Signature	Allow IDNs in `d=` `i=` `s=` tags, allow UTF-8 in `i=` local part	Delete and re-sign
Errors-To	Possibly allow EAI addresses	Substitute EAI addresses or delete header
From	Allow EAI addresses	Substitute or delete EAI addresses
In-Reply-To	None	Delete UTF-8 message-IDs, delete header if no message-IDs remain.
Keywords	Allow UTF-8 text	Use MIME encoded-words for UTF-8 text
List-Archive	Mailto: URI can contain EAI addresses	Substitute or delete EAI addresses, delete header field if no URIs remain
List-Help	Mailto: URI can contain EAI addresses	Substitute or delete EAI addresses, delete header field if no URIs remain
List-ID	None. (UTF-8 text not recommended.)	Delete if contains UTF-8 text

List-Owner	Mailto: URI can contain EAI addresses	Substitute or delete EAI addresses, delete header if no URIs remain
List-Post	Mailto: URI can contain EAI addresses	Substitute or delete EAI addresses, delete header if no URIs remain
List-Subscribe	Mailto: URI can contain EAI addresses	Substitute or delete EAI addresses, delete header if no URIs remain
List-Unsubscribe	Mailto: URI can contain EAI addresses	Substitute or delete EAI addresses, delete header if no URIs remain
Message-ID	None	Should be none, delete and replace any UTF-8 message-ID header
MIME-Version	None	None
Organization	Allow UTF-8 text	Use MIME encoded-words for UTF-8 text
Received	Allow IDN domains and EAI mailboxes	Delete if contains UTF-8 text
References	None	Delete UTF-8 message-IDs, delete header if no message-IDs remain.
Reply-To	Allow EAI addresses	Substitute or delete EAI addresses, delete header field if no addresses remain

Resent-Bcc	Allow EAI addresses	Substitute or delete EAI addresses, delete header field if no addresses remain
Resent-Cc	Allow EAI addresses	Substitute or delete EAI addresses
Resent-Date	None	None
Resent-From	Allow EAI addresses	Substitute or delete EAI addresses
Resent-Message-ID	None	Should be none, delete and replace any UTF-8 ID
Resent-Sender	Allow EAI addresses	Substitute EAI address or delete header field
Resent-To	Allow EAI addresses	Substitute or delete EAI addresses
Return-Path	Allow EAI address	Delete if contains UTF-8
Sender	Allow EAI addresses	Substitute EAI address or delete header field
Subject	Allow UTF-8 text	Use MIME encoded-words for UTF-8 text

To	Allow EAI addresses	Substitute or delete EAI addresses

Appendix B: Summary of steps necessary to move to EAI

Here is a summary of software changes to add EAI capabilities to various kinds of mail software.

Changes to MTAs

- Distinguish EAI from legacy messages, either by external tag or by scanning for UTF-8 in headers
- As a server, advertise `SMTPUTF8 EHLO` keyword, accept `SMTPUTF8` keyword on `MAIL FROM:`, put `UTF8SMTP` in `Received:` header
- As a client, look for `SMTPUTF8 EHLO` keyword, send `SMTPUTF8` keyword on `MAIL FROM:`
- If sending an EAI message fails, downgrade the message and resend, but only if MTA has necessary address book info and DKIM signing keys

Changes to MSAs

- Advertise `SMTPUTF8 EHLO` keyword
- Distinguish EAI from legacy messages by `MAIL FROM:` keyword
- If sending an EAI message fails, downgrade the message and resend, but only if MSA has necessary address book info and DKIM signing keys

Changes to MUAs

- Allow UTF-8 in addresses in messages
- Allow UTF-8 in addresses in address books, allow entries to have both EAI and legacy addresses
- Look for `SMTPUTF8 EHLO` keyword from MSA, add `SMTPUTF8` keyword on `MAIL FROM:`, don't send EAI messages to MSAs that can't handle them
- If sending an EAI message fails, downgrade the message and resend, but only if MUA has necessary address book info

IMAP Servers

- Support UTF-8 folder names
- Tag messages as EAI or legacy, either by tag set when message created or by scanning headers
- Add `UTF-8=ACCEPT` IMAP feature
- (Optional) Downgrade EAI messages as needed for legacy clients

POP Servers

- Tag messages as EAI or legacy, either by tag set when message created or by scanning headers
- Add UTF8 capability and `UTF8:` command
- (Optional) Downgrade EAI messages as needed for legacy clients

Appendix C: How does SMTP work?

SMTP uses a series of commands and responses to transfer a message from one mail server to another. The commands and responses are all ASCII characters (other than EAI addresses.) Responses start with a three-digit number which provides the meaning of the response, followed by text, which is generally a comment.

In the example below, user sam@sender.org is sending mail to ray@receive.net. To keep things simple, the sending host is sender.org and the recipient host is receive.net.

First, the sending computer connects, and the receiving computer sends an initial response. The sending computer sends `EHLO` (extended hello) to identify itself and ask for a list of features that the receiving computer supports:

```
S: <connect>
R: 220 receive.net ESMTP
S: EHLO sender.org
R: 250-8BITMIME
R: 250 PIPELINING
```

Code numbers of the form 2xx indicate success. In this example, the recipient host supports two features, `8BITMIME` and `PIPELINING`. Nearly all hosts support these. (`8BITMIME` allows non-ASCII text in message bodies, `PIPELINING` allows the sender to send multiple commands without waiting for a receipt for each one.)

The *sending* computer sends a `MAIL FROM:` command, identifying the sender address to which status and non-delivery reports can be sent, and optionally also other envelope information.

```
S: MAIL FROM:<sam@sender.org>
R: 250 Sender accepted
```

The *sending* computer sends `RCPT TO:` commands, each with a recipient address to which the message is sent.

```
S:RCPT TO:<ray@receive.net>
R:250 Recipient accepted
```

(If there were more than one recipient, there would be a separate `RCPT TO:` command and response for each one.)

The *sending* computer sends a `DATA:` command, and then, when the receiving computer responds, it sends the message as a single block of text, the header followed by the entire message including header and body, followed by a line with a single dot which indicates the end of the message.

```
S:DATA
R:354 Send your message
S:From: sam@sender.org
S:To: ray@receive.net
S:Subject: lunch
S:
S:How about lunch at 12:30?
S:.
R:250 Message accepted 409343fg34
```

At this point the message is complete, so the sender uses QUIT to end the session.

```
S:QUIT
R:221 Sayonara
```

Adding UTF-8 features to the SMTP example

EAI makes small changes to the protocol. The server's response to `EHLO` contains a `UTF8SMTP` keyword to indicate that the server supports EAI:

```
S: <connect>
R: 220 receive.net ESMTP
S: EHLO sender.org
R: 250-8BITMIME
R: 250-UTF8SMTP
R: 250 PIPELINING
```

The client's `MAIL FROM:` command includes a `SMTPUTF8` keyword to indicate that a message is an EAI message.

```
S: MAIL FROM:<猫王@普遍接受-测试.世界> SMTPUTF8
R: 250 Sender accepted
```

The rest of the SMTP session is unchanged, other than allowing UTF-8 data in mail headers:

```
S:RCPT TO:<ray@receive.net>
R:250 Recipient accepted
S:DATA
R:354 Send your message
S:From: 猫王 <猫王@普遍接受-测试.世界>
S:To: ray@receive.net
S:Subject: 我们要吃午饭吗?
S:
S:How about lunch at 12:30?
S:.
R:250 Message accepted 389dck343fg34
S:QUIT
R:221 Sayonara
```

Appendix D: Relevant RFCs

Most of the specifications for Internet services are developed and maintained by the Internet Engineering Task Force (IETF), a group affiliated with the Internet Society. The IETF's major publication series is known for historical reasons as Requests for Comments or RFCs. Each RFC is numbered, starting with RFC 1 in 1969 and currently up to about RFC 8300. Some RFCs are standards, while others describe best current practices or report other information relevant to the technical operation of the Internet. The IETF sometimes issues new standards RFCs that update or replace prior standards. When they do, they generally avoid changes that are incompatible with existing standards.

All RFCs are available to the public at no charge at the RFC Editor's web site at https://www.rfc-editor.org/, and at many other places around the Internet.

Important mail RFCs

Here are some of the more important RFCs. For a longer list see UASG 006,

For SMTP:

- RFC 5321 - Simple Mail Transfer protocol, from one MTA to another (previously RFCs 821 and 2821)
- RFC 6609 - Mail submission, from MUA to MSA (previously RFCs 2476 and 4409)

For mail message formats:

- RFC 5322 - Internet Message Format (previously RFCs 822 and 2822)
- RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
- RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
- RFC 2047 - MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text
- RFC 2048 - Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures
- RFC 2049 - Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples
- RFC 3464 - An Extensible Message Format for Delivery Status Notifications
- RFC 6854 - Update to Internet Message Format to Allow Group Syntax in the "From:" and "Sender:" Header Fields

For POP and IMAP:

- RFC 1939 - Post Office Protocol - Version 3 (POP3)
- RFC 3501 - Internet Message Access Protocol - Version 4rev1 (IMAP4)

For IDNs:

- RFC 3629: UTF-8, a transformation format of ISO 10646
- RFCs 5890, 5891, 5892, 5893, 5894, and 5895 - Internationalized Domain Names for Applications (IDNA)

For EAI mail:

- RFC 6530 - Overview and Framework for Internationalized Email
- RFC 6531 - SMTP Extension for Internationalized Email
- RFC 6532 - Internationalized Email Headers
- RFC 6533 - Internationalized Delivery Status and Disposition Notifications
- RFC 6855 - IMAP Support for UTF-8
- RFC 6856 - Post Office Protocol Version 3 (POP3) Support for UTF-8
- RFC 6857 - Post-Delivery Message Downgrading for Internationalized Email Messages
- RFC 6858 - Simplified POP and IMAP Downgrading for Internationalized Email

For PRECIS and string mapping:

- RFC 8264 - PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols
- RFC 8265 - Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords
- RFC 8266 - Preparation, Enforcement, and Comparison of Internationalized Strings Representing Nicknames
- RFC 7790 - Mapping Characters for Classes of the Preparation, Enforcement, and Comparison of Internationalized Strings (PRECIS)
- RFC 6885 - Stringprep Revision and Problem Statement for the Preparation and Comparison of Internationalized Strings (PRECIS)
- RFC 6943 - Issues in Identifier Comparison for Security Purposes
- UTS#39 - Unicode Security Mechanisms, sections 3.3 Email Security Profiles for Identifiers and 5.2 Restriction Level Detection

For SASL (logging in and authenticating for POP, IMAP, and submission):

- RFC 4422 - Simple Authentication and Security Layer (SASL)
- RFC 4616 - The PLAIN Simple Authentication and Security Layer (SASL) Mechanism
- The LOGIN SASL Mechanism Internet draft
- IANA SASL methods registry

For URLs and URIs:

- RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax.
- RFC 3897 - Internationalized Resource Identifiers (IRIs)

For message authentication:

- RFC 6376 - DomainKeys Identified Mail (DKIM) Signatures
- RFC 7208 - Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1
- RFC 7489 - Domain-based Message Authentication, Reporting, and Conformance (DMARC)
- Eaiauth draft - Email Authentication for Internationalized Mail

For input and display of RTL and mixed direction text:

- RFC 5893 - Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)

Appendix E: Other sources of advice

- UASG documents at https://uasg.tech/documents/
- For mail format and specification standards, see email RFCs; see the list above
- For HTML and other web data standards, see the W3C website at https://www.w3.org/standards/
- Stack Overflow for general mail programming, https://www.stackoverflow.com
- Discussion lists for mail software. Most lists have online archives; check the archives to see if your question has already been asked and answered.
    - Postfix http://www.postfix.org/lists.html
    - Qmail http://www.qmail.org/top.html#patches and https://lists.gt.net/qmail/users/
    - Sendmail https://groups.google.com/forum/#!forum/comp.mail.sendmail
    - Cyrus IMAP https://cyrusimap.org/imap/support/feedback-mailing-lists.html
    - Dovecot https://dovecot.org/mailinglists.html
- IETF working group discussion lists. These are useful for questions about RFCs and details of how to implement them, but not for general mail or programming questions.
    - DKIM mailing list, ietf-dkim@ietf.org
    - DKIM operations list, dkim-ops@mipassoc.org
    - DMARC mailing list, dmarc-discuss@dmarc.org
    - SMTP mailing list, ietf-smtp@ietf.org

Glossary

A-label: An ASCII encoded version of a UTF-8 domain label, in the form `xn--`*`stuff`* where the *stuff* is a punycode version of the UTF-8. Compare to U-label.

Address book: A list of email addresses and related contact information.

ASCII Compatible Encoding (ACE): The format used in an A-label, in the form `xn--`*`stuff`* where the *stuff* is a punycode version of the UTF-8.

Authentication: Verifying the source of, or other information about, the origin of a message.

Bidi rule: The rule that prohibits combining left-to-right and right-to-left text in a single domain name label. Defined in RFC 5893, section 2.

Body: The contents of a message, which follows the headers. The body may be unformatted text, or it may be one or more formatted or encoded MIME parts.

Encoded words: A MIME format that represents UTF-8 text (other than addresses) as ASCII in message header lines. Looks like `=?utf-8?Q?`*`stuff`*`?=`.

Envelope: Information that accompanies a message in transit, including the address(es) it is being sent to, and the return address to which error or failure reports can be sent.

Field names: The names of mail header fields such as `From:`, `To:`, and `Subject:`.

Hostname: a DNS record that identifies a computer on the Internet. The characters in hostnames are restricted to those allowed by the LDH rule

IDNA: Internationalized Domain Names in Applications. The rules for characters and their combinations that can appear in UTF-8 domain names. The current version is IDNA2008, defined in RFCs 5891-5895.

IMAP: Internet Message Access Protocol, a standard for managing mail messages and folders on remote servers. The current version is IMAP4. Compare to POP.

LDH rule: The rule that DNS host names can only contain Letters, Digits, and Hyphens. Further rules forbid leading or trailing hyphens, and names with hyphens in the third and fourth positions.

Legacy address: An email address that consists of only ASCII characters. Compare to an internationalized address.

Legacy mail: Messages with only legacy addresses. Compare to EAI mail.

Linkify and linkification: Automatically formatting text to add clickable links to HTML URLs in the text.

Mail delivery agent (MDA): A server program that handles incoming mail and typically stores it in a mailbox or folder.

Mail folder: A collection of messages with a name. A mailbox can contain multiple folders.

Mail header: The collection of structured fields at the beginning of a mail message preceding the mail body. Each field starts with a field name and a colon, and consists of one or more lines.

Mail submission agent (MSA): A server program that receives mail from a MUA and prepares it for transmission and delivery.

Mail transmission agent (MTA): A server program that sends and receives mail to and from other Internet hosts. An MTA may receive mail from an MSA and/or deliver mail to an MDA.

Mail user agent (MUA): A client program that a person uses to send, receive, and manage mail. Popular MUAs include Outlook and Thunderbird.

Mailbox: A place to store messages associated with a mail address. A mailbox may be a single folder or a collection of folders.

Metadata: Information about a mail message, distinct from the message itself. Metadata may include the time the message was sent or received and whether the recipient has opened it.

MIME: Multipurpose Internet Mail Extensions, a specification for encoding different kinds of data including non-ASCII text, and for including multiple logical parts in a single message.

MX: Mail Exchanger, a DNS record that identifies a host that receives mail for a domain.

Normalization: Transforming text into a standard representation, e.g., combining separate character and accent codepoints into precomposed codepoints.

Normalization Form C (NFC): A profile of Unicode in which characters are precomposed.

POP: Post Office Protocol, a standard for retrieving messages from a remote server. The current version is POP3. Compare to IMAP.

Precomposed: A character that is represented as a single codepoint rather than as a separate base character and accents and other modifiers.

PRECIS: Preparation and Comparison of Internationalized Strings, a set of IETF documents that set out principles for creating UTF-8 usernames and nicknames.

Procmail: An MDA popular on Unix and Linux systems. Wikipedia article.

Punycode: An ASCII encoding of UTF-8 used in ACE A-labels (RFC 3492).

SASL: Simple Authentication and Security Layer, a set of IETF standards that define the username and password authentication schemes used to log into SUBMIT, IMAP, and POP servers.

Script: A collection of codepoints used to write a language. Some languages such as English are written in a single script, others such as Japanese are written in multiple scripts.

SIEVE: A language for filtering and sorting incoming mail messages (RFC 5228).

SMTP: Simple Mail Transport Protocol, the way computers on the Internet exchange mail

String preparation: Turning a UTF-8 string into a standard form to make it easier to process.

SUBMIT or submission: transmitting a newly created message for relay and/or delivery (RFC 6409)

U-label: A UTF-8 domain label. Compare to A-label.

Variant characters: Characters in a script with the same meaning but different visual representations.