Network Working Group
Internet-Draft
Expires: January 19, 2006

J. Klensin

July 18, 2005

Internationalization of Email Addresses
draft-klensin-emailaddr-i18n-03.txt

Status of this Memo

Copyright Notice

Abstract

Internationalization of electronic mail addresses is, if anything, more important than the already-completed effort for domain names. In most of the contexts in which they are used, domain names can be hidden within or as part of various types of references. Email addresses, by contrast, are crucial: use of names of people or organizations as, or as part of, the email local part is, for obvious reasons, a well-established tradition on the network. Preventing people from spelling their names correctly is, in the long term, inexcusable. At the same time, email addresses pose a number of

special problems -- they are more difficult than simple domain names in some respects, but actually easier in others. This document discusses the issues with internationalization of email addresses, explains why some obvious approaches are incompatible with the definitions and use of Internet mail, and proposes a solution -- for both addressing and email internationalization more generally -- that is likely to serve users and the network well for the long term.

Table of Contents

1. Introduction

Internationalization of electronic mail addresses is, if anything, more important than the already-completed effort for domain names. In most of the contexts in which they are used, domain names can be hidden within, or as part of, various types of references or the references themselves may be hidden. It also remains controversial whether internationalization of domain names is actually necessary, no matter how attractive and important it may appear at first glance. Email addresses, by contrast, are crucial: use of names of people or organizations as, or as part of, the email local part is, for obvious reasons, a well-established tradition on the network. While the characters permitted in domain name strings have always been somewhat constrained so that they are not confused with syntax requirements of present and future applications, preventing people from spelling their names correctly is, in the long term, inexcusable. However, while it is tempting to ignore them, email addresses pose a number of special problems. Unlike domain names --and, consequently, the domain part of an email address (after the last "@")-- the local part (or mailbox name) is essentially unconstrained with regard to syntax or the characters used. There are no special delimiters comparable to the period used to separate domain name labels, there is no standardized structure comparable to the domain name system's hierarchy, and it has always been a firm protocol requirement that no host other than the one to which final delivery is made is permitted to parse or interpret the address (see section 2.3.10 of [RFC2821]). In some respects, this makes things much more difficult: it is far more difficult to know what behavior will cause existing systems to cease working properly. In others, it actually makes them easier, since the originating system is not required to understand how the receiving one will interpret an address and indeed must not do so.

The balance of this document explores these issues in more detail.

1.1 Terminology

While much of the description here depends on the abstractions of "Mail Transfer Agent" ("MTA") and "Mail User Agent" ("MUA"), it is important to understand that those terms and the underlying concepts postdate the design of the Internet's email architecture and the "protocols on the wire" principle. That architecture, as it has evolved, and the "wire" principle have prevented any strong and standardized distinctions about how MTAs and MUAs interact on a given origin or destination host (or even whether they are separate).

This document assumes a reasonable understanding of the protocols and terminology of the most recent core email standards documented in RFC 2821 [RFC2821] and RFC 2822 [RFC2822].

In its present internet-draft form, the document contains a great deal of explanatory material and rationale for the approach chosen. The actual protocol material appears almost entirely in Section 4, especially Section 4.2 through Section 4.4, and in Section 5. If it appears to be a candidate for standards-track publication, the explanatory material, rationale, and most of the other background materials should be removed to a separate document. Those who wish to bypass the reasoning and comparison to other alternatives in this document and examine the protocol proposal should skip to those sections.

In this document, an address is "all-ASCII" if every character in the address is in the ASCII character repertoire [ASCII]; an address is "non-ASCII" if any character is not in the ASCII character repertoire.

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Three Models for Transition

Almost every attempt to extend the Internet mail system to support new formats leads quickly to a controversy about how to implement such changes and fit them into the existing system. The proposals tend to fall into three categories:

2.1 No Infrastructure Changes

Avoid any more infrastructure changes than are absolutely necessary. Instead, use sometimes-elaborate coding and other "tricks" to embed the new facilities in the old ones, even if those facilities will look very odd to those whose client and user interface systems have not been upgraded. MIME, which has been quite successful, is the most-cited example of this approach, although it is worth remembering that it was initially quite unpopular because it exposed many users to rather opaque codings. It continues to be criticized as institutionalizing incompatibility rather than providing good interoperability. Specificially on the latter observation, MIME conformance does nothing to prevent delivery of a message to a user that the user has no capability of decoding and reading.

2.2 Transport-level Negotiation

Use a transport-level negotiation model of some variety to ensure that the recipient machine can and will accept the format and structure of the message and options being sent. Variations on this approach include ensuring that the message can be delivered, but not

that it can be read. This approach has been taken in situations in which, e.g., sending the message without such acceptance could result in actual information loss (as distinct from "mere" information inaccessibility for the MIME case). ESMTP ([RFC1651], [RFC2821]) provides the framework for this approach; options such as 8BITMIME [RFC1652] are clear examples of cases in which the approach is necessary.

2.3 Replace SMTP and the Current Internet Mail Environment

Start a conversation about discarding more or less everything and moving toward a "next generation" of Internet mail in the hope of a huge gain in elegance, capability, or other functions.

2.4 Analysis

Of these three, only the second appears to be plausible for internationalization of email local parts.

The third --a new mail system, replacing SMTP and possibly the mail header and MIME models-- is the most easily discarded as a possibility. Despite many brief bursts of enthusiasm, development of standards by other organizations, and well-funded proprietary products, proposed SMTP replacements have tended to not go anywhere: standard Internet mail, with or without extensions, is sufficiently well-established (and entrenched) as an interoperable interchange standard that historical proposed "next generation" alternatives have been forced, by the marketplace, to interoperate with and, ultimately, to yield to, it. Those that did not make a serious attempt to interoperate with Internet mail have largely disappeared. There is no reason to believe that a new set of proposals will fare any better.

In considering the other two approaches, it is important to note that none of the fundamental transitions have ever been, or will ever be, easy, quick, and without side-effects. While it is easy, safe, and fairly painless to add a new media type to MIME, the MIME framework itself provided a very unpleasant user experience until mail user agents (MUAs) and related software were upgraded. Similarly, while most ESMTP extensions can be added at a relatively low level of pain, the infrastructure upgrades needed to accommodate the extension framework itself (and hence any of the extensions) were significant, especially in the case of MTA software that had been operating smoothly, without maintenance or upgrades, for years.

For internationalization, the important question is whether the transition properties are "more like MIME" (i.e., can be accomplished at an MUA-MUA level, without the transport system getting involved)

or whether the changes required are significant enough to require transport negotiation (or a "next generation mail" environment). It is the position of this document that the complex of changes needed to make Internet email fully internationalized requires, if not "next generation mail", at least the type of position that characterized the original, "framework", MIME and ESMTP deployments: "if you want any of these features, you are going to implement and accept a package of changes; if that is not feasible, you need to stay with ASCII, or ASCII-encoded, addresses and headers until you are ready to upgrade". The alternatives are just too complex, and therefore problem-prone, in terms of combinations of alternate forms, options, transitions, upgrading and downgrading, and so on to preserve a high level of interoperability.

The sections that follow discuss the motivation and implications of this conclusion in more detail before moving on to a specific proposal.

3. History, Context, and Design Constraints

Several key issues in how email works and is handled impose significant constraints on the solution space. Email is often used as a transport mechanism for information that will be acted on by computers, not merely read by people. While the approach is not common, some of the systems that use it as a computer-computer communication medium encode routing, processing, or validation information into the envelope address fields. More commonly, recipient systems use special address formats to encode local routing or priority information. In recent years, some of these addressing techniques have become important anti-spam tools for some users and communities. Most of these techniques have a long history. Most or all of them conform to email standards and practices that, in turn, go back to the first uses of email on the ARPANet. Backward-compatibility --not damaging the interoperability of standards-conforming programs that are now deployed and working correctly-- makes it inappropriate to make decisions by conducting user surveys and concluding that "not too many" people will be hurt. Any new system must preserve existing practices and flexibilities unless there are overwhelming reasons -- e.g., an absence of plausible alternatives -- to not do so.

Historically, when one of these approaches has required that the email address local part be partitioned into components that are then interpreted differently or in some special sequence, the information has been organized according to some lexical convention, typically either based on one or more delimiters or on some sort of position and length notation (or a mixture of the two for different purposes). Either may be applied left-to-right or right-to-left and, again, we

have a history of each, including the notorious "!a!b!c!d%e%f" local parts, which can be interpreted as

- A single mailbox name, "!a!b!c!d%e%f"
- A routing instruction to send the address "!b!c!d%e%f" to host "a"
- A routing instruction to send the address "!a!b!c!d%e" to host "f"

or some combinations of those interpretations.

Because the correct interpretation can only be known at the destination host, attempts by intermediate hosts, or even the originating user, to interpret the structure of the address string cause serious problems with mail reliability. Worse, because the organization of the system(s) that make up the destination host cannot, in general, be known to the sender, approaches that assuming decoding from some coded form at some particular order in the process of receiving and delivering the message will cause some fraction of systems that are now fully conformant to Internet mail standards to fail to properly handle the address.

3.1 The Presentation Issue

Before continuing, it is important to note that any internationalization system, regardless of how it is implemented at the protocol level, will require changes at the user interface level if it is to function in a way that end users consider reasonable. Unless addresses are presented to the user in familiar characters and formats, the user's perception will be, not of internationalization and behavior that is user and culturally friendly, but of a relatively hostile environment. One thing we have almost certainly learned from nearly forty years of experience with email is that users strongly prefer email addresses that closely resemble names or initials to those involving, e.g., user ID numbers or complex coding that makes the local part appear as gibberish. Indeed, that principle --of wanting local parts to appear intelligible-- is arguably the entire reason for wanting to internationalize these addresses. Otherwise, any identifier would suffice whether it had mnemonic value to users or not. If a user sees "xn--fltstrm-5wa1o" (a punycode form) or "F=E4ltstr=F6m" (the MIME quoted-printable form), rather than the correctly-written localized string, the result is almost certain to be unhappiness.

3.2 MUAs, MTAs, addresses, and learning from MIME and ESMTP

The development and deployment of MIME [RFC2045] provided a number of important lessons for the community about how to design extensions and enhanced features without harm to the installed and conforming email system. Perhaps the most important of these was that it is

easier, and often more expedient, to make changes that have impact only on mail user agents. If it is possible to make changes that way --generally changes that involve only message headers and the message body or body parts-- users who need particular features can switch to user agents that support them or press for those features in the user agents they have already selected. Even in the worst case in which support for features the user considers critical is not readily available, it is possible, with proper user agent design, to save the entire message to a file and then use stand-alone software to interpret the information and perform the desired functions.

Providing these functions in the message headers and body permits them to be moved opaquely through the mail transport system, thus avoiding any requirement to modify originating or delivery MTAs or intermediate relays. In practice, the user may have little control over those systems. Since changes to them typically impacts large numbers of users, those who are responsible for them are often reluctant to make changes in response to the needs of a few users.

It is hence reasonable to conclude that, if it is feasible to support address internationalization strictly at the MUA level, keeping the internationalized addresses opaque to the transport system, that is a more desirable approach than requiring MTA changes. The MUA-only approach has been carefully examined by others (see, e.g., the obsolete Internet Draft [Hoffman-IMAA]) and proposed more recently as a temporary measure in [JET-IMA].

The present document argues that

1. Addressing is a fundamental MTA-level function,
2. Some of the complexities encountered when trying to encode addresses so as to avoid MTA interactions are symptoms that attempting to "hide" the MTA function so that it can be handled by MUAs is not an architecturally desirable approach,
3. The restrictions on email uses and syntax required to provide internationalization at MUA level are unnecessarily risky, and almost certainly damaging, to deployed email infrastructure,
4. If internationalization is to be plausible, it is critical that addressing information be represented in essentially the same way in the message envelope (i.e., the SMTP command structure) and the message body (i.e., both message headers and, where feasible, message text). Different encodings in different places, especially ones that are copied back and forth, will make both mail system maintainers and operators and end users very unhappy. and
5. MTA-level solutions are feasible, architecturally more elegant, and perhaps not as difficult to deploy in relevant communities as the strongest advocates of the MUA-only approach appear to

imagine. See Section 3.8 for additional discussion on this point.

The decision as to what to do in message bodies and formats (e.g., [RFC2822] and MIME [RFC2045]) and what to handle in message transport (i.e., in extended SMTP) is critical because, as discussed below, the level at which something is handled is both determined by, and determines, how information is appropriately encoded. This decision ultimately depends on the application of two principles:

1. If body content is opaque, anything still visible to transport requires transport negotiation.
2. Anything an MTA -- be it origin, relay, MX, gateway, or delivery -- needs to understand or process must be handled as part of mail transport. The discussion below might be titled "why the MTA must get involved".

Whether mail addresses meet these criteria, and hence must be comprehensible in transport, depends on how much the sending MUA needs to know to construct, and the delivery MTA needs to know to deliver, a message. Traditionally, we have kept the former knowledge level at zero: if a sender produces "!a!b!c@example.com" in response to information that it is a valid address, it still does not know whether this is a "bang path" or a slightly-perverse name for a single mailbox. Is "xyz%def@example.com" a specification for routing to mailbox "xyz" on host "def" or a mailbox named "xyz%def" on the example.com host? Are "foo+bar@..." or "foo-baz@..." subaddresses "bar" and "baz" for the mailbox "foo", or are they simple addresses? Is "jjoneschem@labs.example.com" a local mailbox on that host or an instruction to route mail to "jjones" in the chemistry department?

Under the rules established in [RFC0821] and [RFC1123], as summarized and updated in [RFC2821], all of those decisions are up to "example.com", its MX alternatives, or hosts in that domain, and they may make very local decisions about them. For example, even within the same domain (on the same apparent host), "xyz%def" might be a mailbox while "xyz%ghi" might contain a route; "foo-baz" might represent and address and subaddress while "foo-blog" might be a mailbox.

The sender cannot, in the general case, know.

Worse, while non-alphanumeric characters like "+", "-", and "%" have been used in these examples, delimiters for subaddresses, implicit routing, embedded commands, and so on are, again, up to the destination MTA and its interpretations. "X" might be as good a delimiter as "+". It might even be a better one in some applications. And, since local-parts are defined as case-sensitive,

"x" might be a normal address character in the same address in which "X" was an important delimiter.

Of course, in a completely non-ASCII environment, it would make sense to substitute characters from the local script for "+", "-", "%", and so on. If one wants a string completely in local language (i.e., non-ASCII) characters, then there may be no desire to break that convention in order to use an ASCII delimiter (see Section 8 for additional discussion of this issue).

It is not even necessary to use a delimiter to support some forms of subaddressing or local routing. Suppose an organization adopted the convention that externally-visible email address local parts were structured as, e.g., a three-letter department code, followed by a five-letter code representing the individual, optionally followed by a code representing a project. Many organizations use just such systems and there is no way (and no need) for an email sender to understand the system or whether it is actually used for mail routing internally.

Consequently, the idea of a sender breaking an address up into its component parts and encoding those parts separately, or even just doing an encoding in sections that preserves the positions of the delimiters (as measured from the left) is an impossibility without major, incompatible, and retroactive changes in how mail addressing is defined. Conversely, if the sender encodes the entire address, or the entire local part, without understanding the structure of the address in the same way that the target system does, it is likely that important information will be lost or, possibly, the message will be mis-delivered.

3.3 An Encoded-address, MUA-transparent, Solution may Eliminate an Important Opportunity

The principle above that addresses should have the same form in headers and in envelopes leads quickly to a reasoning path that argues for representation of most or all mail headers in some form of Unicode, including, but not limited to, those headers that explicitly list addresses. Several proposals have been outlined for doing this; perhaps the best developed (at the time the first version of this document was written) was the "UTF-8-HEADERS" proposal [Hoffman.UTF-8]. Like this proposal, it requires envelope (SMTP extension) negotiation to protect the headers that are encoded in UTF-8. This proposal differs from that one by putting somewhat more reliance on envelope facilities to prevent what its author considers a number of layering and interaction problems, most of them arising from the proposed "Address-map:" headers of that UTF-8-HEADERS proposal.

3.4 An MUA-only-based Solution is Not Necessary

3.4.1 Obtaining an Internationalized Email Address

One of the classic arguments for an approach based on MUA changes only (for international addresses or anything else) is that users will be able to install and use solutions on their own, even if the administrators of their systems are unenthused about the particular function or extension and delay, or decline, to install it. That argument was certainly true for MIME, especially in the presence of the capability to store messages as files and apply post-MUA tools. But it does not seem to apply for email addresses. In general, users cannot create email accounts or aliases controlling delivery of messages from external systems. Those accounts and aliases must be created by system administrators responsible for the mail servers. If those administrators are not sympathetic to internationalized mailbox names, such names will not exist on the receiving system. Having apparatus to send those names through the protocols will be essentially useless: a message that bounces because the relevant account or mailbox does not exist will bounce equally well whether the target address is in ASCII or in some other script and whether or not the receiving MTA is required to explicitly agree to access internationalized addresses. Conversely, if the administrators of the mail system host are sympathetic to internationalization, it is reasonable to expect that appropriate software can and will be installed at the MTA level.

An apparent important exception to the position taken in the above paragraph arises for subscription, often free, email services such as those operated under the "Hotmail", "Juno", "Netscape", and "Yahoo" names. Some of these systems permit users to select their own names (local parts) through an automated process. If the user creates a mailbox using an encoded name, users with MUAs that support the encoding will be able to send mail using a name in the user's preferred characters if, of course, the user or MTA somehow knows the encoding is being used. But the user cannot know what capabilities the correspondents will have available, and hence must give out both the name in local characters and the encoded form. This may turn out to be necessary, but is unlikely to be considered desirable. Also, if the user has presentation software that recognizes the coding conventions, then he or she will be able to see the original-language names in incoming messages and may not know what names to pass on to those who lack such presentation software. And, more important, many users of such systems access them through "web mail" interfaces, using standard (or at least common) browsers. The issues in getting those browsers upgraded to automatically recognize and decode special encoding forms may be as difficult, or more difficult, than those associated with convincing a system administrator to install a

special address or upgraded MTA.

Consider this practice from a user point of view. First, the domain names for these systems --as compared to institutional mail systems-- will generally continue to be in ASCII, so the goal of an email address that is entirely or predominantly in the user's language will be unattainable. If the domain names are non-ASCII (i.e., are IDNA encodings of non-ASCII strings), it is reasonable to assume that an operator who would choose such a name would be willing to internationalize its MTA. Second, such systems are most often accessed through web-based interfaces where most email header information appears to the user browser as running text. Because an email local part can, today, take on the form of almost any ASCII string, it is not reasonable to expect that a browser, even one with some localized functions, will be able to accurately detect an embedded, specially-coded, mailbox local part and correctly decode and render it. Heuristics based on detection of an at-sign ("@") will, of course, work for many cases, but will also produce a certain number of false positives, perhaps destroying URLs or examples in the text. After all, the "@" symbol has been around since long before there was email. It is worth noting that any recognition and decoding of local parts using a local encoding relies on heuristics that may fail: all such strings are historically-valid email local parts, and, unlike the DNS situation, it is impossible to conduct a reliable survey to determine that no one is using any particular encoding form, especially if the encoding indicator appears embedded in the local part string, rather than as a prefix. By contrast, if the MTA sees a Unicode string, and Unicode strings are placed in message headers and message bodies as needed, the transition may be more difficult, but no long-term user confusion or exposure to ugly encodings will be necessary.

To be very specific about this, if the local part of an address is encoded, e.g., with some ACE form as suggested in [JET-IMA], there is no practical way for the receiving system to know to decode it, rather than treat it as an ordinary mailbox name, unless it is notified via a SMTP envelope extension.

3.4.2 Relay environment

As in many other areas with email, many of the difficulties with an MTA-based model for internationalization of addresses arise, not when the originating MTA communicates directly with the delivery MTA, but when relay MTAs are involved. If the both the sending and receiving systems support internationalized addresses, it is still possible that an intermediate relay will not do so, forcing mail to bounce that could be delivered if there were a direct connection between sender and receiver. But, as with the installation of email

addresses on a system, relays do not get inserted in the mail path by accident. If internationalized addresses are important to the destination host, its administrators will chose lower-preference MX hosts or other relays that can support internationalized addresses.

3.4.3 Internationalizing the Sender

If we assume a destination host that can accept, and properly handle, an internationalized address, and we assume that any MX-designated intermediaries for that host will be chosen to be similarly capable, one situation is left in which it would be advantageous to have an MUA-only-based solution. If a originating/ sending system is not capable of generating or sending an internationalized address, but the prospective receiving system is, it would be good to enable the originating user to generate and somehow send to the relevant address.

This is a real issue, and deserves some serious consideration. But it seems better to find a good temporary, transitional, mechanism for it than to permanently burden the email system with an uncomfortable mechanism just to accommodate this case. One example of a transitional mechanism might be to use encapsulation, i.e., ESMTP tunneling over MIME [RFC2442], to route the address and message to a friendly gateway host that would unpack the message and transmit it using this specification. Other examples, less attractive at first glance but still plausible, would include defining and using small variations on the message encapsulation mechanisms that are integral to MIME [RFC2046], or the more complex encapsulation designed for HTML [RFC2557], to accomplish the same purpose.

The one transitional option that is not plausible is to simply send an encoded string without envelope modification. "Just send ACE" has the same unfortunate properties as the "just send eight" system proposed when MIME was adopted: in both cases, even if the message or address are not damaged in transit, the recipient will not have sufficient information available to be able to accurately determine what it has received and, hence, whether or not to decode it.

So, a user with an MUA that has the capability to handle an internationalized address, but who does not have access to an originating MTA with the capabilities defined here, may be given access to a reasonable transition strategy until the needed capabilities are available. Note that this does not require an open relay, since all of the user authentication capabilities of ESMTP [RFC2554] and SUBMIT [RFC2476] would be available. One can even imagine a service with a per-message charging system, which would presumably encourage rapid upgrading.

3.5 A Solution Based on MUA Changes Alone is Unworkable

The difficulties identified in the examples above are, perhaps obviously, not the only ones. Other issues arise with intermediate MX relay and gateway hosts, commands embedded in local parts, and special formats used in gateways to other environments, among other cases. Some of those additional cases are described briefly below.

3.5.1 MX Diversion

If the domain part of an email address is associated with several MX records and the mail is delivered to one of them that is not the best preference host, subsequent mail processing between that intermediate host and the ultimately destination one is not required to use SMTP. If, instead, it performs some gateway function, it may need to inspect or alter the local part to determine how to route and deliver the message. If the local part were encoded in some fashion that prevented that inspection process, and the MTA was not aware that it needed to apply special techniques, mail delivery might well fail.

3.5.2 Embedded commands

In addition to the address forms with special syntax or semantics described elsewhere, systems have been developed that embed commands in address local parts. These might, of course, use entirely different syntax constructions and formats than are typical in conventional addresses and, in an internationalized environment, might reasonably use character coding conventions that are neither ASCII nor Unicode-based.

A number of specialized applications of email do require, or recommend, specific syntax in the local part. These are identified, not to indicate that they are the only cases (they are not) but to reinforce the point that one must be quite cautious in doing anything that makes global assumptions about local part syntax and significant characters. These applications include local part explicit routing with the "percent hack" [RFC1123], gateways to and from X.400 environments [RFC2156], and gateways to fax systems [RFC3192].

3.6 Encoding the Whole Address String

Much of the above demonstrates why selective encoding of parts of the local-part string is not practical, will exclude many important cases, or will subject users to permanent use of the crytpic encoded forms. Why, then, not encode the entire string and insist that the delivery MTA recognize the presence of an encoded form and do whatever decoding is needed before it does other processing? There are three major reasons to approach the problem this way:

1. Any change in address syntax interpretation is likely to be a major, incompatible, change, since we do not now impose any restrictions on how an MTA is organized or even on how, or whether, the MTA, MUA, and other delivey-related functions are actually divided up on a given host. Converting user agents to handle international forms of addresses in a way that does not produce user astonishment is likely to be a major undertaking, regardless of what is done to the protocols and at what level.
2. Imposing a requirement that MTAs "understand" local-parts so that they can be partially decoded as part of mail routing would seem to defeat the main goal of encoding internationalized strings into a compact ASCII-compatible form, i.e., to keep MTAs from needing to understand the extended naming system
3. We potentially have three different encodings of an internationalized string: the one used by the MTA, the one used by the MUA, and the one seen by the user through applications software or the operating system's display interface. Having all three of these identical or closely compatible is desirable from the standpoint of user understanding and debugging. Having them different can cause many "interesting" problems, e.g., having to return an error message that uses different coding, and hence might represent an entirely different string, than the string the user put into the process.

Instead, it would seem sensible to move from a straightforward encoding of mail addresses in ASCII to a straightforward encoding in Unicode via UTF-8 [RFC2277], imposing only those restrictions on the characters in the local part that are implied by Unicode itself.

3.7 Looking back and looking forward

Another principle is implied by some of the discussion above. Internationalization measures for the Internet will be with us for as long as there are multiple languages and scripts in the world, i.e., probably forever. If a satisfactory long-term solution can be found, and a reasonable transition strategy can be defined for it, it is much better to optimize for the long term. The alternative of making things more difficult or less functional forever -- for the transport, the MUA, and/or the user interface system -- in order to save some small effort in transition, or even to make the transition a few months faster, represents a very poor tradeoff.

3.8 Summary of Design Issues and Tradeoffs

Each of the above subsections describes a strong case for continuing to treat addressing as an MTA function, opaque except at the end systems. The main alternative is to rely on the sending system being able to understand the addressing system of the target host, and any

relays accessed through MX relays, potentially needing to be able to remove IDN encoding ("punycode" or otherwise) in order to determine how to process or route the message. That alternative violates a long-standing and important design principle of Internet email, complicates a number of other cases, and does not offer sufficient transition advantages to be worth any of those difficulties.

The protocol proposed here takes a giant step toward true internationalization of electronic mail, providing a good functional approximation to what we might have done several decades ago had Unicode and the necessary understanding been available. It does not go as far as one could imagine going in providing address forms that would be compatible with local styles and models all over the world. The issues in considering, and taking, those extra steps are discussed in Section 8.

4. A Mail Transport-level Protocol

4.1 General Principles and Objectives

1. Whatever encoding is used should apply to the whole address and be directly compatible with software used at the user interface.
2. An SMTP relay must either recognize the format explicitly, agreeing to do so via an ESMTP option, select and use an ASCII-only address, or bounce the message so that the sender can make another plan.
3. In the interest of interoperability, charsets other than UTF-8 or punycode are strongly discouraged. If a mail environment chooses to use them anyway in the local part, interpretation at the "what does this mean" level is the responsibility of the receiving MTA.

4.2 Framework for the Internationalization Extension

The following service extension is defined:

1. The name of the SMTP service extension is "Address Internationalization";
2. The EHLO keyword value associated with this extension is "I18N";
3. No parameter values are defined for this EHLO keyword value. In order to permit future (although unanticipated) extensions, the EHLO response MUST NOT contain any parameters for that keyword. If a parameter appears, the SMTP client that is conformant to this version of this specification MUST treat the ESMTP response as if the I18N keyword did not appear.
4. An optional parameter is added to the SMTP MAIL and RCPT commands. This parameter is named ALT-ADDRESS. It requires an argument that may be useful in "downgrading" (see Section 4.5) as a substitute for the internationalized (UTF-8 coded) address.

This all-ASCII address MAY incorporate the IDNA "punycode" form if the domain name is internationalized. No algorithmic transformation is specified for the local-part; in the general case, it may identify a completely separate mailbox from the one identified in the primary command argument.

5. No additional SMTP verbs are defined by this extension.

Most of the remainder of this memo specifies how support for the extension affects the behavior of an SMTP client and server and what message header changes it implies.

4.3 The Address Internationalization Service Extension

In the absence of this extension, SMTP clients and servers are constrained to using only those addresses permitted by RFC 2821. The local parts of those addresses may be made up of any ASCII characters, although certain of them must be quoted as specified there. It is notable in an internationalization context that there is a long history on some systems of using overstruck ASCII characters (a character, a backspace, and another character) within a quoted string to approximate non-ASCII characters. This form of internationalization should be phased out as this extension becomes widely deployed but backward-compatibility considerations require that it continue to be supported.

An SMTP Server that announces this extension MUST be prepared to accept a UTF-8 string [RFC3629] in any position in which RFC 2821 specifies that a "mailbox" may appear. That string must be parsed only as specified in RFC 2821, i.e., by separating the mailbox into source route, local part and domain part, using only the characters colon (U+003A), comma (U+002C), and at-sign (U+0040) as specified there. Once isolated by this parsing process, the local part MUST be treated as opaque unless the SMTP Server is the final delivery MTA. Any domain names that are to be looked up in the DNS MUST be processed into punycode form as specified in IDNA [RFC3490] unless they are already in that form. Any domain names that are to be compared to local strings SHOULD be checked for validity and then MUST be compared as specified in IDNA.

An SMTP Client that receives the I18N extension keyword MAY transmit a mailbox name as an internationalized string in UTF-8 form. It MAY transmit the domain part of that string in either punycode (derived from the IDNA process) or UTF-8 form but, if it sends the domain in UTF-8, it SHOULD first verify that the string is valid for a domain name according to IDNA rules. As required by RFC 2821, it MUST not attempt to parse, evaluate, or transform the local part in any way. If the I18N SMTP extension is not offered by the Server, the SMTP Client MUST not transmit an internationalized address. Instead, it

MUST either return the message to the user as undeliverable or replace it, either using the ASCII-only address specified with the ALT-ADDRESS parameter or using some process (such as a directory lookup) outside the scope of this specification, with a local-part that conforms to the syntax rules of RFC 2821.

4.4 Extended Mailbox Address Syntax

RFC 2821, section 4.1.2, defines the syntax of a mailbox as

```
Mailbox = Local-part "@" Domain

Local-part = Dot-string / Quoted-string
      ; MAY be case-sensitive

Dot-string = Atom *("." Atom)

Atom = 1*atext

Quoted-string = DQUOTE *qcontent DQUOTE

Domain = (sub-domain 1*("." sub-domain)) / address-literal
sub-domain = Let-dig [Ldh-str]
```

(see that document for productions and definitions not provided here -- their details are not important to understanding this specification). The key changes made by this specification are, informally, to

- Change the definition of "sub-domain" to permit either the definition above or a UTF-8 (or other, see Section 7.1) string representing a DNS label that is conformant with IDNA [RFC3490]. That sub-domain string MUST NOT contain the characters "@" or ".".
- Change the definition of "Atom" to permit either the definition above or a UTF-8 (or other, see Section 7.3) string. That string MUST NOT contain any of the ASCII characters (either graphics or controls) that are not permitted in "atext"; it is otherwise unrestricted.

4.5 The ALT-ADDRESS parameter

If the I18N extension is offered, the syntax of the SMTP MAIL and RCPT commands is extended to support the optional "ALT-ADDRESS" parameter, which takes one argument. Its syntax is

"alt-address=" Mailbox

where "Mailbox" is strictly according to the (unextended) syntax specified in RFC 2821. If the receiving SMTP server is required to send the message on to a system that does not support this extension or the one for UTF-8 headers (see Section 6, it MAY substitute the specified address for the internationalized one to permit the message to be delivered to a mailbox specified by the sender. If UTF-8 headers are not supported, it SHOULD also encapsulate the message as described in Section 3.4.3. An SMTP server that cannot forward an internationalized message using the addressing and UTF-8 header extensions MUST either

- Substitute all-ASCII addresses and encapsulate the message, or
- "Bounce" the message, either rejecting it during the SMTP transaction or returning it, as specified in RFC 2821.

Under normal circumstances, the final delivery SMTP server should be configured so that the two mailbox names point to the same physical store. However, just as case-matching for traditional ASCII local-parts is not a requirement of the unmodified SMTP protocol, mapping of these two mailbox names is not a requirement here: sites for whom other issues outweigh the potential confusion should configure their systems as they find appropriate.

While further analysis is required, it will probably be desirable to extend the "Return-path:" trace header to include this parameter when it is provided; doing so would increase the odds that an error message could be properly routed in a number of edge and transition cases.

4.6 Formation of the Alternate Address

It is tempting to want to form the alternate, all-ASCII, address algorithmically so that, e.g., the sending MUA could compute it without any user input other than the native-form address. Such a computation could, for example, be the ACE transformation contemplated by [JET-IMA]. In the general case, this is not possible for the same reason that the use of an ASCII-compatible form without option negotiation is not feasible: the sending system cannot know the way in which the receiving one parses and interprets address local parts.

However, for the range of simple cases in which the local part really is atomic and represents a simple mailbox without subaddresses or other internal structure, it would probably be helpful to have a convention that the destination host could use to derive the alternate address, such as a standard ACE-style encoding. One can imagine prohibiting the use of alternate addresses that used selected ACE prefix except as an ACE-introducer so that the sending MUA, upon

receiving an reply or bounce on the alternate address could at least produce a helpful error message for the user.

4.7 Additional ESMTP Changes and Clarifications

The mail transport process involves addresses ("mailboxes") and domain names in contexts in addition to the MAIL and RCPT commands and extended alternatives to them. In general, the rule is that, when RFC 2821 specifies a mailbox, this document expects UTF-8 to be used for the entire string; when RFC 2821 specifies a domain name, the name should be in punycode form if its raw form is non-ASCII.

The following subsections list and discuss all of the relevant cases. [[Note in draft: I hope]]

4.7.1 The Initial SMTP Exchange

When an SMTP or ESMTP connection is opened, the server sends a "banner" response consisting of the 220 reply code and some information. The client then sends the EHLO command. Since the client cannot know whether the server supports internationalized addresses until after it receives the response from EHLO, any domain names that appear in this dialogue, or in responses to EHLO, must be in hostname form, i.e., internationalized ones must be in punycode form.

4.7.2 Trace Fields

Internationalized domain names in Received fields should be transmitted in Unicode form. Addresses in "for" clauses need further examination and might be treated differently depending on whether 8BITMIME is a requirement for internationalized addresses (See Section 6. The reasoning in the introductory portion of Section 5 strongly suggests that these addresses be in Unicode form, rather than some specialized encoding, but a counterargument is that users do not look at Received fields and, if there is a standard encoding available that is completely interoperable and information-preserving, it should be used for both domain names and addresses (perhaps in a comment or other supplemental information).

5. Impact on the MUA and on Message Headers

In addition to the trace fields ("Received" headers), mentioned above, there are many other places in MUAs or in user presentation in which email addresses or domain names appear. Each one, whether the conventional From, To, or Cc header fields, or Message-IDs, or In-Reply-To fields that may contain addresses or domain names, or in message bodies or elsewhere, must be examined from an

internationalization perspective. The user will expect to see mailbox and domain names in local characters, and to see them consistently: a situation in which an address is coded one way in a "From" field, another way in a signature line in the body of a the message, and, apparently arbitrarily, in one or the other of those forms in Return-Path, Received, or reference fields, will create confusion and frustration. Variations on that problem will exist with any internationalization method, whether transport or MUA-only in structure. Perhaps, if we have to live with it for a short time as a transition activity, that is worthwhile. But the only practical way to avoid it, in both the medium and the longer term, is to have the encodings used in transport be as nearly as possible the same as the encodings used in message headers and message bodies.

There appears to be a very strong case for concluding that the point at which we internationalize email local parts is the point that we should simply shift email headers to a full internationalized form, presumably using UTF-8 rather than ASCII. The transition to that model might involve support for legacy systems by extending the encoding models of [RFC2045] and [RFC2231] to cover address, and address-related, fields within headers but our target should be fully internationalized headers, as discussed elsewhere in this document.

6. Bundling of Extensions and Options

An ongoing concern about any extensions to SMTP is that they will combine to create a situation with enough possible combinations to require profiling and the consequent increased complexity and negative impact on interoperability. Reducing the number of different options and combinations of them is therefore a useful goal. In this particular case, we have

1. This proposal, to permit UTF-8 addresses and a alternate addresses when it is not possible to reliably transmit or use the UTF-8 ones.
2. Several proposals to permit the use of UTF-8 (or other non-ASCII encodings) in mail headers. See, e.g., [Hoffman.UTF-8]
3. Several proposals to protect those UTF-8 headers with an SMTP extension. See, again, e.g., [Hoffman.UTF-8]
4. The established "8BITMIME" extension [RFC1652] that permits message bodies to be transmitted in 8 bit form, rather than requiring a content transfer encoding to force them into 7 bit form. Given the difficulties that would occur if UTF-8 (or other 8 bit) headers were significantly encoded, this extension is all but required for the UTF-8 header extensions to function properly.

5. Possibly also some type of "get envelope components out of the headers" proposal, such as [Klensin.envelope]

In the interest of compatability and interoperability, it is suggested that all of these extensions be bundled, with only one EHLO keyword for all of them other than 8BITMIME, and with that new keyword implying (and requiring support for) all of the listed functions. This requires MTA implementers who wish to support these features to do a slightly larger job, but avoids the interoperability costs of excessive incrementalism. Put differently, internationalization, taken seriously, is a large issue and a large job, and these appear to be the pieces needed to get the job done

7. Protocol Loose Ends

These issues should be resolved, and this section eliminated, before the document is considered complete.

7.1 Punycode in Domain Names?

It is not clear whether the flexibility of being able to pass domain names in punycode, as well as UTF-8, form is needed. If it is not, it should be eliminated as excess complexity.

7.2 Local Character Codes in Local Parts?

There are some reasons for permitting local-parts to be written in locally-used character codes, i.e., in other than the UTF-8 encoding of UNICODE. This could be done by tagging the local part in a fashion similar to the technique of [RFC2047] but without encoding the local part string itself. It clearly increases flexibility, and the mailbox part can be defined as a simple octet string (as it essentially is in the sections above). We can reasonably expect that some systems, operating in local environments, will use local character codes no matter what we specify. On the other hand, having an application presented with an octet (or bit) string and not knowing what charset is involved would wreak havoc on any attempt to intelligently display local parts: if one cannot know the character coding being used, then it is not possible to accurately decode the characters and display appropriate character glyphs.

Use of local coding also implies an encoding for the local part different from that for the domain part -- any MTA in the path must be able to resolve the domain part into something that can be looked up in the DNS and resolved and that, in turn, requires a globally-known encoding.

On the other hand, if local codings can be avoided entirely, it will

considerably reduce complexity and "opportunities" for systems to not interoperate.

7.3 Restrictions on Characters in Local Part?

This specification is extremely liberal about what can be included in a UTF-8 string that represents a local-part. In return, it effectively prohibits the use of quoted strings, or quoted characters, in non-ASCII local parts. Quoted strings and characters in local parts have, in general, been nothing but trouble and there appears to be no reason to carry that trouble forward into an internationalized world (and the much greater complexity that quoting in that environment might imply). There may also be a strong case for applying restrictions, e.g., by use of a stringprep [RFC3454] profile that would eliminate particularly problematic characters while not forcing, e.g., even an approximation to case-mapping (remember that ASCII local-parts are inherently case sensitive, even though local systems are encouraged to not take advantage of that feature).

7.4 Requirement for 8BITMIME?

This extension is carefully defined to be independent of "8BITMIME". However, given the length of time 8BITMIME has been around, the amount of deployment of it that exists, and the rather low likelihood that any MTA implementer in his or her right mind will go to the trouble of implementing this extension without also implementing 8BITMIME, it may be sensible to permit this extension only if 8BITMIME also appears or is combinated with it as suggested in Section 6

7.5 Message Header and Body Issues with MTA Approach?

By viewing i18n addresses as an MTA problem, this document may not address all of the interesting 2822/MIME and MUA implementation and presentation style issues.

In particular, if both this extension and 8BITMIME are in use, is it sensible to drop the requirement for RFC 2047/ 2231 encoding of personal name fields? And, whether or not that requirement is dropped, is the MUA description of Section 5 adequate?

7.6 The Received field 'for' clause

Decide what to do about the value of the "for" clause in Received fields. See Section 4.7.2.

8. Internationalization and Full Localization

Whenever one considers a new protocol, or revision of an existing one, for internationalization or other aspects of support for an improved user interface, important tradeoffs arise. These tradeoffs can be described in several ways, e.g.,

- Simplicity versus localization capability
- User convenience, especially within a particular area or culture versus global interoperability
- and so on

Maximum global interoperability is obtained by confining a protocol to an very limited number of characters, ideally ones that are easily distinguishable by people. The historical choice in this regard has been the 26 upper-case ASCII letters, plus digits, plus a very small number of special characters. It is probably no coincidence that these characters (with different, bit-minimizing, encodings) are the normal ones in early telegraphy and subsequent Telex character sets. But, as soon as users start looking at these characters, the complaints start to appear: text in all-upper-case is ugly, people should be able to write their names as they normally do and not in some transliterated or variant form, people should be able to communicate in their own languages using their own character sets, and so on. Ultimately, not only are the characters used in writing at issue; so are the structures for constructing, e.g., command sequences, with different preferences typically reflecting the grammatical structures of different languages. With sufficient ingenuity, all of these requirements can be accommodated, but only typically at the cost of global interoperability or at the expense of convenient use by people outside the locality or cultural group.

Email addresses illustrate this problem at its most difficult. They are seen and used by end users and there has been little success in hiding the forms that are actually used in the protocols. Worldwide, most communication is almost certainly among people who share languages and cultural assumptions, not in situations in which global interoperability is important (and where it is important that global interoperability be convenient and very reliable). On the other hand, situations and communications that require global interoperability are still common and are commercially and intellectually important.

So the question is how far should one go. It is clearly important and sensible to accommodate local character sets, and to do so in a way that creates maximum convenience and attractive user interfaces in the long term. But, as pointed out in passing in Section 4.3, RFC2821 still requires the ASCII at-sign character to divide the

local part from the domain name. If even lexical support for the long-deprecated source routes is to be provided, comma and colon must also be preserved and supported. This implies that a mailbox name that is completely in some character script other than ASCII is impossible without further changes to the email protocols. In addition, the ordering implied by the "local-part@domain" construction, usually read in English as "local part at domain", seems quite strange and foreign in some other languages and cultures. It is interesting that X.400 avoided this delimiter and ordering problem entirely by using Distinguished Names in which the various elements of an address were explicitly identified. But, when Distinguished Names appear at the presentation layer or above, they appear with the various fields identified by tags which are, themselves, keywords that use a very restricted set of ASCII (actually ISO 646 or IA5) characters.

In principle at least, the protocol extensions proposed here could be further extended to specify a separator character to distinguish local part from domain name and the order in which those names occurred. For example, the MAIL and RCPT commands could be extended with parameters like

SEPARATOR="UTF-8-character" ORDER-RL to identify a form consisting of the domain name followed by the local part, separated by the designated character

But, while this would not impose particularly heavy burdens on SMTP processors, it would be a potential nightmare for users, who would have no way to accurately identify the components of an email address, at least without significant out-of-band information. In addition, going that far would almost certainly touch off the debate, again, as to whether domain names should be presented in little-endian or big-endian order -- an issue that is, again, culturally sensitive as to which one feels most natural.

It is not clear how far one should go, and the community should consider the issue very carefully.

9. Advice to Designers and Operators of Mail-receiving Systems

As discussed above, in the historical Internet email context, the interpretation and permitted syntax for an email local-part is entirely the responsibility of the receiving system. Systems can get themselves into trouble and, more particularly, can seriously restrict the number and type of users who can send mail to their users, by poor choices of format and syntax. For example, general advice to system designers has long included "treat addresses in a case-independent fashion" and "do not use addresses that require

quoting" in order to increase the odds that remote users will be able to properly compose and transmit intended addresses. In a way, that advice is an extreme generalization of the "receiver" side of the robustness principle: being generous in what one accepts implies accepting as many plausible variations of an address local-part string as possible and designing the strict forms of those strings to facilitate differentiation when it is appropriate.

As one moves toward internationalization of local parts, an expanded version of these principles is useful and may be even more appropriate, even though it is neither necessary nor desirable to turn those principles into protocol requirements. For example, a receiving host should normally consider any string that would match under nameprep rules --or perhaps any string that would match under a stringprep profile that provides more matching and exclusions than nameprep-- as matching for local-part purposes. An even more "liberal" receiving host might use some sort of variant tables for its script(s) of interest to further expand the matching rules.

But, whatever extended matching rules the local host adopts, those rules are a property of that host. Senders should continue to be conservative about what they send, and relays should continue to avoid presumptions about their understanding of the content of local-parts. Receiving systems that have reason to adopt more restricted syntax rules, or interpretations of matching, should continue to be able to do so.

10. Internationalization Considerations

This entire specification addresses issues in internationalization and especially the boundaries between internationalization and locationalization and between network protocols and client/user interface actions.

11. IANA Considerations

This specification does not contemplate any IANA registrations or other actions.

12. Security considerations

Any expansion of permitted characters and encoding forms in email addresses raises the risk, however slight, of misdirected or undeliverable mail. The problem is worsened if address information is carried in local character sets and must be converted to some standard form. Any conversion of character sets may also be problematic for digitally-signed information. Modulo those concerns, the ideas proposed here do not introduce new security issues.

Since email addresses are often transcribed from business cards and notes on paper, they are subject to problems arising from confusable characters. Those problems are somewhat reduced if the domain associated with the mailbox is unambiguous and supports a relatively small number of mailboxes whose names follow local system conventions; they are increased with very large mail systems in which users can freely select their own addresses.

13. Acknowledgements

The author acknowledges the contributions and comments of Dave Crocker in a personal conversation, and the efforts of a private discussion group, led by Paul Hoffman and Adam Costello, to develop an MUA-only solution to this problem. The author had hoped that effort would succeed, since the idea of requiring transport changes to support internationalization (or any other new function) is unattractive and should be avoided when possible. Difficulties that group has encountered in properly defining a number of boundary conditions, including appropriate delimiters for permitting internal parsing of the local part and problems with right-to-left characters and substrings, have led to the conclusion that it is time to get a specific, transport-based, approach on the table. That conclusion has been reinforced by increasing interest in the IETF in more radical changes to the mail system, starting with extensions to permit mail headers to be written in UTF-8. While the ideas leading to the "IMAA" and other drafts have inspired several of the properties of this proposal, their authors are, of course, not responsible for the result and will probably disagree with it. Comments from Adam Costello on the first public draft were particularly helpful, and James Seng identified some internationalization issues that had not been addressed in the previous version.

14. An Appeal

The author received a number of favorable comments on the general principles and design discussed in early drafts of this specification. He is not, however, able to continue its development as a one-person, or even one-person with occasional comments from others, basis. In particular, he has almost no resources for developing MTA, MUA, or presentation code to test and demonstrate the concepts and details outlined above; without such resources, this approach will, inevitably, fail sooner or later. So those who consider the idea attractive should think about, and develop, ways to join with the author in design team and development efforts.

15. References

15.1 Normative References

[Hoffman.UTF-8]
Hoffman, P., "SMTP Service Extensions for Transmission of Headers in UTF-8 Encoding", draft-hoffman-utf8headers-00 (work in progress), December 2003.

[Klensin.envelope]
Klensin, J., "A Cleaner SMTP Envelope for Internet Mail", draft-klensin-email-envelope-00 (work in progress), January 2004.

[RFC0821] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[RFC1651] Klensin, J., Freed, N., Rose, M., Stefferud, E., and E. Crocker, "SMTP Service Extensions", RFC 1651, July 1994.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels'", RFC 2119, March 1997.

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

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

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

15.2 Informative References

[Hoffman-IMAA]
Hoffman, P. and A. Costello, "Internationalizing Mail Addresses in Applications (IMAA)", draft-hoffman-imaa-03 (work in progress), October 2003.

[JET-IMA] Yao, J. and J. Yeh, "Internationalized eMail Address (IMA)", draft-lee-jet-ima-00 (work in progress), June 2005.

[RFC1652] Klensin, J., Freed, N., Rose, M., Stefferud, E., and E. Crocker, "SMTP Service Extensions", RFC 1652, July 1994.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[RFC2056] Denenberg, R., Kunze, J., and D. Lynch, "Uniform Resource Locators for Z39.50", RFC 2056, November 1996.

[RFC2156] Kille, S., "MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME", RFC 2156, January 1998.

[RFC2231] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, November 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2442] Freed, N., Newman, D., and Hoy, M., "The Batch SMTP Media Type", RFC 2442, November 1998.

[RFC2476] Gellens, R. and J. Klensin, "Message Submission", RFC 2476, December 1998.

[RFC2554] Myers, J., "SMTP Service Extension for Authentication", RFC 2554, March 1999.

[RFC2557] Palme, F., Hopmann, A., Shelness, N., and E. Stefferud, "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", RFC 2557, March 1999.

[RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[RFC3192] Allocchio, C., "Minimal FAX address format in Internet Mail", RFC 3192, October 2001.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

Author's Address

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

Phone: +1 617 491 5735
Email: john-ietf@jck.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.