           The Use of URLs as Meta-Syntax for Core Mail List Commands
               and their Transport through Message Header Fields

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   The mailing list command specification header fields are a set of
   structured fields to be added to email messages sent by email
   distribution lists. Each field typically contains a URL (usually
   mailto [RFC2368]) locating the relevant information or performing the
   command directly. The three core header fields described in this
   document are List-Help, List-Subscribe, and List-Unsubscribe.

   There are three other header fields described here which, although
   not as widely applicable, will have utility for a sufficient number
   of mailing lists to justify their formalization here. These are
   List-Post, List-Owner and List-Archive.

   By including these header fields, list servers can make it possible
   for mail clients to provide automated tools for users to perform list
   functions. This could take the form of a menu item, push button, or
   other user interface element. The intent is to simplify the user
   experience, providing a common interface to the often cryptic and
   varied mailing list manager commands.

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119.

1. Introduction

   This is a proposal for additional header fields to be added to email
   messages sent by email distribution lists. The content of each new
   field is typically a URL - usually mailto [RFC2368] - which locates
   the relevant information or performs the command directly. MTAs
   generating the header fields SHOULD usually include a mailto based
   command, in addition to any other protocols used, in order to support
   users who do not have access to non-mail-based protocols.

   Implementing these fields will be optional. Significant functionality
   and convenience can be gained by including them, however. Many list
   managers, especially as the proposal first gains acceptance, MAY
   choose to implement only one or two of the fields. The List-Help
   field is the most useful individual field since it provides an access
   point to detailed user support information, and accommodates almost
   all existing list managers command sets. The List-Subscribe and
   List-Unsubscribe fields are also very useful, but cannot describe
   some list manager syntaxes at this time (those which require variable
   substitution). See appendix A.5 for an explanation.

   The description of command syntax provided by the fields can be used
   by mail client applications to provide simplified and consistent user
   access to email distribution list functions. This could take the form
   of menu items, push buttons, or other user interface elements. The
   intent is to simplify the user experience, providing a common
   interface to the often cryptic and varied mailing list manager
   commands.

   Consideration has been given to avoiding the creation of too many
   fields, while at the same time avoiding the overloading of individual
   fields and keeping the syntax clear and simple.

   The use of these fields does not remove the requirement to support
   the -Request command address for mailing lists [RFC2142].

2. The Command Syntax

   The list header fields are subject to the encoding and character
   restrictions for mail headers as described in [RFC822]. Additionally,
   the URL content is further restricted to the set of URL safe
   characters [RFC1738].

   The contents of the list header fields mostly consist of angle-
   bracket ('<', '>') enclosed URLs, with internal whitespace being
   ignored. MTAs MUST NOT insert whitespace within the brackets, but
   client applications should treat any whitespace, that might be
   inserted by poorly behaved MTAs, as characters to ignore.

A list of multiple, alternate, URLs MAY be specified by a comma-
separated list of angle-bracket enclosed URLs. The URLs have order of
preference from left to right. The client application should use the
left most protocol that it supports, or knows how to access by a
separate application. By this mechanism, protocols like http may be
specified while still providing the basic mailto support for those
clients who do not have access to non-mail protocols. The client
should only use one of the available URLs for a command, using
another only if the first one used failed.

The use of URLs allows for the use of the syntax with existing URL
supporting applications. As the standard for URLs is extended, the
list header fields will gain the benefit of those extensions.
Additionally, the use of URLs provides access to multiple transport
protocols (such as ftp and http) although it is expected that the
"mailto" protocol [RFC2368] will be the focus of most use of the list
header fields. Use of non-mailto protocols should be considered in
light of those users who do not have access to the specified
mechanism (those who only have email - with no web access).

Command syntaxes requiring variable fields to be set by the client
(such as including the user's email address within a command) are not
supported by this implementation. However, systems using such
syntaxes SHOULD still take advantage of the List-Help field to
provide the user with detailed instructions as needed or - perhaps
more usefully - provide access to some form of structured command
interface such as an HTML-based form.

The additional complications of supporting variable fields within the
command syntax was determined to be too difficult to support by this
protocol and would compromise the likelihood of implementation by
software authors.

To allow for future extension, client applications MUST follow the
following guidelines for handling the contents of the header fields
described in this document:

1) Except where noted for specific fields, if the content of the
   field (following any leading whitespace, including comments)
   begins with any character other than the opening angle bracket
   '<', the field SHOULD be ignored.

2) Any characters following an angle bracket enclosed URL SHOULD be
   ignored, unless a comma is the first non-whitespace/comment
   character after the closing angle bracket.

   3) If a sub-item (comma-separated item) within the field is not an
      angle-bracket enclosed URL, the remainder of the field (the
      current, and all subsequent, sub-items) SHOULD be ignored.

3. The List Header Fields

      This document presents header fields which will provide the
      command syntax description for the 'core' and key secondary
      functions of most email distribution lists. The fields implemented
      on a given list SHOULD be included on all messages distributed by
      the list (including command responses to individual users), and on
      other messages where the message clearly applies to one distinct
      list. There MUST be no more than one of each field present in any
      given message.

      These fields MUST only be generated by mailing lists, not end
      users.

3.1. List-Help

      The List-Help field is the most important of the header fields
      described in this document. It would be acceptable for a list
      manager to include only this field, since by definition it SHOULD
      direct the user to complete instructions for all other commands.
      Typically, the URL specified would request the help file, perhaps
      incorporating an HTML form for list commands, for the list, and
      alternatively provide access to an instructive website.

      Examples:

      List-Help: <mailto:list@host.com?subject=help> (List Instructions)
      List-Help: <mailto:list-manager@host.com?body=info>
      List-Help: <mailto:list-info@host.com> (Info about the list)
      List-Help: <http://www.host.com/list/>, <mailto:list-info@host.com>
      List-Help: <ftp://ftp.host.com/list.txt> (FTP),
          <mailto:list@host.com?subject=help>

3.2. List-Unsubscribe

   The List-Unsubscribe field describes the command (preferably using
   mail) to directly unsubscribe the user (removing them from the list).

   Examples:

      List-Unsubscribe: <mailto:list@host.com?subject=unsubscribe>
      List-Unsubscribe: (Use this command to get off the list)
          <mailto:list-manager@host.com?body=unsubscribe%20list>
      List-Unsubscribe: <mailto:list-off@host.com>

```
List-Unsubscribe: <http://www.host.com/list.cgi?cmd=unsub&lst=list>,
    <mailto:list-request@host.com?subject=unsubscribe>
```

3.3. List-Subscribe

   The List-Subscribe field describes the command (preferably using
   mail) to directly subscribe the user (request addition to the list).

   Examples:

```
   List-Subscribe: <mailto:list@host.com?subject=subscribe>
   List-Subscribe: <mailto:list-request@host.com?subject=subscribe>
   List-Subscribe: (Use this command to join the list)
       <mailto:list-manager@host.com?body=subscribe%20list>
   List-Subscribe: <mailto:list-on@host.com>
   List-Subscribe: <http://www.host.com/list.cgi?cmd=sub&lst=list>,
       <mailto:list-manager@host.com?body=subscribe%20list>
```

3.4. List-Post

   The List-Post field describes the method for posting to the list.
   This is typically the address of the list, but MAY be a moderator, or
   potentially some other form of submission. For the special case of a
   list that does not allow posting (e.g., an announcements list), the
   List-Post field may contain the special value "NO".

   Examples:

```
   List-Post: <mailto:list@host.com>
   List-Post: <mailto:moderator@host.com> (Postings are Moderated)
   List-Post: <mailto:moderator@host.com?subject=list%20posting>
   List-Post: NO (posting not allowed on this list)
```

3.5. List-Owner

   The List-Owner field identifies the path to contact a human
   administrator for the list. The URL MAY contain the address of a
   administrator for the list, the mail system administrator, or any
   other person who can handle user contact for the list. There is no
   need to specify List-Owner if it is the same person as the mail
   system administrator (postmaster).

   Examples:

```
   List-Owner: <mailto:listmom@host.com> (Contact Person for Help)
   List-Owner: <mailto:grant@foo.bar> (Grant Neufeld)
   List-Owner: <mailto:josh@foo.bar?Subject=list>
```

3.6. List-Archive

   The List-Archive field describes how to access archives for the list.

   Examples:

      List-Archive: <mailto:archive@host.com?subject=index%20list>
      List-Archive: <ftp://ftp.host.com/pub/list/archive/>
      List-Archive: <http://www.host.com/list/archive/> (Web Archive)

4. Supporting Nested Lists

   A list that is a sublist for another list in a nested mailing list
   hierarchy will need to modify some of the List- header fields, while
   leaving others as the parent list set them.

   Sublists SHOULD remove the parent list's List-Help, List-Subscribe,
   List-Unsubscribe and List-Owner fields, and SHOULD insert their own
   versions of those fields.

   If the sublist provides its own archive, it SHOULD replace the List-
   Archive with its own. Otherwise, it MUST leave the List-Archive field
   untouched.

   Dependant on how postings to the list are handled, the sublist MAY
   replace the List-Post field. The appropriateness of whether to
   replace List-Post is left to the determination of the individual list
   managers. If the intention is that postings should be distributed to
   all members of the primary list, List-Post should not be changed by a
   sublist in such a way that postings will be distributed only to
   members of the sublist.

5. Security Considerations

   There are very few new security concerns generated with this
   proposal. Message headers are an existing standard, designed to
   easily accommodate new types. There may be concern with multiple
   fields being inserted or headers being forged, but these are problems
   inherent in Internet email, not specific to the protocol described in
   this document. Further, the implications are relatively harmless.

   Mail list processors should not allow any user-originated list header
   fields to pass through to their lists, lest they confuse the user and
   have the potential to create security problems.

   On the client side, there may be some concern with posts or commands
   being sent in error. It is required that the user have a chance to
   confirm any action before it is executed. In the case of mailto, it

may be appropriate to create the correctly formatted message without
sending it, allowing the user to see exactly what is happening and
giving the user the opportunity to approve or discard the message
before it is sent.

All security considerations for the use of URLs [RFC1738] apply
equally to this protocol. Mail client applications should not support
list header field URLs which could compromise the security of the
user's system. This includes the "file://" URL type which could
potentially be used to trigger the execution of a local application
on some user systems.

6. Acknowledgements

The numerous participants of the List-Header [5], ListMom-Talk [6],
List-Managers and MIDA-Mail mailing lists contributed much to the
formation and structure of this document.

Keith Moore <moore@cs.utk.edu> and Christopher Allen
<ChristopherA@consensus.com> provided guidance on the standards
process.

A. Background Discussion

   This proposal arose from discussions started on the ListMom-Talk
   Discussion List [6]. When the discussion reached a sufficient level,
   a separate list was formed for discussing this proposal, the List
   Headers Mail List [5] for deeper discussion. We have included
   summaries of key issues raised, in order to show some of the
   alternatives examined and reasons for our decisions.

A.1. Multiple header fields vs. a single header field

   Use of a single header field for transporting command meta-syntax was
   rejected for a number of reasons.

   Such a field would require the creation of a new meta-syntax in order
   to describe the list commands (as opposed to the use of the widely
   deployed URL syntax which was chosen for this implementation). Every
   additional layer of complexity and newness reduces the likelihood of
   actual implementation because it will require additional work to
   support. Also, by using the existing URL syntax, we can profit from
   the end users' knowledge of that syntax and ability to use it even if
   their client applications do not support the list header fields.

   Restricting the transport of meta-syntax to the use of a single
   header field also introduces complications with header field size
   limitations. Most individual commands can easily be described in a
   single line, but describing a multitude of commands can take up many
   lines in the field and runs a greater risk of being modified by an
   existing server on route.

   The client implementation is also easier with multiple fields, since
   each command can be supported and implemented individually,
   completely independent of the others. Thus, some list managers or
   mail clients can choose to implement a subset of the fields based on
   the specific needs of their individual lists.

   Finally, the format described in this document is simple and well
   recognized, which reduces the chances of errors in implementation and
   parsing.

A.2. URLs vs. parameter lists

   URLs are already an established syntax which is flexible, well-
   defined, and in wide spread use. As its definition matures and
   expands, the abilities of the list fields will grow as well, without
   requiring modification of this proposal. URLs are well prepared to
   handle future protocols and developments, and can easily describe the
   different existing access protocols such as mailto, http and ftp.

Many clients already have functionality for recognizing, parsing, and
evaluating URLs, either internally or by passing the request to a
helper application. This makes implementation easier and more
realistic. As an example, this existing support for URL parsing
allowed us to add prototype list header functionality to existing
mail clients (Eudora and Emailer for the Macintosh) without modifying
their source code.

A.3. Why not just create a standard command language?

A standard command language, supported by all email list services,
would go a long way to reducing the problems of list access that
currently plague existing services. It would reduce the amount of
learning required by end users and allow for a number of common
support tools to be developed.

However, such standardization does pose problems in the areas of
multi-lingual support and the custom needs of individual mailing
lists. The development of such a standard is also expected to be met
with a slow adoption rate by software developers and list service
providers.

These points do not preclude the development of such a standard (in
fact, it would suggest that we should start sooner rather than
later), but we do need a solution that can be widely supported by the
current list services.

We can support most existing list manager command syntaxes without a
standard command language. By using URLs, we allow alternate access
methods a standard command language probably wouldn't enable, such as
web based control.

Finally, client support for a standard command language is not at all
clear or necessarily simple to implement. The variety and large
number of commands existing today would require complicated user
interfaces which could be confusing and difficult to implement. By
restricting this proposal to the core functions, the client

implementation is much simpler, which significantly increases the
likelihood of implementation (as evidenced by the support already
announced by a number of client and server application authors).

A.4. Internationalization

Multilingual support is up to the URL standard. If URLs support it,
then the List- header fields support it. This is another advantage of
using URLs as the building blocks for the list header fields.

A.5. Variable Substitution

   Variables would allow the List- header fields to accommodate nearly
   every existing list manager. However, it would immeasurably increase
   the complexity of the entire proposal, and possibly involve
   redefining the URL standard, or force us to use something more
   complicated (and hence more difficult to implement) than URLs to
   describe the command syntax.

   Parameters would either have to be mandatory (i.e. the user agent
   doesn't submit the message if it doesn't know what text to
   substitute) or you need a way to say "if you know this parameter, add
   its text here; otherwise, do this" where "this" is either: (a)
   substitute a constant string, or (b) fail.

   The reason you would want a facility like this is because some list
   server applications insist on having certain parameters like users'
   names, which the user agent might or might not know. e.g. listserv
   insists on having a first name and a last name if you supply either
   one.

   Which could lead to something like the UNIX shell syntax, where
   ${foo-bar} means substitute the value of parameter "foo" if "foo" is
   defined, else substitute the string "bar". Perhaps $foo would mean
   "substitute the value of parameter foo if it is defined, else
   substitute the empty string"

   This all seems far too complicated for the gains involved, especially
   since the use of variables can often be avoided.

   The use of variables in the command syntaxes of list services appears
   to be lessening and does not, in any case, apply to all commands.
   While the unsubscribe and subscribe command header fields may not be
   usable by those systems which require the use of variables, the help
   field will still provide end users with a consistent point of access
   through which they can get support for their use of the list.

A.6. Why not use a specialized MIME part instead of header fields?

   MIME parts were considered, but because most mail clients currently
   either don't support MIME or are not equipped to handle such
   specialized parts - such an implementation would result in problems
   for end users. It is also not as easy for many list servers to
   implement MIME as it is to implement new header fields.

   However, we are looking at the design of a MIME part to more fully
   describe list command syntax, as well as trying to find ways to get
   it supported by the applicable software.

A.7. Why include a Subscribe command?

   Subscribe and Unsubscribe are the key commands needed by almost every
   list. Other commands, such as digest mode, are not as widely
   supported.

   Additionally, users who have unsubscribed (before going on vacation,
   or for whatever other reason) may want to resubscribe to a list. Or,
   a message may be forwarded/bounced from a subscriber to a non-
   subscriber. Or, the user may change addresses and want to subscribe
   from their new address. Having the List-Subscribe field available
   could certainly help in all these cases.

A.8. The Dangers of Header Bloat

   At what point are there just too many header fields? It really
   varies on a list by list basis. On some lists, the majority of users
   will never be aware of a field unless the client software provides
   some alternative user interface to it (akin to the Reply-To field).
   On others, the users will often see the header fields of messages and
   would be able to recognize the function of the URLs contained within.

   The flexibility afforded by the protocol described in this document
   (in that the header fields may be individually implemented as deemed
   appropriate) provides list administrators with sufficient 'room to
   maneuver' to meet their individual needs.

B. Client Implementation

B.1. Guidelines

   For 'mailto' URL based commands, mail client applications may choose
   to provide specialized feedback (such as presenting a dialog or
   alert), instead of the actual command email message, asking for
   command confirmation from the user. The feedback should identify the
   message destination and command within a more descriptive
   explanation. For example:

      "Do you want to send the unsubscription command 'unsubscribe
      somelist' to 'somelist-request@some.host.com'? Sending the command
      will result in your removal from the associated list."

   If the user has multiple email addresses supported by the mail
   client, the client application should prompt the user for which
   address to use when subscribing or performing some other action where
   the address to use cannot be specifically determined. When
   unsubscribing or such, the address that is subscribed should be used,
   unless that is not known by the application and cannot be determined
   from the message headers.

B.2. Implementation Options

   The following implementation possibilities are suggested here to give
   some idea as to why these new header fields will be useful, and how
   they could be supported.

   In most cases, it may be helpful to disable the interface for the
   commands when not applicable to the currently selected message.

B.2.1. Key combinations and command lines

   On text based systems which utilize command lines or key
   combinations, each field could be implemented as a separate command.
   Thus one combination would subscribe the user, another would
   unsubscribe, a third request help, etc. The commands would only be
   available on messages containing the list header fields.

B.2.2. Menu items

   On graphical systems which have menus, these commands could take the
   form of a menu or sub-menu of items. For example, a "Lists" menu
   might appear when viewing messages containing the header fields, with
   items named "Subscribe", "Unsubscribe", "Get Help", "Post Message to

List", "Contact List Owner" and "Access List Archive". This menu
could be disabled when not applicable to the current message or
disappear entirely.

B.2.3. Push Buttons and Pallettes

On graphical window systems, buttons could be placed in the window of
the message, a toolbar, or in a floating pallette of their own. Each
button could correspond to a command, with names "Subscribe",
"Unsubscribe", "Get Help", "Post to List", "List Owner" and
"Archive". These buttons or pallettes could be disabled when not
applicable to the current message or disappear entirely.

B.2.4 Feedback to the User

If using a dialog interface (or other feedback element) the client
application MUST include an option for the user to review (and
possibly modify) the message before it is sent. The application may
also find it useful to provide a link to more detailed context-
sensitive assistance about mail list access in general.

References

    [RFC822] Crocker, D., "Standard for the Format of ARPA
             Internet Text Messages", STD 11, RFC 822, August 1982.

    [RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill,
              "Uniform Resource Locators (URL)" RFC 1738, December 1994.

    [RFC2142] Crocker, D., "Mailbox Names for Common Services, Roles and
              Functions", RFC 2142, May 1997.

    [RFC2368] Hoffman, P., Masinter, L., and J. Zawinski, "The mailto URL
              scheme", RFC 2368, July 1998.

    [5] "List-Header" Mail list. list-header@list.nisto.com
        <URL:http://www.nisto.com/listspec/mail/>
        <URL:http://www.nisto.com/listspec/>

    [6] "ListMom-Talk" Mail list. listmom-talk@skyweyr.com
        <URL:http://cgi.skyweyr.com/ListMom.Home>

Editors' Addresses

     Joshua D. Baer
     Box 273
     4902 Forbes Avenue
     Pittsburgh, PA 15213-3799
     USA

     EMail: josh@skyweyr.com

     Grant Neufeld
     Calgary, Alberta
     Canada

     EMail: grant@acm.org
     Web: http://www.nisto.com/

Full Copyright Statement