The following paper was originally published
in the Proceedings of the
Tenth USENIX System Administration Conference (LISA X)
Chicago, IL, USA, Sept. 29 - Oct. 4, 1996

# MajorCool: A Web Interface to Majordomo

Bill Houle
NCR Corporation

# MajorCool: A Web Interface To Majordomo

*Bill Houle* – NCR Corporation


## ABSTRACT

Majordomo is a popular mailing list manager and auto-responder that is administered via embedded e-mail commands. The mail-based command structure and side-effects introduced by the behavior of certain mail agents can sometimes cause problems for novice users. By its very design, Majordomo has always lacked a GUI tool to help simplify its usage for non-power users. However, with the World Wide Web providing the perfect foundation, it may be an appropriate time for just such an application.

MajorCool is a CGI application that provides Majordomo mailing list configuration and subscriber management via WWW. Unlike other Web-to-mailinglist interfaces currently available, MajorCool is uniquely different. It supports administrative as well as end-user functions, and, rather than simply generating e-mail requests, interacts directly with Majordomo files wherever possible. Developed for Majordomo 1.93, MajorCool was written in Perl 4.036 and has been successfully implemented under NCSA *httpd* 1.4 and 1.5.


### Introduction

Majordomo[1] is a Perl-based mailing list manager written by Brent Chapman. Chances are, anyone who has ever subscribed to an Internet mailing list has probably used Majordomo to do it. List configuration, subscription management, identifying the members or purpose of a list, or even finding what lists are available are all functions easily provided by commands sent in e-mail to a Majordomo listserver address.

As as easy as Majordomo is to use, there are still numerous opportunities for users to encounter difficulties. Being subscribed under multiple addresses, the word-wrapping behavior of some mail agents, and even the "delayed gratification" nature of the mail-based system often confuse users. And if end-user functions such as *subscribe* and *unsubscribe* are confusing to the Internet novice, imagine the first-time list administrator staring at a complex configuration file that was requested (and must be returned) via Majordomo e-mail commands!

After the introduction of Majordomo at my site, mailing list usage by the technical community grew steadily. Having previously used only standard aliases, the engineers enjoyed the added flexibility provided by Majordomo (open lists, moderation, etc). Word about the benefits of Majordomo eventually spread to the non-technical users, and soon we had administrative assistants and product managers requesting access for purposes of maintaining customer and department lists. Here is where the trouble started.

It was soon discovered that the concept of a syntax-driven command structure was difficult for a large number of people to comprehend. Users much preferred to interact with Majordomo person-to-person

```
Hi there. I am in Sally Smith's
TQM organization and I would
very much like to be a member
of the Web-a-Holics mailing
list. Thank you.
```

rather than be so robotic as to say

```
subscribe webaholics
```

Once briefed on the proper procedures required to control Majordomo, a large percentage of our Microsoft Mail using community were then stymied when they ran into the mailer's annoying tendency to wrap long text lines. Believing an exceptional job was done when they issued the

```
subscribe a-really-long-list-name
```

command, they were frustrated in trying to understand the Majordomo error message returned, and became thoroughly perplexed when we tried to explain the concept of \-escaping long lines in order to bypass MSMail line-wrapping behavior.

These problems were exacerbated 10-fold whenever an MSMail user became administrator of a 300+ line list configuration file.

### Defining The Problem

Clearly, some user-interface improvement was needed to help the non-technical users over this Majordomo usability hump, and a few members of my IS group sat down to figure out exactly what. For our PC users, a GUI tool would be a god-send. We had Microsoft *Visual Basic* and C/C++ development experience in our department, as well as a long history with Web/CGI interfaces. We were open to anything that

[1] *http://www.greatcircle.com/majordomo/*

would allow us to produce a decent Majordomo graphical front-end.

We decided that we needed to support basic administrative tasks as well as end-user functions. Interface support for "subscribe" and "unsubscribe" would satisfy the majority of users, but problems in the list configuration arena were quickly becoming our number one trouble spot, so it would be desirable to address that as well.

We needed something that did not add too much extra administration overhead. Up to this point, we had managed to push most of the list administration responsibility down to the users (to the point where we never had to deal with adding new aliases whenever a list was requested – more on this later). We did not want to introduce a new tool that would needlessly insert us back into the process. Because our lists were in a constant state of flux due to a high list-owner turnover rate, we strongly desired that the GUI be in sync with Majordomo so as to minimize any per-list changes to the interface.

Taking the "in-sync" requirement even further, we felt that it was advantageous that the application interact directly with Majordomo files wherever applicable. Although we were not married to the idea of direct updates, it was felt that at least read-only access to the Majordomo support files was a viable prospect. As yet we were not sure if this would be done via NFS, server co-location, or by some other means.

Regardless of function, a final criterion established was that nothing would be made available via the GUI that was not also accessible through standard e-mail – in other words, the interface could not bypass any Majordomo restrictions or permissions.

By this time (Jan 1996) there were already a few Majordomo/Web interfaces available which we investigated, most notably Patrick Fitzgerald's *MailServ*[2] and David Baker's *LWGate*[3]. While both excellent tools, neither addressed the list configuration issue, our primary concern. In addition, both simply served to construct and send a syntactically correct e-mail message and did not actually provide the user with a display of the requested Majordomo function results. So, for example, any request to display current list members would be initiated on the Web but subsequently viewed via e-mail. We desired something a little more interactive. Lastly, Web interfaces to date were never assumed to be on the same system. This meant that adding a new list to Majordomo also meant updating the configuration of the Web front-end. This was just the sort of sysadmin busy-work we were trying to avoid.

[2] *http://iquest.com/~fitz/www/mailserv/*
[3] *http://www.netspace.org/users/dwb/lwgate.html*

## Designing A Solution

We briefly considered a client/server protocol running on some TCP port, but the thought of *Visual Basic* programming (or similar) just didn't sit right with us. With our local intranet gaining in popularity since its introduction in 1993, we felt instead that a Web-based interface would be the perfect addition to our Majordomo implementation. Our users were comfortable with HTML Form applications, and the multiplatform nature made it equally usable for both PC and UNIX systems.

The more we thought about it, the more we began to embrace the concept of CGI co-location with the mail server because it would make access to required Majordomo files a non-issue. The additional load on the server caused by the single-purpose *httpd* process was expected to be minimal.

One aspect we needed to accommodate was the fact that NCR utilizes a database-driven e-mail address resolution system called *Rolo* to support *First.Last* e-mail addressing. Our Southern California sites were additionally saddled with dual-domain support. Thus, valid addresses for our users could take on many forms:

```
Bill.Houle
Bill.Houle@SanDiegoCA.NCR.COM
Bill.Houle@ElSegundoCA.NCR.COM
bhoule@host.sandiegoca.ncr.com
bhoule
```

To enable MajorCool to understand the equivalencies between these many forms of addresses, the ability to call an external user-lookup mechanism was planned. Rather than perform a straight lexical address comparison to determine if a user was subscribed, the lookup function would return a tuple containing a regular expression pattern which would then be matched against the list addresses. This would enable the application to identify and display all subscribed lists, regardless of which equivalent address a user was subscribed under.

When it came time to design for permissions-protection of the Web interface, we decided to take the lazy approach. Rather than coding to the low-level Majordomo Perl subroutines and dealing with file-locking and other collision issues, we opted for the method utilized by past Web interfaces: e-mail! Our Web tool would interact directly with all Majordomo files for the purpose of querying status, but when it came time to modify any files, we would fall back to the usual mail-based commands. In an effort to "eliminate the middleman", we chose to take advantage of the co-location on the server and pass the simulated mail messages directly to the Majordomo process rather than actually sending the mail.

One final design choice came after testing other Web-to-listserver tools. Most implementations provided the ability to find out details about a list before deciding to subscribe to it. Once ready, the user would then enter an e-mail address and choose to subscribe. However, after browsing another list's details, one had to re-enter the e-mail address for subscribing. We did not like this idea of sign-in at subscription time. We chose to implement the sign-in process before giving access to list information. Not only did this match Majordomo's authentication requirements (where many list query functions are unavailable depending upon the requester address), but it also saved the duplicate entry of subscriber address in a case where multiple lists were selected within a single session.

## Implementing A Solution

Development of MajorCool began by focusing on the user-interface issues of the *subscribe* and *unsubscribe* functions. The first task was to create the interface to the user-lookup program in order to determine subscription status for all possible addresses. By utilizing our *Rolo* database, not only were we able to create a regular expression to match potential subscribed address, but we were able to treat the function as a name-to-address mapping routine. We designed the process to accept a fullname rather than e-mail address and convert that to a list of e-mail patterns. Since *Rolo* supported the concept of nicknames (Rob=Bob=Robert), we were able to generate a regexp of all possible *First.Last* name/nickname permutations as well as the standard *user@host* addresses.

To provide some modularity and allow other sites to implement a different user-lookup function without impacting the rest of the MajorCool code, the user-lookup function was placed in the MajorCool configuration file as a generic *siteaddr(),* function and also included a *%siteaddr* string array to define the HTML prompts, as well as a *by_siteaddr()* function to allow a sort mechanism to be associated with the address types. A sort mechanism allows lists to be displayed, for example, in domain order to make locating a user easier.

Implementing subscribed and unsubscribed lists as HTML checkboxes was straightforward. The difficulties began when one considers the fact that CGI scripts are stateless applications, and we needed to pass the submitted list of checkboxes back to MajorCool for the second phase of subscription processing (and eventual hand-off to Majordomo). We did not want to waste time unsubscribing to previously unsubscribed lists, nor did we wish to re-subscribe to ones already in effect. Therefore, we built a "before and after" state picture of subscription choices and submitted only the differences.

A problem with subscriptions remained: how to deal with Majordomo's concept of "unadvertised" lists. With unadvertised lists, users may be subscribed to lists that they cannot even see on the form, yet we still needed to pass their status to the next phase. To handle this, we used HTML "hidden" fields to maintain the state of unadvertised lists when building the list of checkboxes. This enabled us to keep an accurate view of membership status as the form was submitted to phase two of the subscription processing.

The next obstacle was slow performance. The subscription management screen of Majordomo provides basic list info such as the list name, subscribe policy (closed/open/auto), and list description. All of this information was obtained through the *get_config()* function call provided in the Majordomo libraries. We soon found that invoking *get_config* on any multiple of lists became unbearably slow. While more than adequate for offline mail-based processing, *get_config* was death to an interactive application.

To eliminate the performance problems caused by *get_config,* we simply stopped using it. We developed a "key cache" file that contained all of the information required for a basic subscriber view of each list. Although the cache is a simple flat-file database, it is able to pre-load much faster than multiple calls to *get_config* ever would. The only problem introduced by this design change was the added maintenance of the cache file, but this is easily managed via a cron job set at some reasonable update interval.

In retrospect, the subscription management portion of the user-interface was the most troublesome to implement given *Rolo,* the unadvertised lists, "key cache" workaround, and state maintenance between phases. The remainder of the interface, from password-protected list administration to HTML *textarea* fields for *list.info* and subscriber file management proved relatively easy in comparison.

The next hurdle was converting the user interface selections into Majordomo-style mail commands. Subscription management changes were already supported, but we were still left with the task of modifying the list configuration file. Since only a subset of configuration fields are supported by the interface, we were unable to build and submit a complete configuration file out of the supported fields. This left us little choice but to perform an in-stream modification of the existing config file through string substitutions. Although effective, this limited our support to *variable=value* configuration items only.

Because MajorCool requires read access to all the Majordomo per-list files, the tool needed to operate with the appropriate permissions. The easiest way to accomplish this was through an intermediate shell script "shim" which invokes the real MajorCool program via the suid Majordomo *wrapper.* Thus, MajorCool operates with the same privileges as Majordomo itself.

The use of the *wrapper* program solved one problem, but introduced another. Access was needed to CGI environment variables, yet the *wrapper* is specifically designed to provide a minimum environment space for security reasons. Therefore, in order to support these CGI variables, an additional function of the CGI "shim" is to convert any needed *httpd* environment variables to command-line arguments that the *wrapper*'ed MajorCool could use.

### Configuring MajorCool

From an end-user perspective, the `cgi-bin` "shim" is the heart of MajorCool and therefore should be named appropriately for usage as a URL – for obvious reasons we use `majordomo`. But from a system standpoint, the invoked main script is the true MajorCool workhorse. This script parses its command line for any needed arguments, then reads `/etc/majordomo.cf` and its own configuration file to obtain the basic details about the Majordomo configuration and how the Web interface should operate.

The `majorcool.cf` configuration file defines the basic appearance and behavior of the MajorCool application. Site name, the use of pick-list or type-in field for list administration selection, and *siteaddr*-specific customizations are defined in this file. A sample configuration is included below.

```
$site = "NCR Southern California";
$siteinfo = "www/majorcool.info";
$header = "Page.Header";
$footer = "Prog.Footer";
$hidden_admin = 0;
$hidden_admin_name = "MajorCool";
$hidden_lists = 0;
$hidden_passwd = 1;
```

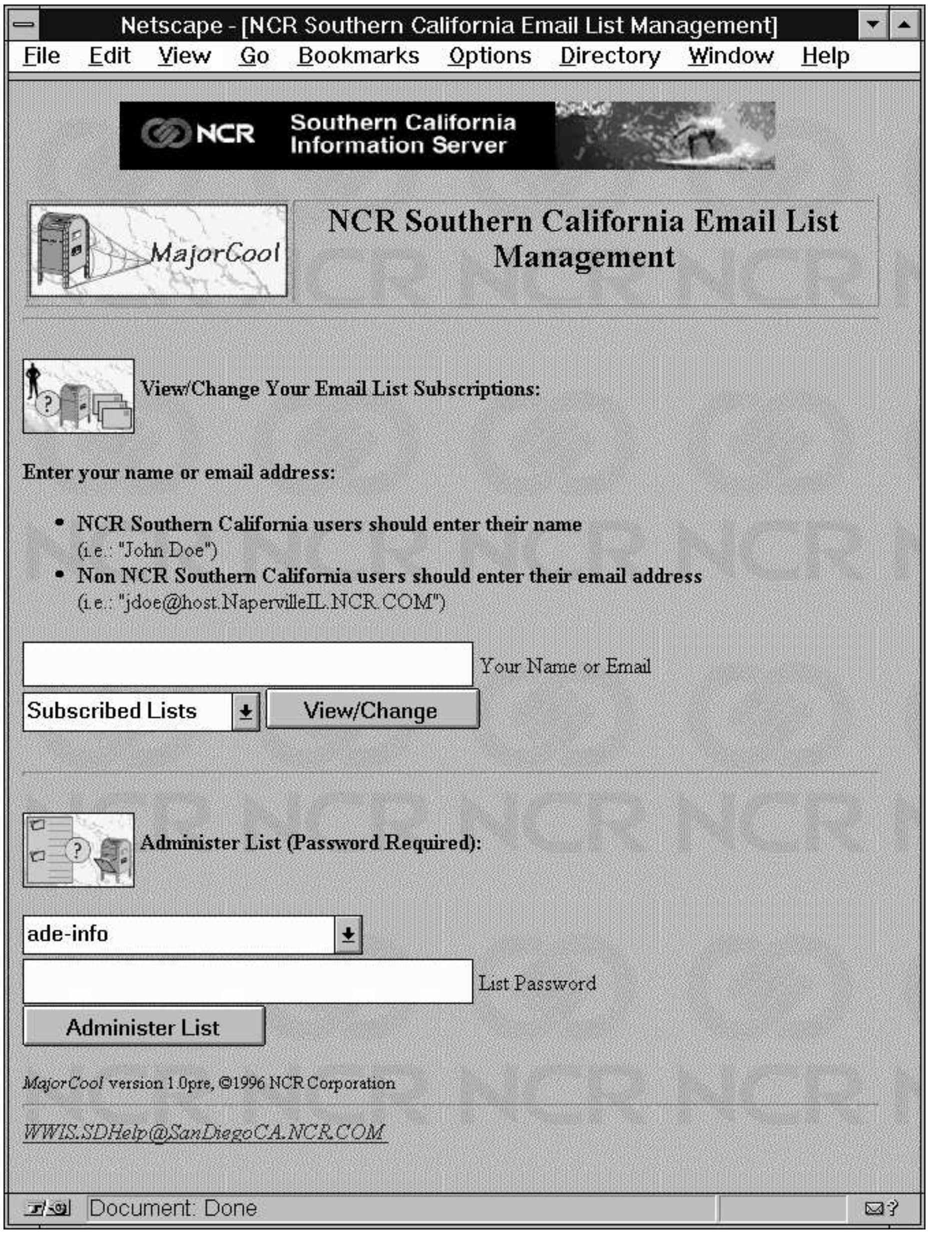


**Figure 1**: MajorCool Opening Screen

```
$log_suffix = "last";
$member_advertised = 0;

###############################################
# This "siteaddr" LOOKUP section defines local
# address customizations and prompts.
###############################################

%siteaddr = (
    'prompt',    "Your Name or Email",
    'intro',     "<B>Enter your name or email address:</B>
    [....omitted....]
);

sub siteaddr {
    local($target) = @_;
    #
    # NCR employees may be subscribed to lists as:
    #   First.Last
    #   First.Last@Domain
    #   login@Domain
    #   login@host.Domain
    #
    # This implementation example uses a
    # 'ROLO' tool to map
    [....omitted....]
    return ($user, $preferred_addr, $regexp);
}

sub by_siteaddr {
    [....omitted....]
    $x cmp $y;
}
```

## Using MajorCool

MajorCool is invoked through the `cgi-bin` URL. It can be configured to run separately as end-user and list-administrator functions, or combined in a single screen interface. The screen capture shown in Figure 1 is the unified version.

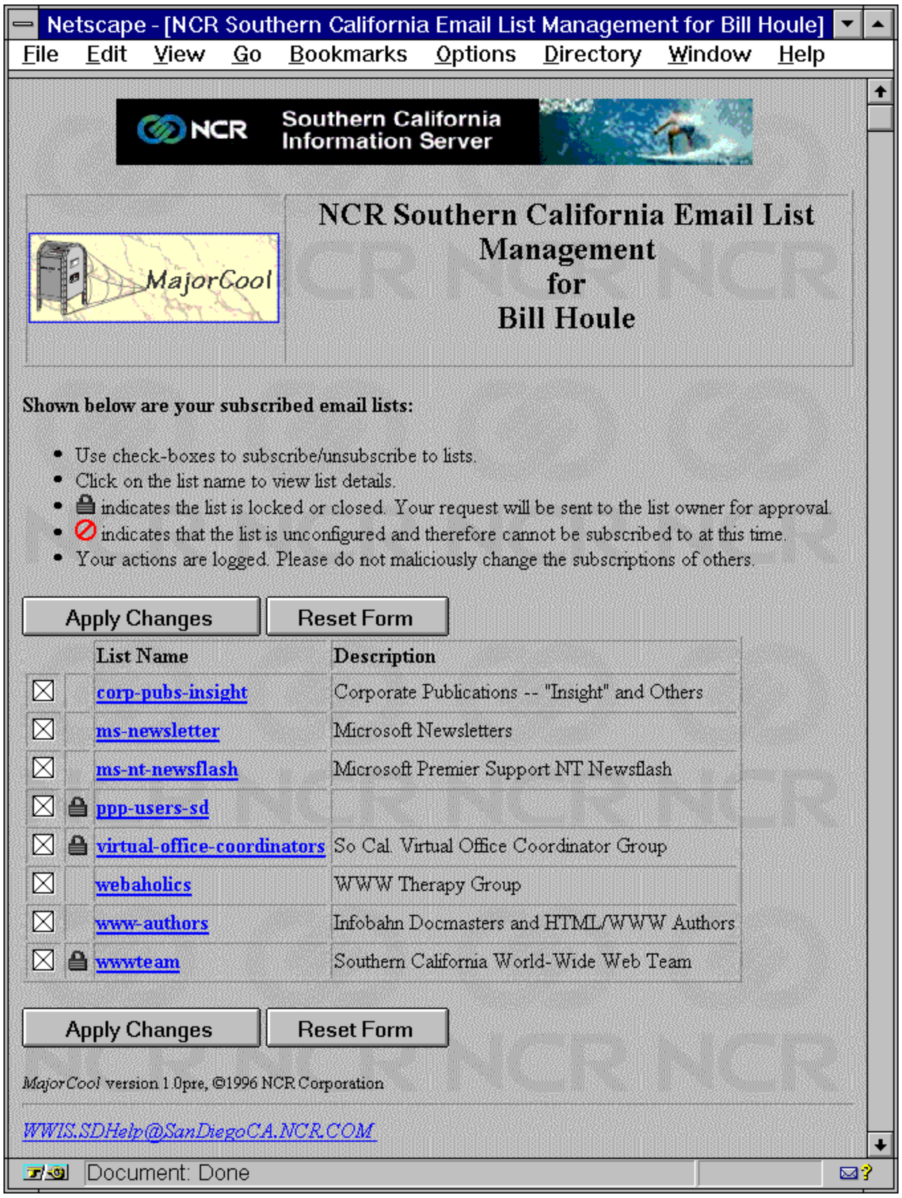


**Figure 2**: Checkbox subscription display

All displays incorporate a configurable header and footer to allow a site to maintain a common look-and-feel between the MajorCool application and the rest of the intranet.

After establishing an e-mail address, the end-user screen displays all subscribed lists (or alternately, all available lists) in the form of checkboxes; see Figure 2. Users may toggle between subscribe and unsubscribe status by changing the checkbox selections, and upon submission, the appropriate commands are issued directly to Majordomo (bypassing *sendmail*).

The description of the lists are shown in the display, with further details available by clicking on the hyperlink of the list name. Traveling to the list details screen will allow access to information such as the *list.info* file, number of subscribers, subscriber list (if permitted) and more; see Figure 3.

Access to the list administrator functions are protected by the standard Majordomo per-list passwords. Once the password has been accepted, the list administrator is presented with a screen of some of the more commonly-modified aspects of Majordomo *list.config* file management; see Figure 4. As before, additional details are available by selecting the active hyperlinks. Through these links, one may also view or modify the *list.info* file, as well as see/alter the subscriber list itself.

Just as with end-user subscription changes, when the modifications to the list configuration are submitted, the changes are converted to standard Majordomo command syntax and handed to Majordomo without using e-mail as the intermediary. Unlike the subscription changes (which operate with the name & address of the end-user), list-configuration changes may be submitted with a configurable address. This is useful when multiple administrators manage the list and actions need to be submitted as other than the default list owner.

You can see that a pattern has developed: all "view" operations are supported directly through the MajorCool interface. However, any "change" operations are converted into equivalent Majordomo commands and handed to the Majordomo/*wrapper* duo for processing. Upon successful completion, the user will receive normal mail confirmation just as if the commands were issued via e-mail. As much as it might have been nice to edit files directly rather than submitting to Majordomo, this method certainly proved safer and infinitely quicker to implement.

## Experiences with Majordomo

I believe the Internet community owes a great debt of gratitude to Brent Chapman and the entire Majordomo-workers community. The Majordomo product is well-supported, easily extensible, and most

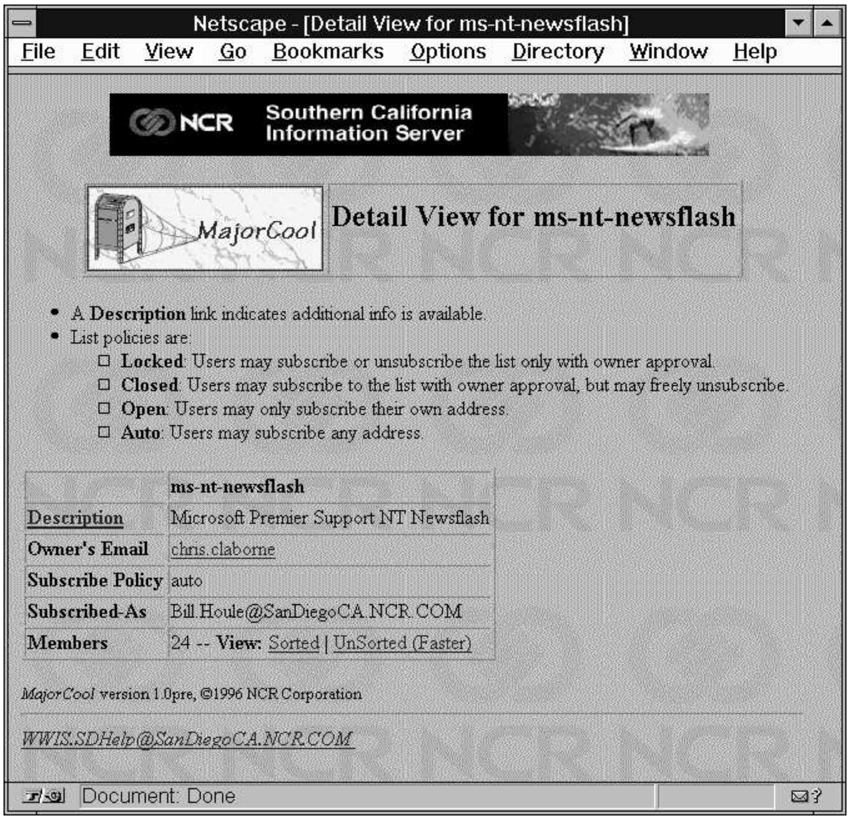


**Figure 3**: List details

importantly, widely recognized and accepted by end-users. Brent's initial work was featured at the *LISA 6*[1] conference.

As evidence of Majordomo's extensibility, the fact that MajorCool was not implemented on a "plain vanilla" Majordomo installation needs to be explained.

Over the course of Majordomo implementation at our site, we have made numerous minor enhancements to the default distribution in order to meet the requirements of our users. Such minor enhancements include the addition of new keywords in the list configuration files that would, for example, control whether or not list names are included in the MSMail Global Address List or whether the sending of the "Welcome" message to new subscribers was On or Off.

Support for these options are included in the MajorCool release, but are not required and can easily be removed. However, there are two additions that are important to the operation of MajorCool, and a slight amount of re-work would be required to make it operate on a default Majordomo distribution.

The first major change is the use of an *owner* keyword in the configuration file. As I mentioned earlier, our lists have a high degree of entropy due to changing job functions and department roles. We found that list owners changed relatively frequently, and we were not thrilled with the idea of constantly tweaking aliases whenever someone changed jobs.

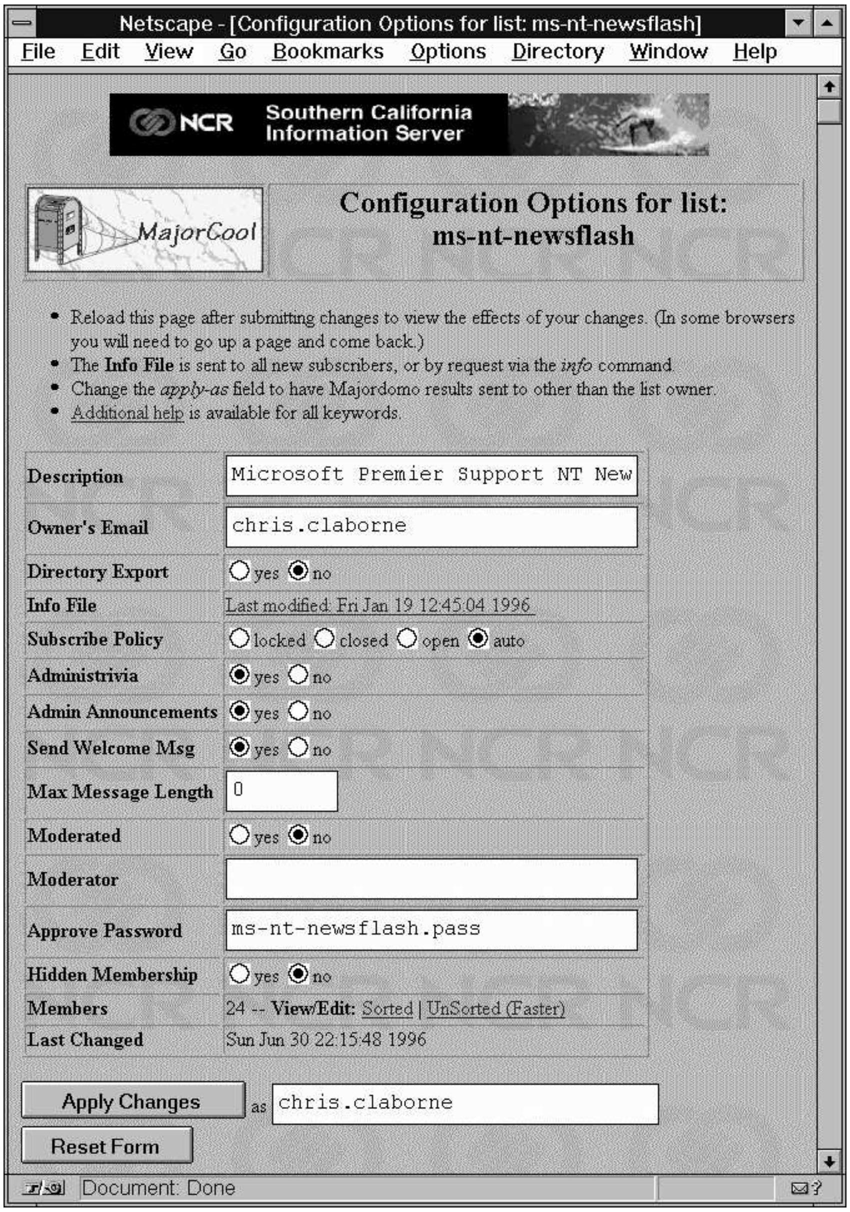


**Figure 4**: List configuration screen

Our best defense was to push that ownership responsibility back down to the users. By implementing an *owner* keyword (which is a reference to a valid e-mail address), list admins could re-assign list ownership to another simply by modifying their configuration file. A cron job on the system regularly parses all config files to build a *sendmail* alias file from the entries. This additional feature makes our sysadmin life easier because it removes us completely from the business of changing list owners. Everything is automated and hands-off. Although the enhancement is not critical to MajorCool itself, there is nonetheless liberal use of *$config_opts{'owner'}* in the code that would have to be changed if this keyword were not present.

The other Majordomo modification was required to support the full-text editing of the subscriber list on the list administration screen. It is actually another case of laziness rearing it's ugly head. We did not feel motivated enough to implement a *diff*(1) routine within MajorCool to process changes to the subscriber list, so we instead implemented a *newwho* Majordomo command to provide wholesale replacement of the list. The command, similar to *newinfo* (with almost identical code), simply replaces the current list with the input provided. The in-line list editing function will not work without this added function. A side-effect is that list members are not greeted with the usual Welcome message when added using *newwho.* For this reason, an *approve* sub/unsub mechanism is also provided on the admin form for times when adding individual users is preferred.

## Experiences with MajorCool

The bulk of MajorCool took approximately one month of part-time work to implement. Naturally, it has undergone a steady stream of revisions as bugs were fixed and new features were added. A minor rewrite phase required an additional two weeks as the code was made more configurable and less NCR-specific in preparation for release back to the Majordomo community. MajorCool is now approaching 1600 lines of code.

During beta-testing of the product, a list administrator remarked that the tool was "majorly cool". Given that we were still seeking to name the application, MajorCool seemed an appropriate choice.

Majordomo/MajorCool in NCR is currently supporting 105 mailing lists, the largest of which contains over 2000 subscribers. We support an average of 650 "change" requests per month, 30% of which are made via MajorCool. A significant data point is the fact that, using the number of requests-per-month as an indicator, overall Majordomo awareness has increased by 250% since the announcement of MajorCool availability in March. MajorCool, in essence, seemed to help promote Majordomo.

User reaction to the MajorCool tool has been marked by a distinct lack of "help me" mail sent to the `majordomo-owner` address; for this reason alone I consider MajorCool a success. Since replacing the `majordomo-owner` alias with a canned response referring people to the MajorCool URL, e-mail to the alternate "human" address has dwindled down to nothing.

MajorCool as a GUI tool extends the functionality of Majordomo. Most notably, its unique features address two of the problem areas previously identified by Brent in his original *LISA* paper. The externalized user-lookup function (the NCR *Rolo,* in this case) corrects the problem of identifying duplicate address formats while the Web-based nature of the interface eliminates line-wrapping problems caused by other (mostly PC) mailers. Finally, the administration capabilities help simplify the process of list management.

## Future Work

As of this writing, we are on the cusp of a Majordomo 1.94 final release, and MajorCool has not been tested against the new version. First and foremost in the way of future work would be this undertaking.

Another much needed addition is the support for string array variables in the list configuration mode. You'll recall that a new configuration file is submitted by performing an in-stream string substitution on the var/value pairs. The multi-line string array data type does not fit this regexp pattern which is unfortunate because support for such array variables as *message_fronter* and *advertise/noadvertise* are sorely missed.

Archive support would also be a welcome enhancement to the MajorCool command suite. Provision for the Majordomo *get* command immediately comes to mind, but the idea of list archive management can be taken even further if support for Web-based archive browsers such as Kevin Hughes' *Hypermail* were added.

## Acknowledgements

I would like to acknowledge the contribution of Matt Bateman, who was involved in the initial design and coding of MajorCool prior to his leaving NCR.

## Availability

MajorCool was developed at the NCR Corporation with the full intent to release it back to the Internet community. For my own reasons of convenience, it is available for download via World Wide Web access at *http://www.DigitalMasters.COM/˜bhoule/* rather than from the NCR external site. It is freely modifiable and distributable provided all copyrights are maintained.

It is hoped that other Majordomo users will benefit from the MajorCool release and install it at their sites. From this, we can build upon the base of the `Majordomo-workers` and `Majordomo-`

`users` support groups. I will be monitoring these lists in support of the MajorCool application, and I encourage others to enhance the product further to make it truly useful for everyone.

### Author Information

Following his Math/Computer Science & Cognitive Science undergraduate work at the *University of California – San Diego,* Bill Houle has spent eight years at the NCR Corporation. Working variously in Software Engineering and support, he has spent the last five years in the IS area, currently as the *Information Access* Team Leader. In his present position, he is responsible for Internet services (E-Mail, News, World Wide Web, etc) at the two Engineering facilities in Southern California. He can be reached via U. S. Mail at NCR Corporation; 17095 Via Del Campo San Diego, CA 92127. Reach him electronically at Bill.Houle@SanDiegoCA.NCR.COM .

### References

[1] Chapman, D. Brent, "Majordomo: How I Manage 17 Mailing Lists Without Answering 'request-' Mail", *Proceedings of the Sixth Systems Administration Conference (LISA VI),* 1992, pp 135-143.