Customization Guide

Sun™ ONE Messenger Express

Version 6.0

816-6743-10
December 2003

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.

Contents

About This Guide

This guide explains how to customize the look and feel of Sun™ Open Net Environment (Sun ONE) Messenger Express. Although the product architecture permits an almost unlimited customization of the “static” portion of the pages served by the Messenger Express HTTP daemon, this guide focuses on how to perform the most commonly requested customizations. In addition, the customizations have been tied together into an application scenario so that examples, code, screen shots, all relate to one another and provide a common frame of reference.

Topics covered in this chapter include:

Who Should Read This Book

You should read this book if you are responsible for administering, configuring, and customizing Messenger Express at your site. Developers may also find this guide useful for reference.

What You Need to Know

This book assumes a knowledge of Sun ONE Messaging Server software and an understanding of the following:

- JavaScript™
- HTML
- Email applications
- Web development

How This Book is Organized

This book contains the following chapters:

- About This Guide (this chapter)
- Chapter 1, “Introduction to Messenger Express and Customization”

  This chapter provides a high-level overview of how to customize the look and feel of Sun ONE Messenger Express.
- Chapter 2, “Customizing General Features”

  This chapter explains how to customize the general features of Sun ONE Messenger Express.
- Chapter 3, “Customizing User Interface Features”

  This chapter explains how to customize the Sun ONE Messenger Express user interface.
- Chapter 4, “Customizing Advanced Features”

  This chapter discusses advanced customization techniques.
- Chapter 5, “Managing Authentication to the Messenger Express Service”

  This chapter describes how to integrate other authentication mechanisms with Sun ONE Messenger Express.

Document Conventions

Monospaced Font

`Monospaced font` is used for any text that appears on the computer screen or text that you should type. It is also used for file names, distinguished names, functions, and examples.

Bold Monospaced Font

`Bold monospaced font` is used to represent text within a code example that you should type. For example, you might see something like this:

```
./installer
```

In this example, **`./installer`** is what you would type at the command line.

Italicized Font

Italicized font is used to represent text that you enter using information that is unique to your installation (for example, variables). It is used for server paths, names.

For example, throughout this document you will see path references of the form:

msg_svr_base/. . .

The Messaging Server Base (*msg_svr_base*) represents the directory path in which you install the server. The default value of the *msg_svr_base* is `/opt/SUNWmsgsr`.

Italicized font is also used for variables within the synopsis of a command line utility. For example, the synopsis for the `commadmin admin remove` command is:

```
commadmin admin remove -D login -l userid -n domain -w password [-d domain]
     [-h] [-i inputfile] [-p port] [-X host] [-s] [-v]
```

In the above example, the italicized words are arguments for their associated option. For example, in the `-w` *password* option, you would substitute the Administrator's password for *password* when you enter the `commadmin admin remove` command.

Square or Straight Brackets

Square (or straight) brackets `[]` are used to enclose optional parameters. For example, in the installation guide you will see the usage for the `installer` command described as follows:

`./installer` *[options] [arguments]*

It is possible to run the `installer` command by itself as follows to start the Messaging Server installation:

```
./installer
```

However, the presence of *[options]* and *[arguments]* indicate that there are additional optional parameters that may be added to the `installer` command. For example, you could use `installer` command with the `-b` option to specify the *msg_svr_base* prior to running the installation program:

```
./installer -b /opt/SUNWmsgsr
```

Command-Line Prompts

Command-line prompts (for example, `%` for a C-Shell, or `$` for a Korn or Bourne shell) are not displayed in the examples. Depending on which operating system environment you are using, you will see a variety of different command-line prompts. However, you should enter the command as it appears in the document unless specifically noted otherwise.

Platform-specific Syntax

Also, all paths specified in this manual are in Unix format. If you are using a Windows NT-based Sun ONE Messenger Express, you should assume the Windows NT equivalent file paths whenever Unix file paths are shown in this book.

Where to Find Related Information

In addition to this guide, Sun ONE Messaging Server comes with supplementary information for administrators as well as documentation for end users and developers. Use the following URL to see all the Messaging Server documentation:

`http://docs.sun.com/db/prod/s1msgsrv`

Where to Find This Book Online

You can view this documentation online in PDF and HTML formats by pointing your browser to the following URL:

`http://docs.sun.com/db/prod/s1msgsrv`

Chapter 1

Introduction to Messenger Express and Customization

Messenger Express is a web-based electronic mail program that enables end users to access their mailboxes using a browser. Messenger Express clients send mail to a specialized web server that is part of Sun ONE Messaging Server. The HTTP service then sends the message to the local Message Transfer Agent (MTA) or to a remote MTA for routing or delivery.

Almost all features of Messenger Express are fully customizable. This guide explains the features that are customizable. Most features can also be customized easily during an upgrade.

This chapter contains the following sections:

Messenger Express Customization Overview

Messenger Express lets you rewrite the “static” portion of the pages served by the Messenger Express HTTP daemon to produce a fully customized webmail service. Messenger Express supports both JavaScript and HTML in implementing customization schemes.

Messenger Express Components

Sun ONE Messenger Express consists of two components, the client and the server. The client reads and interprets the JavaScript language. The HTTP server understands proprietary protocols that communicate with Messenger Express. The JavaScript files reside on the server and are downloaded to the client. The client extracts data from the JavaScript code to customize Messenger Express functions. All modifications and customizations are done on the server.

HTML files contain both text and markup describing how the text is formatted and handled. Markup is implemented through a set of tags, which specify things like headers, indents, font size, italics and so on. These are largely static tags that deal exclusively with text within the HTML file on the client. However, the HTML also contains dynamic tags, such as JavaScript embedded in the client file, that point to files and functions on the server. The dynamic tags enable the HTML file client to pull in data processed on the server for use in otherwise static Web pages.

HTML files provide the skeleton structure of each of the interfaces, whereas JavaScript files give the specific attributes. Each of the JavaScript main functions are contained within an HTML parent function. There are differentiations within the JavaScript files themselves. The file `main.js` primarily controls the layout of the interfaces, whereas the file `i18n.js` controls the text. The `i18n.js` file also can be localized to fit the language of many regions in the world.

The HTML files are the `parent` functions that call the `main` functions in the JavaScript files to initiate actions.

Location of Customizable Files

The Messenger Express JavaScript and HTML files that can be customized reside in the *msg_svr_base*`/html directory`, where *msg_svr_base* represents the directory path in which you install the Messaging Server software. The server needs to be restarted for any changes in the HTML and JavaScript files to take effect.

Table 1-1 lists the files to be edited to customize Messenger Express, and which part of Messenger Express each file controls.

Table 1-1 Sun ONE Messenger Express Customizable Files

Files	What the File Controls in Messenger Express
`main.js`	Layout of the UI
lang`/i18n.js`	Text on the UI
`mbox_fs.html`	Mailbox management portion of the UI

Table 1-1 Sun ONE Messenger Express Customizable Files *(Continued)*

Files	What the File Controls in Messenger Express
`msg_fs.html`	Message management portion of the UI
`fldr_fs.html`	Folder management portion of the UI
`opts_fs.html`	Option management portion of the UI
`comp_fs.html`	Message composition
lang`/default.html`	Login screen
`lookup_fs.html`	Address search
`attach_fs.html`	Attachments
`collect_fs.html`	Collection of mail from another server
`receipt_fs.html`	Return receipt
`subscribe_fs.html`	Subscribe Folders
`emoticons.html`	Alter emoticons

Messenger Express Localization

You can localize any feature of Messenger Express. You can create different pages for different languages. When you create language-specific static webmail pages, you group them in subdirectories under the main document directory. The webmail code automatically detects the client's language preference and fetches the webmail pages from the appropriate subdirectory.

When you change common sections of the static webmail pages, you must make the changes on multiple occasions if modifications are desired across languages (for example, to JavaScript behavior). Conversely, you can make language-specific modifications selectively throughout the application.

Specific Locales

Table 1-2 lists the specific locales and their abbreviations that Sun ONE Messenger Express services. The protocol's default language is English.

Table 1-2 Messenger Express Specific Locales

Locale	Abbreviation
English	en

Table 1-2 Messenger Express Specific Locales *(Continued)*

Locale	Abbreviation
Japanese	ja
Spanish	es
French	fr
German	de
Russian	ru
Arabic	ar

Location of Locale-Specific Customizable Files

The localized Messenger Express JavaScript and HTML files reside in the *msg_svr_base*/html/*locale_specific* directory, where *msg_svr_base* represents the directory path in which the Messaging Server software is installed.

Basic Interfaces and Associated Functions

This section presents the functions associated with various Messenger Express screens, including:

- Inbox screen
- Message screen
- Folders screen
- Options screen
- Composition window
- Message Search window

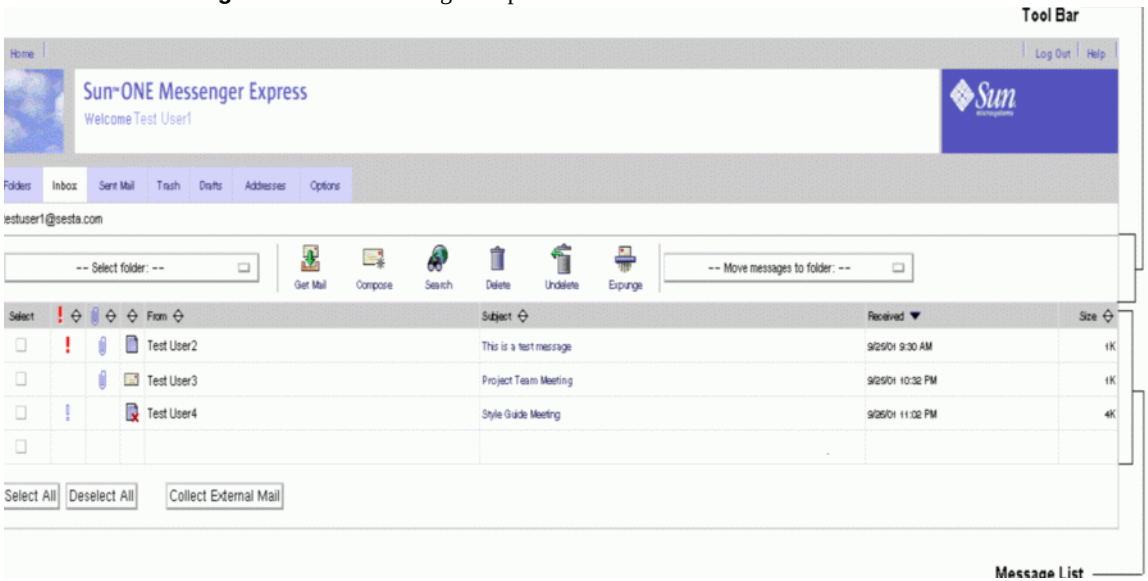
The tables list the functions associated with each button on Messenger Express screens.

The files containing the functions reside in the *msg_svr_base*/html directory, where *msg_svr_base* represents the directory path in which the Message Server software is installed.

Inbox Screen

The Messenger Express Inbox screen, shown in Figure 1-1, enables you to view all messages and its basic features— for example, subject, from, received and size. The Inbox screen gets new messages and enables you to search for or delete old messages, as well as move messages into other folders.

Figure 1-1 Messenger Express Inbox Screen



Inbox Screen Functions

Table 1-3 lists the functions needed to customize the Inbox screen, including `main` functions (found in `main.js`) and `parent` functions (found in `mbox_fs.html`).

Table 1-3 Inbox Screen Functions

Item	Function
Folders	`main.displayFolders()`
Inbox	`main.displaySpecialMbox('Inbox')`
Sent Mail	`main.displaySpecialMbox('Sent')`
Trash	`main.displaySpecialMbox('Trash')`
Drafts	`main.displaySpecialMbox('Drafts')`
Addresses	`main.displayPab()`

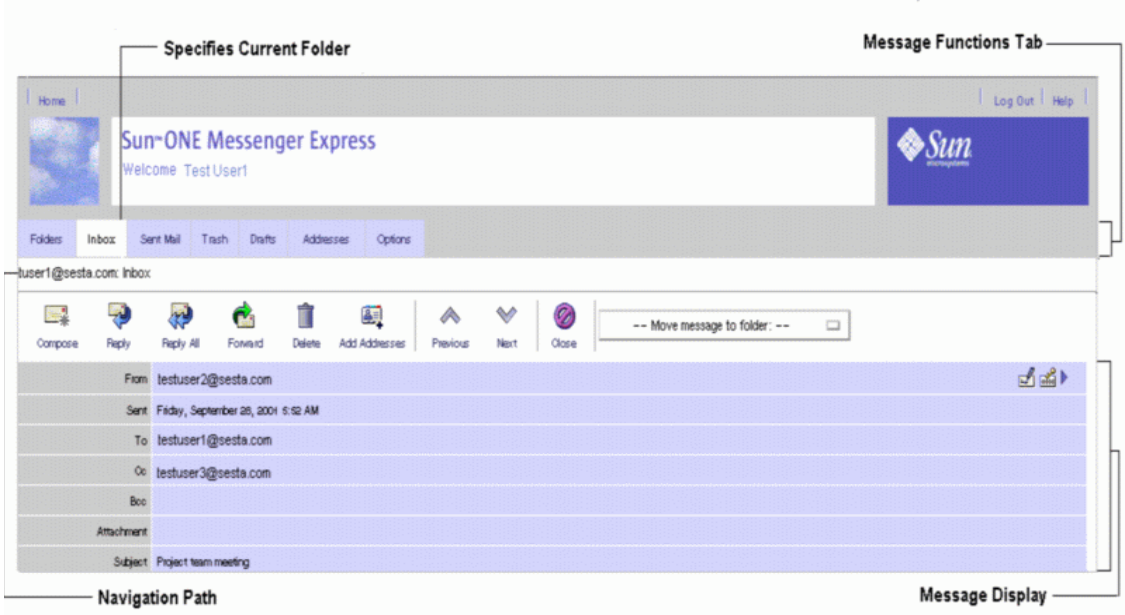
Table 1-3 Inbox Screen Functions *(Continued)*

Item	Function
Options	`main.selectOptions()`
Help	`main.help()`
Logout	`main.logout()`
Get Mail	`main.check_mail = 1;` `main.displaySpecialMbox('Inbox')`
Compose	`main.compose('new')`
Search	`parent.srch()`
Expunge	parent.exmsg()
Move Messages to Folder	`parent.move()`
Delete and Undelete	`parent.delmsg()`
Collect External Mail	`main.collect()`

Message Screen

The Messenger Express Message screen, shown in Figure 1-2, displays the message selected from the Inbox screen. The Message screen gives the option of replying to the sender, forwarding the message, moving the message to a different folder, or deleting the message. The Message screen also enables navigation to the next or previous message.

Figure 1-2 Messenger Express Message Screen



Message Screen Functions

Table 1-4 lists the functions needed to customize the Message screen, including `main` functions found in `main.js` and `parent` functions found in `msg_fs.html`.

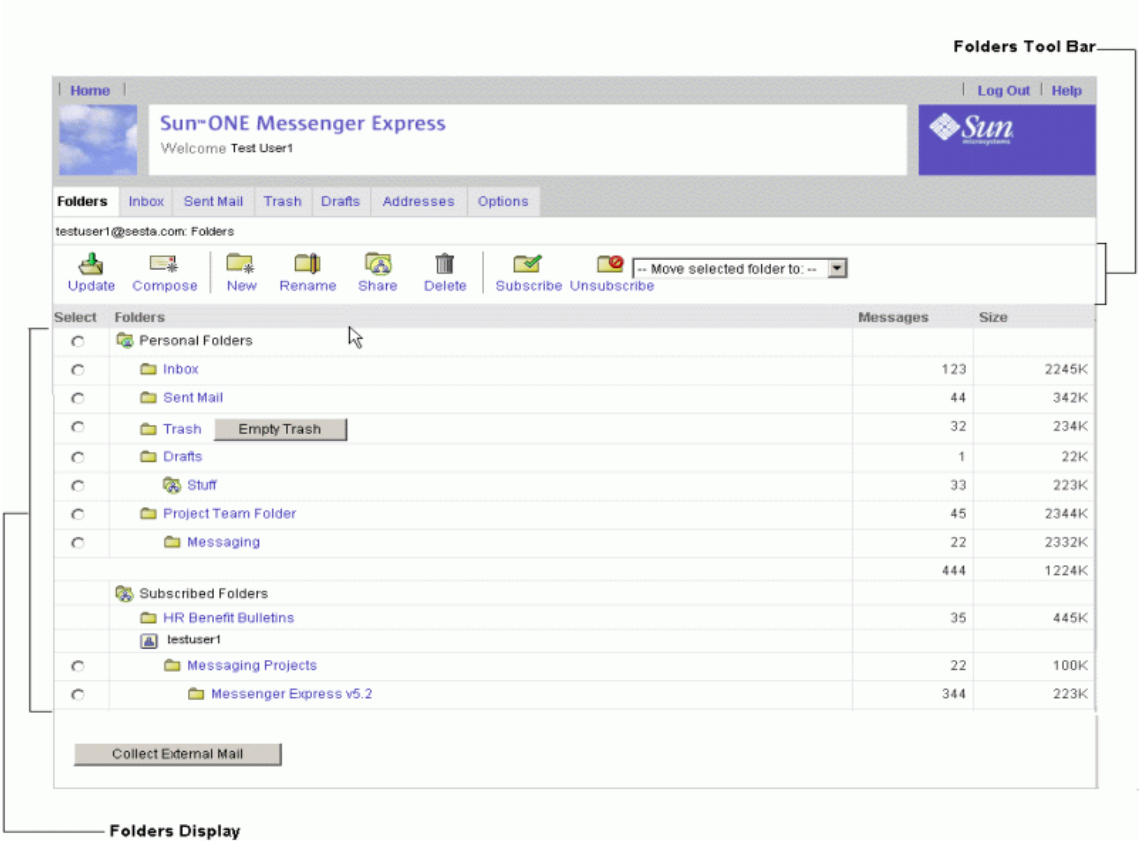
Table 1-4 Message Screen Functions

Item	Function
Compose	`main.compose('new')`
Reply	`main.compose('reply')`
Reply All	`main.compose('replyall')`
Forward	`main.compose('forward')`
Delete and Undelete	`parent.delmsg()`
Add Addresses	`parent.addAllAddresses()`
Previous	`parent.prev()`
Next	`parent.next()`
Close	`parent.gotofolder()`

Folders Screen

The Messenger Express Folders screen, shown in Figure 1-3, displays all folders that can be accessed. The Folders screen lists the number of messages contained and the size of each folder. The Folders screen also enables creating new folders, renaming or deleting old ones, subscribing or unsubscribing shared folder, sharing folder, moving a folder within another folder, updating the inbox, and composing new messages. Like the Inbox screen, the Folders screen also enables collection of external mails.

Figure 1-3 Messenger Express Folders Screen



Folders Screen Functions

Table 1-5 lists the functions needed to customize the Folders screen, including `main` functions found in `main.js` and `parent` functions found in `fldr_fs.html`.

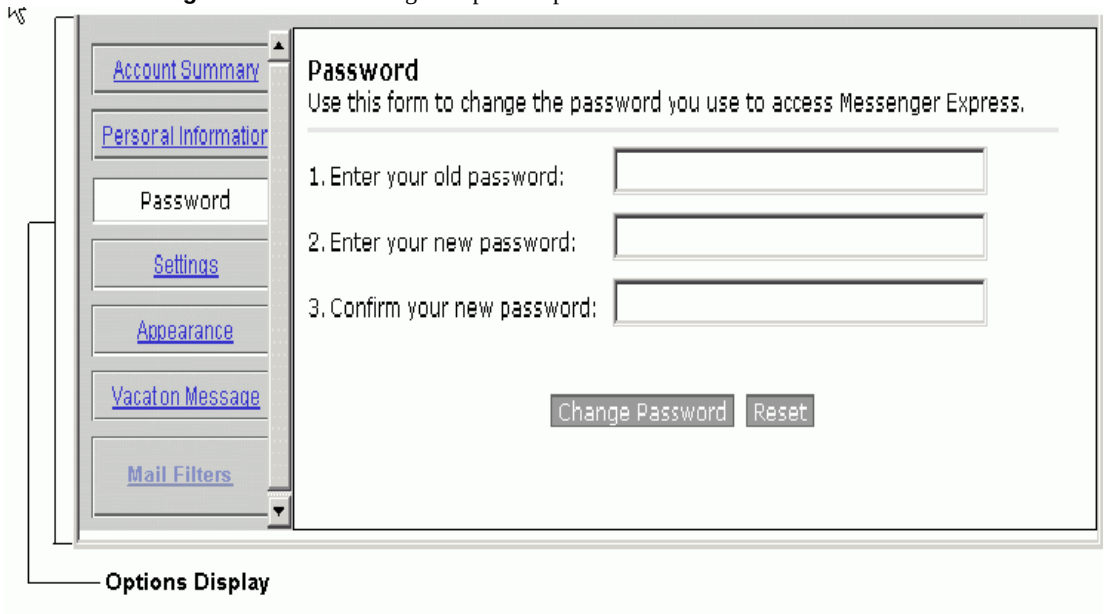
Table 1-5 Folders Screen Functions

Item	Function
Update	`main.refreshFolders()`
Compose	`main.compose('new')`
New	`parent.addFolder()`
Rename	`parent.renFolder()`
Share	`parent.setfolder()`
Delete	`parent.delFolder()`
Subscribe	`main.subscribeFolder()`
Unsubscribe	`main.unsubscribeFolder()`
Move Folder	`parent.moveFolder(options[selectedIndex].value)`

Options Screen

The Messenger Express Options screen, shown in Figure 1-4, enables access to the subscriber's account summary, personal information, password, settings, appearance, vacation message, and mail filters, all of which can be customized.

Figure 1-4 Messenger Express Options Screen



Options Screen Functions

Table 1-6 lists the `parent.` functions found in `opts_fs.html` needed to customize the Options screen.

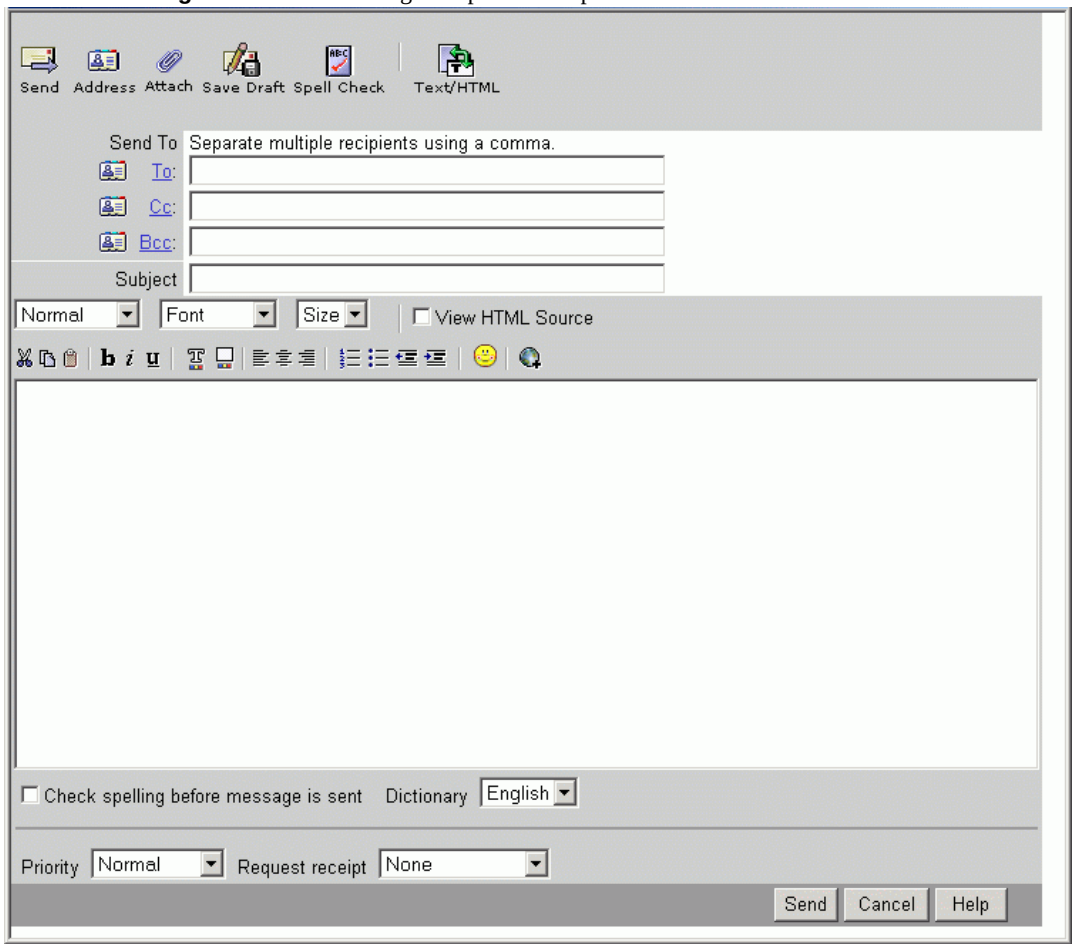
Table 1-6 Options Screen Functions

Item	Function
Account Summary	`parent.toggle('summary')`
Personal Information	`parent.toggle('personal')`
Password	`parent.toggle('password')`
Settings	`parent.toggle('settings')`
Appearance	`parent.toggle('appearance')`
Vacation Message	`parent.toggle('vacation')`
External Mail	`parent.toggle('main.collect')`
Mail Filters	`parent.toggle('mailFilters')`

Composition Window

The Messenger Express Composition window, shown in Figure 1-5, is used primarily to compose a new message. You can also use the window to save a draft or attach a file to the message, look up a recipient in the address book, access the help file, and cancel the composition altogether. Mail recipients can be added in “To”, “Cc”, or “Bcc” fields. You can edit the message in Text or HTML format if you are using Internet Explorer and this feature is not supported in Netscape Navigator. In the Composition window you can also check the spelling. The Composition window also enables you to set the mail priority or request for a return receipt.

Figure 1-5 Messenger Express Composition Window



Composition Window Functions

Table 1-7 lists the functions needed to customize the Composition window, including `main` functions (found in `main.js`) and `parent` functions (found in `comp_fs.html`).

Table 1-7 Composition Window Functions

Item	Function
Send	`parent.send('smtp')`
Address	`parent.lookup()`
Attach	`main.attach()`
Save Draft	`parent.send('draft')`
Spell Check	`parent.spellck()`
Help	`main.help(1007399)`
Cancel	`parent.cancel()`
To / Cc / Bcc	`parent.lookup()`
Text / HTML	`parent.switchEditor()`

Message Search Window

The Messenger Express Message Search window, shown in Figure 1-6, is primarily used to search messages by entering sender's name, subject, body text, or recepient's name. You can also delete the messages from the search list.

Figure 1-6 Message Search Window

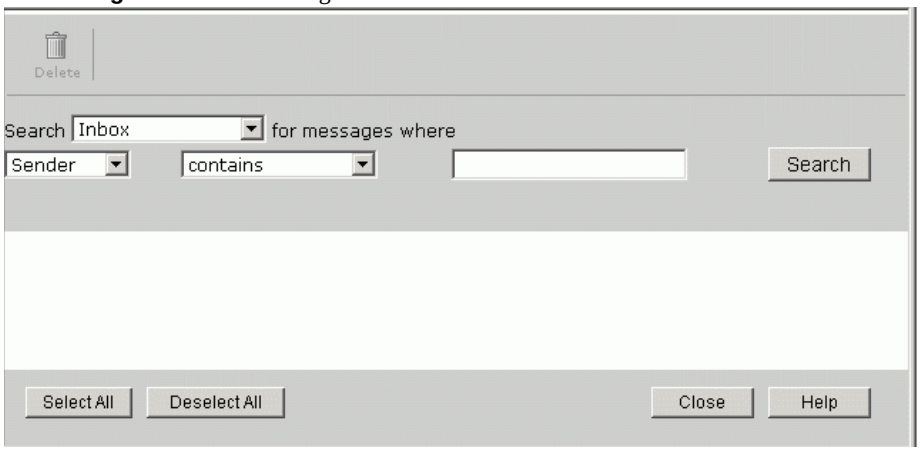


Table 1-8 lists the functions needed to customize the Message Search window, including `main` functions (found in `main.js`) and `parent` functions (found in `searchMessage.html`).

Table 1-8 Message Search Window Functions

Item	Function
Select All	`parent.selectAll(what)`
Deselect All	`parent.selectAll(what)`
Help	`main.help()`
Close	`parent.close()`
Delete	`parent.delmsg()`
Search	`parent.doSearch()`
Undelete	`parent.undelmsg()`

Chapter 2

Customizing General Features

This chapter describes how to customize the general features of Messenger Express.

This chapter contains the following sections:

Modifying the Login Screen

This section describes how to modify the Messenger Express Login screen shown in Figure 2-1.

Figure 2-1 Messenger Express Login Screen.

You can modify the following on the Messenger Express Login screen:

- Replace the logo with a custom graphic
- Change the color scheme
- Replace the service name (for example, Messenger Express)
- Replace images with advertisement banners and links

To Modify the Login Screen

To modify the Login Screen, edit the *lang*/`default.html` file.

Customize the user interface by editing the body of *lang*/`default.html`. Functionally, *lang*/`default.html` contains three forms, two visible and one hidden:

- Username Form (visible)
- Password, Login Button (visible)
- Login Same Window Form (hidden)

The hidden form is the one that is submitted to the server (`POST` username and password to `login.msc`).

You can also insert a new banner, image or link on the Messenger Express Login screen, by editing the *lang*`/default.html` file.

NOTE *lang* is the language specific file that you need to edit.

Example—Login Screen Modifications

The example shown in Figure 2-2 replaces the Sun ONE logo with a custom graphic and adds an advertisement banner with a link.

Figure 2-2 Example Login Screen Modifications

Code Example 2-1 shows the Login screen HTML before editing the `en/default.html` file to customize the Login screen.

Code Example 2-1 Before Altering Login Screen Features

```
....
<meta http-equiv="Content-Type" content="text/html; ">
<link rel="stylesheet" href="master-style.css" type="text/css">
....
<BODY marginwidth="0" marginheight="0" topmargin="0"
leftmargin="0"
rightmargin="0" bottommargin="0" bgcolor="#800000">
....
<td rowspan="4" valign="bottom"><img src="imx/sun_logo.gif"
width="96" height="66"></td>
....
<td><img src="imx/WebMail_splash.gif" width="200" height="30"
alt="Sun ONE Messaging Server"></td>
....
<td bgcolor="#cccccc">
....
```

Code Example 2-2 shows the changes to be made in the `en/default.html` file to replace the Sun ONE logo with a custom graphic and add an advertisement banner with a link in the Login screen.

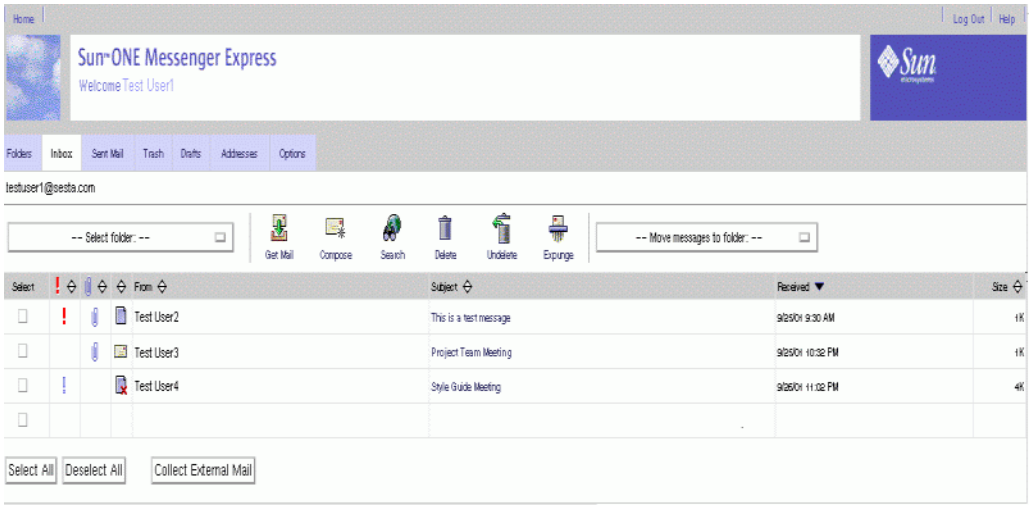
Code Example 2-2 After Altering Login Screen Features

```
....
<meta http-equiv="Content-Type" content="text/html; ">
<link rel="stylesheet" href="master-style.css" type="text/css">
....
<BODY marginwidth="0" marginheight="0" topmargin="0"
leftmargin="0"
rightmargin="0" bottommargin="0" bgcolor="#ffffff">
....
<td rowspan="4" valign="bottom"><img src="imx/siroe.gif"
width="96" height="66"></td>
....
<td><a href=”http://www.siroe.com”>
<img src="imx/ad.gif" width="450" height="50"
alt="Advertisement Banner tip"></a><BR>
<img src="imx/siroewebservice.gif" width="200" height="30"
alt="Siroe Webmail Service">
.....
<td bgcolor="#C0C0C0">
....
```

Modifying Color Sets

This section describes how to modify the Messenger Express user interface color sets shown in Figure 2-3.

Figure 2-3 Messenger Express Color Sets



You can customize the default color sets for the Messenger Express user interface to change items such as the title bar, tab outlines, and column headers.

To Modify Color Sets in the User Interface

To modify color sets in the user interface, edit the `ui[] array` definitions near the top of the `main.js` file.

The function `refreshColorSet()` in `main.js` sets the color scheme of the user interface to color values such as `chrome1` and `accent2`. These color values are used by the rest of the display functions in `main.js`.

See `refreshColorSet()` in `main.js` for the translation of the `ui[]` elements to the color values.

The `ui[] array` can have as many rows as desired. Additional color themes are displayed on the preferences page as new rows are defined in `main.js`. The user's JavaScript application will not start when the rows are deleted from the definition script with the user preferences still pointing to a higher color table index than exists in the `ui[] array`.

See Table 2-1 for the color index of `ui[]` controls.

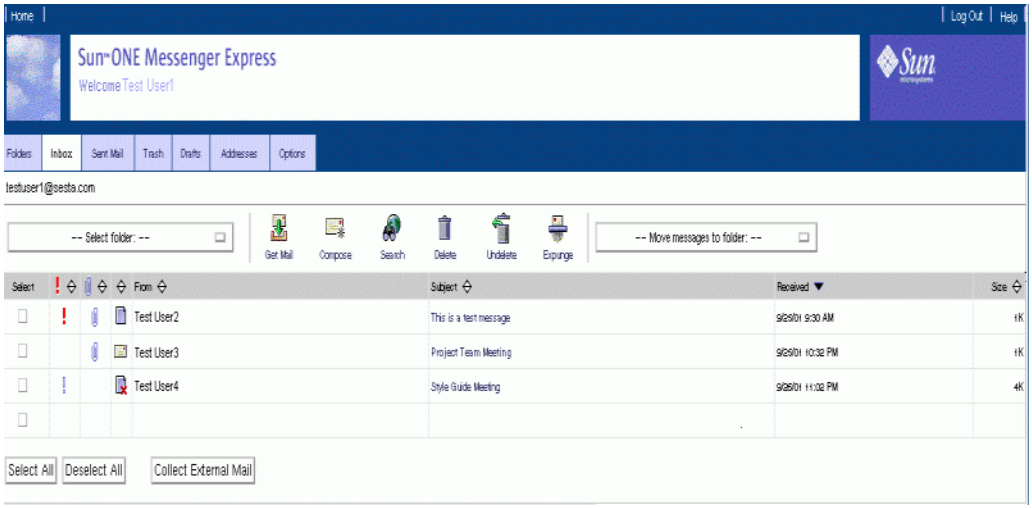
Table 2-1 Contains the color index of `ui[]` controls, their names and what they determine

Index	Name	Determines
0	accent0	Not used
1	accent1	Title bar
2	accent2	Not used
3	chrome0	Tab outlines
4	chrome1	Unselected tab background
5	chrome2	Selected tab background, tool bar, column headers, and so on
6	chrome3	Table cell background
7	ch3_img	Not used
8	link0	Unvisited link
9	link1	Visited link (almost always the same as unvisited)
10	link2	Active link
11	positive	Line drawing (white)
12	negative	Line drawing (black)
13	white	Page background color

Example—Color Sets Modifications

The example shown in Figure 2-4 customizes the default color set `ui[0]` to have `accent1` color maroon, `chrome4` color navy blue, and `chrome5` color silver.

Figure 2-4 Example Color Sets Modifications



Code Example 2-3 shows the necessary changes to be made in the file `main.js`.

Code Example 2-3 Altering Color Sets

```
var ui = new Array()

ui[0] = new
array('666699','800000','CCCCFF','666666','000080','C0C0C0',
'E6E6E6','gray90.gif','3333CC','3333CC','333366','FFFFFF',
'000000','FFFFFF','000000')
```

Modifying the Logo and Link

This section describes how to modify the Messenger Express corner logo and link shown in Figure 2-5.

Figure 2-5 Messenger Express Corner Logo and Link

You can modify the following on the Messenger Express corner logo and link:

- Replace logo with custom graphic
- Change destination of link

To Modify the Logo and Link

To modify the logo and link, edit the function `toolFrame()` in the `main.js` file.

Example—Logo Modification

The example shown in Figure 2-6 replaces the Sun ONE logo with a custom logo having different dimensions.

Figure 2-6 Example Corner Logo

Code Example 2-4 shows the necessary changes to be made in file `default.html` for replacing the Sun ONE logo `SunONE.jpg` with a custom logo `siroe.gif`.

Code Example 2-4 Replacing the Sun ONE logo with a custom logo.

```
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="10"><img src="images/spacer.gif" width="10" height="1"
alt=""></td>
<td valign="top" class="MstTdLogo" width="20%"><img
src="images/siroe.gif"
width="110" height="43" alt="Sun Microsystems, Inc."></td>
```

Code Example 2-5 shows the necessary changes to be made in file `default.html` for changing the link to the new URL for `siroe.com`.

Code Example 2-5 Changes to be made in the file `default.html` for changing the link to the new URL to `siroe.com`

```
....
<td colspan="2"><a href="http://www.siroe.com">
onMouseOver="over('bannerlinkone')"
onMouseOut="out('bannerlinkone')"
onClick="loadandswap('bannerlinkone', 'http://www.siroe.com');
return true"> <font size="-1"><span
 class="banner-links">siroe.com</span></font></a></td>
 </tr>
....
```

Modifying the Title Graphic and Text

Code Example 2-6 Changes to be made in file `default.html` for changing the link to the new URL for `siroe.com`.

```
....
<td colspan="2"><a href="http://www.siroe.com">
onMouseOver="over('bannerlinkone')"
onMouseOut="out('bannerlinkone')"
onClick="loadandswap('bannerlinkone', 'http://www.siroe.com');
return true"> <font size="-1"><span
 class="banner-links">siroe.com</span></font></a></td>
 </tr>
....
```

This section describes how to modify the title graphic and the text shown in Figure 2-7.

Figure 2-7 Messenger Express Title Text

You can modify the following on the Messenger Express title graphic and text:

- Replace title graphic with custom graphic
- Change the title text

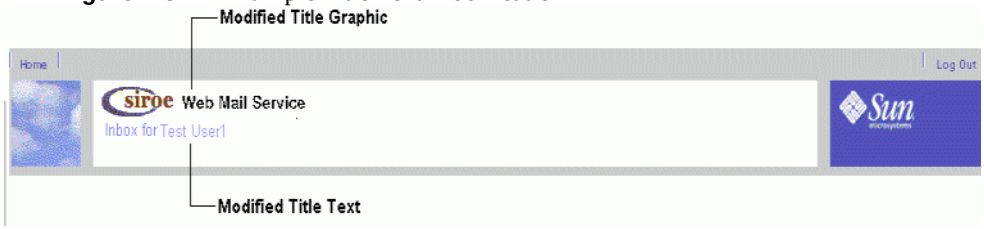
To Modify the Title Graphic and Text

To customize the layout of the title text, edit the function `toolFrame()` in the `main.js` file.

Example—Title Text Modification

The example shown in Figure 2-8 customizes the text to “Inbox for *user*.”

Figure 2-8 Example Title Text Modification



Code Example 2-7 shows the necessary changes to be made in file `main.js` to alter the graphic title.

Code Example 2-7 Altering Graphic Title

```
....
<TD width=1% nowrap bgcolor="'+ chrome1 +'"><NOBR>
<IMG src="imx/Banner.gif width="273" height="27"
 alt="'+i18n['Sun ONE messenger express']+'"
align=texttop border=0 hspace=5 vspace=3
</nobr></TD>
....
```

Code Example 2-8 shows the necessary changes to be made file `en/i18n.js` to alter the title text.

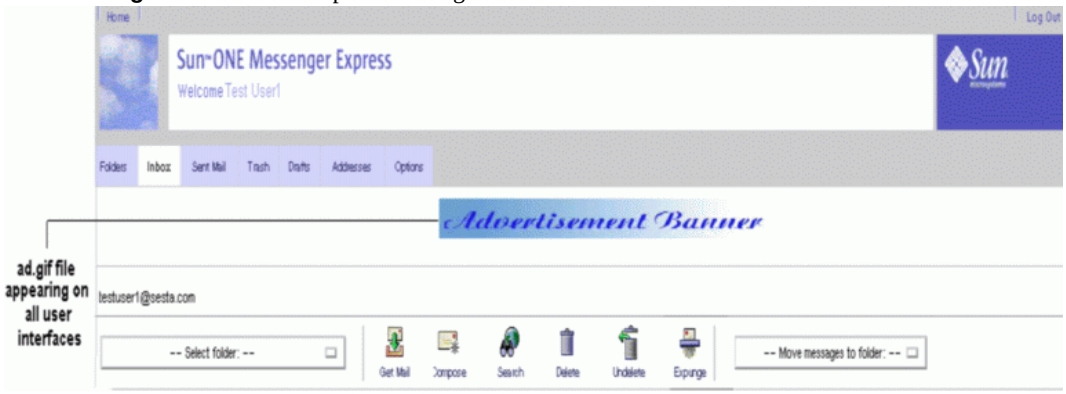
Code Example 2-8 Altering Title Text

```
function i18n_tab_header(user) {
  return '<nobr>  Inbox for ' + user
}
```

Inserting Banners and Links

This section describes how to insert advertisement banners and links on the Messenger Express user interface shown in Figure 2-9.

Figure 2-9 Example Inserting Common Banner and Link



You can modify the following on the Messenger Express user interface:

- Add a banner and link to appear on all the pages
- Add a banner and link to appear only on certain pages

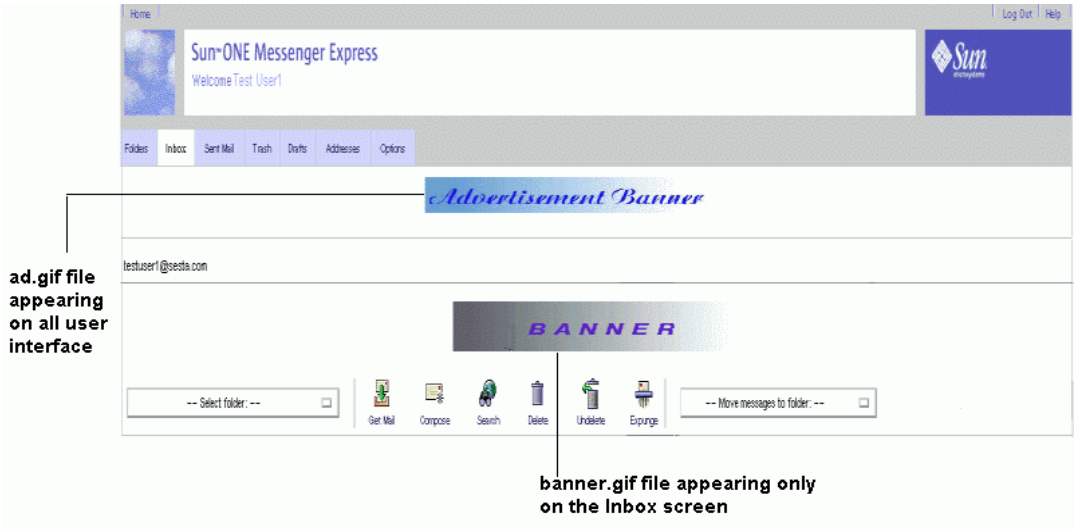
To Insert Banner and Link

- Edit the function `toolFrame()` in the `main.js` file to add a common banner and link.
- Edit the function `getToolbar()` and `frameset` height in the files `comp_fs.html`, `fldr_fs.html`, `mbox_fs.html`, `msg_fs.html`, `opts_fs.html`, and `searchmsg_fs.html` to add separate banners and links.

Example—Inserting Banners and Links

The example shown in Figure 2-10 inserts a banner and link in the Messenger Express Inbox screen.

Figure 2-10 Example Inserting a Banner and Link in the Inbox Screen



Code Example 2-9 shows the code before inserting banner and link on all user interface in the `main.js` file.

Code Example 2-9 Before Inserting a Banner and Link

```
function toolFrame() {
if (isRefreshing())
    return ''
....
tab(i18n['options'], state == 'options', 'selectOptions()') +
   '<td width=50%>' + nbsp + '</td>\n' +
   '</tr></table>\n' +
   '</td>\n' +
   '</tr></table>\n' +
   getBreadCrumb() +
   mailFrame.getToolbar() + '</form>'
}
```

Code Example 2-10 shows how to insert a banner and link to the `main.js` file.

Code Example 2-10 After Inserting a Banner and Link

```
function toolFrame() {
 if (isRefreshing())
    return ''
 ....
   tab(i18n['options'], state == 'options', 'selectOptions()') +
 '<td width=50%>' + nbsp + '</td>\n' +
   '</tr></table>\n' +
   '</td>\n' +
   '</tr><tr><td width=100% bgcolor='+main.chrome1+'
   align=center><a href="http://www.sesta.com">
   <img src="imx/ad.gif" width=200
   height=25></a></td></tr></table>\n' + getBreadCrumb() +
   mailFrame.getToolbar() + '</form>'
}
```

Code Example 2-11 shows the code before inserting banner and link to the Inbox screen, in `mbox_fs.html` file.

Code Example 2-11 Before Inserting Banner and Link to the Inbox Screen

```
function getToolbar() {
  var s
  var enable = main.sortid.length > 0
  s = '<table border="0" cellpadding="0" cellspacing="0">\n' +
     '<TR><TD colspan=42><IMG src="imx/spacer.gif" width="1"
     height="8"></TD></TR>'+
    '<tr align="center">\n' +
 '<td width="10"> </td>\n' +  '<td align="center" nowrap>'
 + main.font(1) +
main.folderSelectionBreadCrumbs('folderList','main.selectMbox(op
tions[selectedIndex].value);selectedIndex=0',
```

Code Example 2-12 shows how to insert a banner and link to the Inbox screen by editing `mbox_fs.html`. The banner and link are displayed only on the Inbox screen.

Code Example 2-12 After Inserting Banner and Link to the Inbox Screen

```
function getToolbar() {
var s
var enable = main.sortid.length > 0
```

Code Example 2-12 After Inserting Banner and Link to the Inbox Screen *(Continued)*

```
s = '<table border="0" cellpadding="0" cellspacing="0">\n' +
'<TR><TD width=100% align=center>'+
'<img src="imx/spacer.gif" width="405" height=1>
<a href ="http://www.siroe.com"><img src="imx/banner.gif"
width=200 height=35></a></td></tr>'+
    '<tr align="center">\n' +
 '<td width="10"> </td>\n' +  '<td align="center" nowrap>'
 + main.font(1) +
main.folderSelectionBreadCrumbs('folderList','main.selectMbox(op
tions[selectedIndex].value);selectedIndex=0',
....
```

Customizing User Interface Features

This chapter describes how to customize user interface features of Messenger Express.

This chapter contains the following sections:

Modifying the Main Function Tabs

This section describes how to modify the Messenger Express main function tabs shown in Figure 3-1.

Figure 3-1 Messenger Express Main Function Tabs, With Default Labels

You can modify the following on the Messenger Express main function tabs:

- Interchange the location of tabs
- Change the text of the tab labels

To Modify the Main Function Tabs

To modify the main function tabs, edit the appropriate files as follows:

- For tab layout, edit the `toolFrame()` function in the `main.js` file.
- For text used in the default tab labels, in the *lang*`/i18n.js` file, edit the `i18n[]` values for `folders`, `message`, and `options` in the `// Tabs` section, and `help` and `logout` in the `// Tool Bars` section.
- For text of the default folder name labels (including the initially displayed Inbox tab label), edit the `fldr[]` values in the `// Localized folder names` section in the *lang*`/i18n.js` file.

Functionally, the tabs are constructed by the `toolFrame()` function. The `toolFrame()` function calls the `tab()`function in the `main.js` file and specifies the text of the tab label to be displayed.

The following functions, located in `main.js`, handle the default tabs:

- Folders: `displayFolders()`
- Inbox: `displayMbox()` or `refreshMbox()` (depending on the state selected)
- Message: `selectMsg()`
- Options: `selectOptions()`
- Help: `help()`
- Logout: `logout()`

Example—Main Function Tabs Modifications

The example shown in Figure 3-2 moves the Options tab to the right, and changes the text of its tab label to “Preferences.”

Figure 3-2 Example Main Function Tabs Modifications

Code Example 3-1 shows the necessary changes to be made in file `main.js` (layout).

Code Example 3-1 Altering Function Tabs Layout (`main.js`)

```
function toolFrame() {
.....
'<td width=30%>' + nbsp + '</td>\n' +
tab(i18n['options'], state == 'Options', 'selectOptions()') +
....
}
```

Code Example 3-2 shows the necessary changes to be made in file `en/i18n.js` (tab labels).

Code Example 3-2 Altering Function Tabs Labels (`en/i18n.js`)

```
// Tabs
i18n['folders'] = 'Folders'
i18n['message'] = 'Message'
i18n['pab'] = 'Addresses'
i18n['options'] = 'Preferences'
```

Modifying the Mailbox Tool Bar

This section describes how to modify the Messenger Express mailbox tool bar shown in Figure 3-3.

Figure 3-3 Messenger Express Mailbox Tool Bar

You can modify the following on the Messenger Express mailbox tool bar:

- Change the layout of the mailbox tool bar relative to the rest of the page
- Rearrange the order of tools
- Change the tools text

To Modify the Mailbox Tool Bar

To modify the mailbox tool bar, edit the appropriate files as follows:

- To customize the layout relative to the rest of the page, edit the `toolFrame()` function in the `main.js` file.
- To customize the layout within the tool bar and the associated graphics, edit the `getToolbar()` function in the `mbox_fs.html` file.
- To customize the text associated with the graphics in the Tool Bar, edit the `i18n[]` values `get mail`, `compose`, `search`, `new search`, `file selected message`, `delete`, `undelete`, and `expunge` in the *lang*`/i18n.js` file.

Functionally, `toolFrame()` in `main.js` calls `getToolbar()`in `mbox_fs.html` to get the HTML code to write out to the page.

The `getToolbar()` function in `mbox_fs.html` assembles the code and assigns the functions to the graphics by calling `toolbar()` in `main.js`, which takes care of items such as colors and text-only versions.

The `getToolbar()` function in `mbox_fs.html` also calls `folderSelection()` in `main.js` to generate the drop-down folder list.

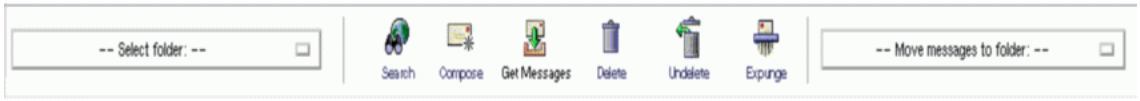
The functions assigned by `getToolbar()` in `mbox_fs.html` that handle the tool clicks are:

- Get Mail: `main.refreshMbox()`
- Compose: `main.compose("new")`
- Search: `parent.srch()`
- Move Messages to Folder: `parent.move()`
- Delete: `parent.delmsg()`, `parent.undelmsg()`, `parent.exmsg()` (depending on whether the message is in the trash folder or not)

Example—Mailbox Tool Bar Modifications

The example shown in Figure 3-4 makes “Search” as the first tool and changes the text of the “Get Mail” tool to “Get Messages.”

Figure 3-4 Example Mailbox Tool Bar Modifications



Code Example 3-3 shows the necessary changes to be made in files `mbox_fs.html` (layout).

Code Example 3-3 Altering Tool Bar Layout (`mbox_fs.html`)

```
function getToolbar() {
....
main.WMtoolbar(
....
(main.srch != '' ? i18n['new search'] : i18n['search']),
 'parent.srch()', 'imx/search.gif', 27, 25, true,
i18n['get mail'], 'main.refreshMbox()', 'imx/pull.gif', 27, 25,
true, i18n['compose'], 'main.compose("new")', 'imx/compose.gif',
27, 25, true)
....
}
```

Code Example 3-4 shows the necessary changes to be made in file `en/i18n.js` (text)

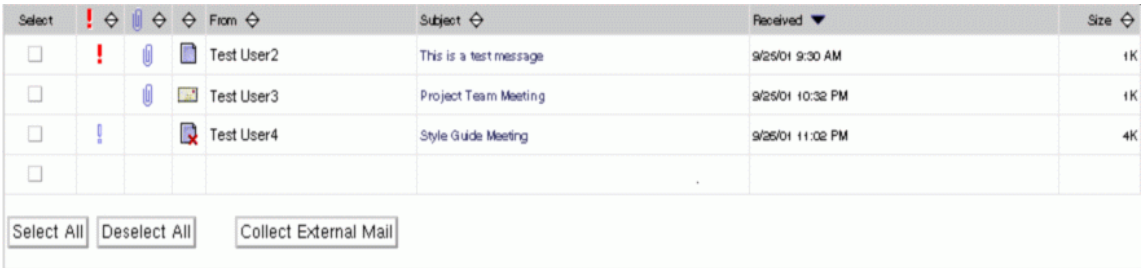
Code Example 3-4 Altering Tool Bar Text (`en/i18n.js`)

```
// Tool Bars
....
i18n['get mail'] = 'Get Messages'
```

Modifying the Message List Window

This section describes how to modify the Messenger Express Message List window shown in Figure 3-5.

Figure 3-5 Messenger Express Message List Window



You can modify the following in the Messenger Express Message List window:

- By default change the sorting order
- Change the text of the default column heading
- Change the text on “Collect External Mail” button

To Modify the Message List Window

To modify the Message List window, edit the appropriate files as follows:

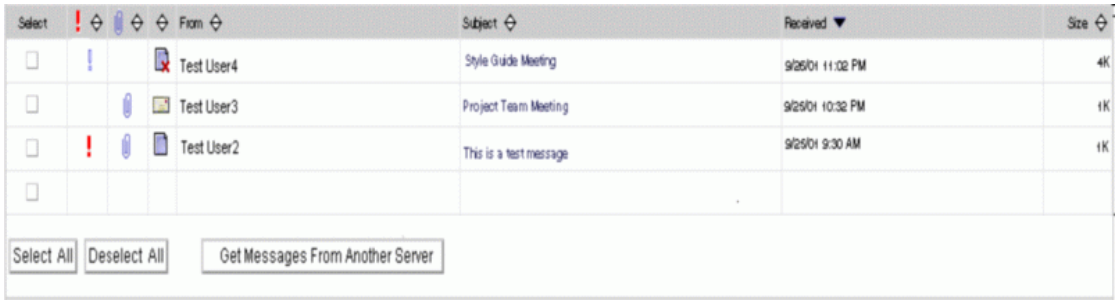
- To customize how the message list appears, edit the `listFrameHTML()` function in the `mbox_fs.html` file.
- To customize the order in which messages are listed, edit the `defaults[]` values near the top of the `main.js` file.
- To change the text of default column heading, edit the `i18n[]` values for `search results`, `date`, `from`, `to`, `size`, and `subject` in the *lang*`/i18n.js` file.
- To change the text on the “Collect External Mail” button, edit `i18n['collect long]` in the *lang*`/i18n.js` file.

Functionally, `listFrameHTML()` calls `getSortHeader()` in `mbox_fs.html` and assigns column headings with appropriate call to the sorting function `sortMsgs()` in `main.js`. The `listFrameHTML()` function also links the “Collect External Mail” hyperlink to `collect()` in `main.js`.

Example—Message List Window Modifications

The example shown in Figure 3-6 displays the most recently received mails first, and changes the text on “Collect External Mail” button to “Get Messages From Another Server”.

Figure 3-6 Example Message List Window Modifications



Code Example 3-5 shows the necessary changes to be made in files `main.js`.

Code Example 3-5 Altering Message List Window Layout (`main.js`)

```
var defaults = new Array(
.....
'meSortOrder', 'L',
.....
)
```

Code Example 3-6 shows the necessary changes to be made in file `en/i18n.js`.

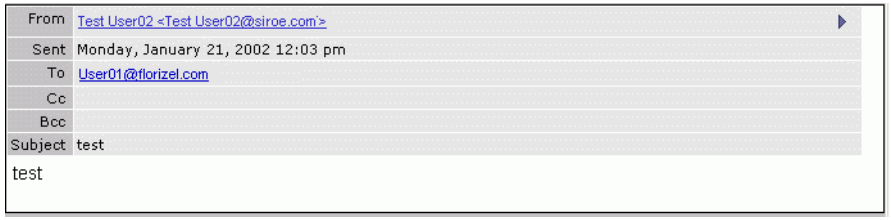
Code Example 3-6 Altering List Window Text (`en/i18n.js`)

```
// POP3 Collection
....
i18n['collect long'] = 'Get Messages From Another Server'
```

Modifying the Message Display Window

This section describes how to modify the Messenger Express Message Display window shown in Figure 3-7.

Figure 3-7 Messenger Express Message Display Window



You can modify the following in the Messenger Express Message Display window:

- Change the message display
- Alter the layout of, window
- Change the text of the fields
- Display User Defined Header Fields

To Modify the Message Display Window

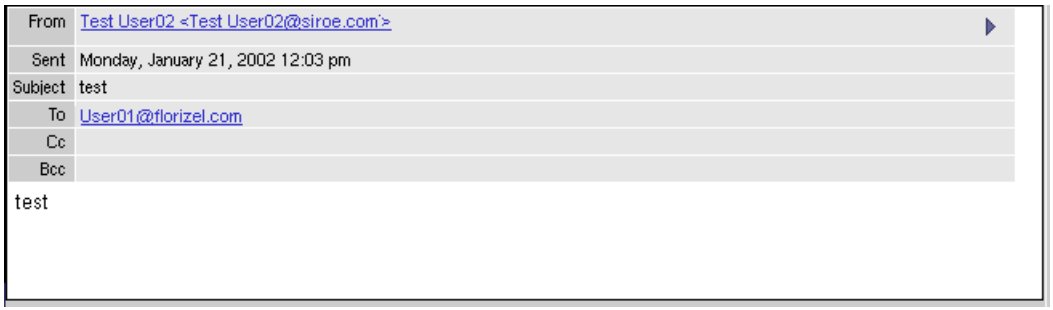
To modify the Message Display window, edit the appropriate files as follows:

- To customize how the message appears, edit the `listFrameHTML(doc)` function in the `msg_fs.html` file.
- To customize the default text, edit the `i18n[]` values under `// Message Headers` and `// Message` in the *lang*`/i18n.js` file.
- To customize the display defaults (word wrap), edit the `defaults[]` values `main.js` file.

Example—Message Display Window Modifications

The example shown in Figure 3-8 moves “Subject” before “To.”

Figure 3-8 Example Message Display Window Modifications



Code Example 3-7 shows the necessary changes to be made in file `msg_fs.html`.

Code Example 3-7 Altering Message Display Window Layout

```
function listFrameHTML(doc) {
....
s += header('from') + header('date') + header('subject')+
header('to') + header('cc')
....
}
```

Modify the Message Display Window to Display User Defined Header Fields

This section describes how to add and display user defined header fields in the Message Display window.

To Display User Defined Header Fields

Edit the `listFrameHTML(doc)` function in the `msg_fs.html` file.

Example—Modifying the Message Display Window to Display User Defined Header Fields

The example shows how to display the user defined field `X-document-id` in the Message Display Window.

Code Example 3-8 shows the changes made to the function `listFrameHTML()` in the `msg_fs.html` file.

Code Example 3-8 Changes made to the `msg_fs.html` file.

```
function listFrameHTML(doc) {
....
s += header('from') + header('sentdate') + header('to') +
header('cc') + header('bcc') + header('subject')
var hdrstr =  getHeaderStr(main.msgFrame.hdr[0], 'X-document-id')
if(hdrstr != '') {
s += <tr><td nowrap align=right valign=top width=5%' +
main.base_line + ' bgcolor=' + main.chrome2 + '>' + main.font() +
html('X-document-id') + nbsp + '</td>\n<td ' + main.cellBgString
+ '>' + extra + main.font() + hdrstr + '</td></tr>\n'
}
....
}
```

Modifying the Message Tool Bar

This section describes how to modify the Messenger Express message tool bar shown in Figure 3-9.

Figure 3-9 Messenger Express Message Tool Bar

You can modify the following in the Sun ONE Messenger Express message tool bar:

- Change the layout of the tool bar relative to the rest of the page
- Alter the layout within the tool bar
- Change the text associated with the graphics

To Modify the Message Tool Bar

To modify the message tool bar, edit the appropriate files as follows:

- To customize the layout relative to the rest of the page, edit the `toolFrame()` function in the `main.js` file.
- To customize the layout within the tool bar and the associated graphics, edit the `getToolbar()` function in the `msg_fs.html` file.
- To customize the texts associated with the graphics in the Tool Bar, edit the `i18n[]` values `compose`, `reply`, `reply all`, `forward`, `delete`, `undelete`, `previous`, and `next` in the *lang*`/i18n.js` file.

Functionally, `getToolbar()` in `msg_fs.html` assembles the code and assigns the functions to the graphics by calling `toolbar()` in `main.js`, which takes care of items such as colors and text-only versions.

The `getToolbar()` function in `msg_fs.html` also calls `folderSelection()` in `main.js` to generate the drop-down folder list.

The functions assigned by `getToolbar()` in `msg_fs.html` to handle the tools are:

- Compose: `main.compose('new')`
- Reply: `main.compose('reply')`
- Reply All: `main.compose('replyall')`
- Forward: `main.compose('forward')`
- Move Messages to Folder: `parent.move()`
- Delete and Undelete: `parent.delmsg()`
- Add all Addresses: `parent.addAllAddresses()`
- Previous: `parent.prev()`
- Next: `parent.next()`
- Close: `parent.gotofolder()`

Example—Message Tool Bar Modifications

The example shown in Figure 3-10 moves “Compose” to the right and abbreviates the text from “Previous” to “Prev.”

Figure 3-10 Example Message Tool Bar Modifications

Code Example 3-9 shows the necessary changes to be made in file `msg_fs.html`.

Code Example 3-9 Altering Message Tool Bar Layout (`msg_fs.html`)

```
function getToolbar() {
var s
...
main.WMtoolbar(
 null,null,'imx/spacer.gif',5,1,false,
 i18n['close'], 'parent.gotofolder()', 'imx/cancel.gif" alt="'
+i18n['msg close'], 24, 24, true,
null,null,'imx/spacer.gif',5,1,false,
 i18n['compose'], 'main.compose("new")', 'imx/compose.gif" alt="'
+ i18n['msg compose'], 24, 24, true) +
 <td><img src="imx/spacer.gif" width="1" height="1"></td>\n' +
....
```

Code Example 3-10 shows the necessary changes to be made in file `en/i18n.js`.

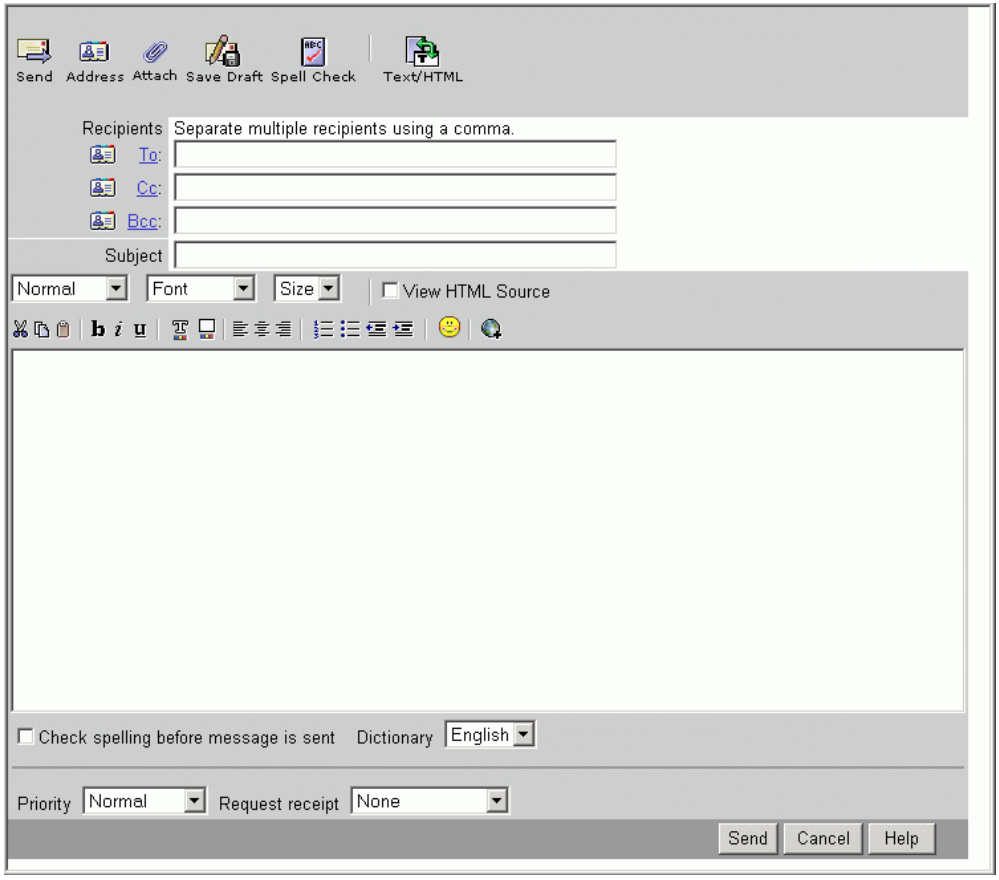
Code Example 3-10 Altering Message Tool Bar Text (`en/i18n.js`)

```
// Tool Bars
....
i18n['previous'] = 'Prev'
```

Modifying the Message Composition Window

This section describes how to modify the Messenger Express Message Composition window shown in Figure 3-11.

Figure 3-11 Messenger Express Message Composition Window



You can modify the following in the Messenger Express Message Composition window:

- Change the location of the tools in the window
- Alter text associated with the tools
- Enable and disable emoticons
- Create your own dictionary

To Modify the Message Composition Window

To modify the Message Composition window, edit the appropriate files as follows:

- To customize the window, edit the `compFrameHTML()` function in the `comp_fs.html` file.
- To customize the text, edit the values under `// Message Composition` and `// Tool Bars` in the *lang*`/i18n.js` file.
- To enable and disable the emoticons, edit the variable `iconHREF` in the `main.js` file. By default, the emoticon files are located in *msg_svr_base*`/html/imx` directory.

Functionally, `compFrameHTML()` in `comp_fs.html` assembles the code and assigns the functions to the graphics by calling `WMtoolbar()` in `main.js` which also handles colors and text-only versions. The `compFrameHTML()` function in `comp_fs.html` generates the “To”, “Cc”, and “Bcc” control area by calling `i18n_compose_controls()` in *lang*`/i18n.js`.

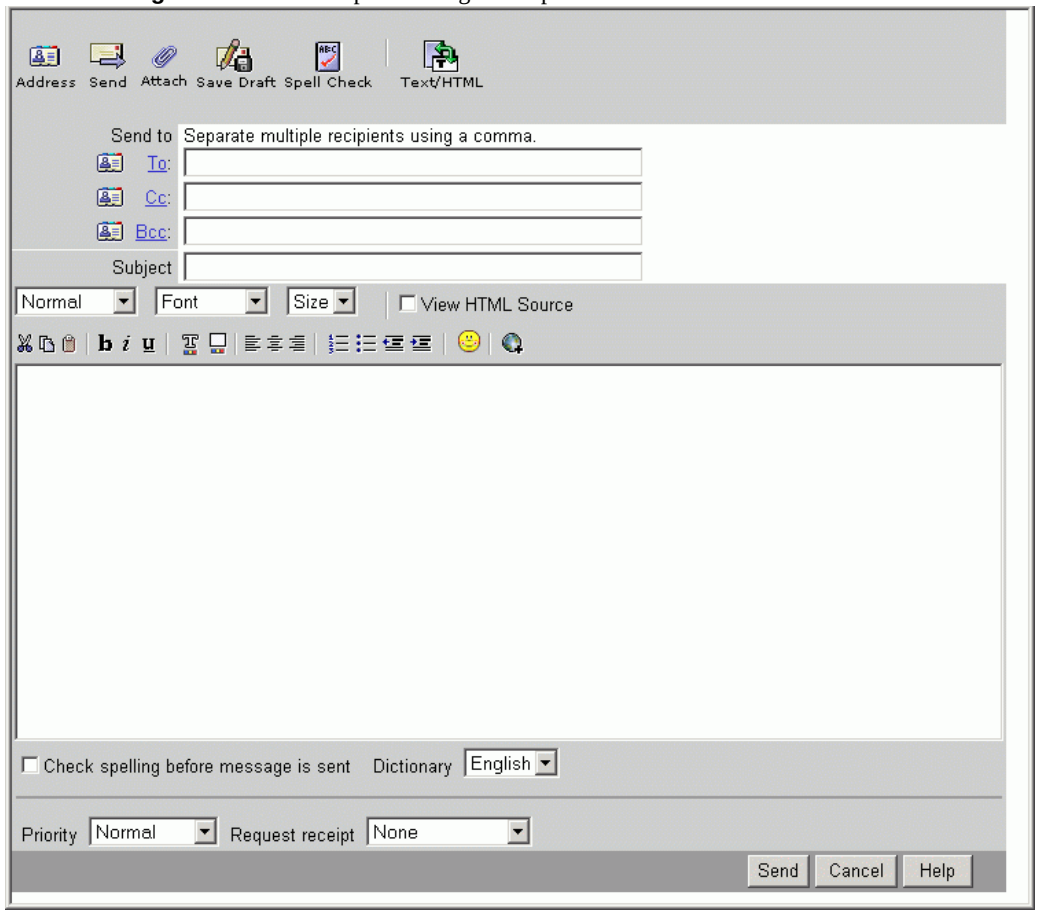
The functions assigned by `compFrameHTML()` in `comp_fs.html` are:

- Send: `parent.send('smtp')`
- Address: `parent.lookup()`
- Attach: `parent.attach()`
- Save Draft: `parent.send('draft')`
- Spell Check: `parent.spellchk()`
- Text / HTML: `parent.switchEditor()`
- Help: `main.help(1007399)`
- Cancel: `parent.cancel()`

Example—Message Composition Window Modifications

The example shown in Figure 3-12 moves the Address tool to the left so that it appears first on the tool bar, and changes the text “Recipients” to “Send to.”

Figure 3-12 Example Message Composition Window Modifications



Code Example 3-11 shows the necessary changes to be made in file `comp_fs.html` for swapping Address tool and Send tool.

Code Example 3-11 Altering Composition Window Layout (`comp_fs.html`)

```
function compFrameHTML() {

main.WMtoolbar(i18n['lookup'], 'parent.lookup()',
'imx/address.gif"
alt="' + i18n['lookup'], 27, 25, true, i18n['send'],
'parent.send('smtp')', 'imx/send.gif " alt="'
+ i18n['compose_send'],27, 25, true)

}
```

Code Example 3-12 shows the necessary changes to be made in file `en/i18n.js` for changing the text “Recipients” to “Send to”.

Code Example 3-12 Altering Composition Window Text(`en/i18n.js`)

```
// Message Composition
....
i18n['recipient'] = 'Send To'
```

The emoticons appear on the screen if the Text / HTML option is set to HTML. By default the Text / HTML option is set to Text format.

Code Example 3-13 shows how to edit the `main.js` file to enable emoticons.

Code Example 3-13 Altering Composition Window to Enable Emoticons

```
var iconHREF = 'msg_svr_base/imx/'
```

Code Example 3-14 shows how to edit `main.js` file to disable emoticons.

Code Example 3-14 Altering Composition Window to Disable Emoticons

```
var iconHREF = ''
```

To Include Additional Dictionary for Spell Check

1. Get the dictionary file and the affix file for the language you want to add to your dictionary.

   The dictionary file contains language-specific vocabulary and the affix file contains grammar rules for the specific language. For information on dictionary and affix files refer to the site: `http://fmg-www.cs.ucla.edu/fmg-members/geoff/ispell-dictionaries.html`

   Messenger Express is shipped with French and English (United States) dictionaries, which are located in the *msg_svr_base*`/dict` directory.

2. Use the `buildhash` utility to create a platform-specific and language-specific hash file from the dictionary and affix files. This hash file is used by the Messenger Express spell checker.

   To run the `buildhash` utility, download the `ispell` source files available at the site: `http//www.gnu.org/software/ispell/ispell.html` or, use the `buildhash` utility in the *msg_svr_base*`/dict/bin` directory. The syntax for the `buildhash` utility is:

   `buildhash dictionary_file affix_file` *language_name*`.hash`

   The `language_name` in the *language_name*`.hash` file is the two-letter language code used by Messenger Express (such as: `en` for English, `fr` for French).

   To determine your language's two-letter code, enter the command:

   msg_svr_base`/msg-instance/configutil | grep local.supportedlanguages`

NOTE The double-byte character set is not supported by the Messenger Express spell checker.

3. Copy the newly created *language_name*`.hash` file in the *msg_svr_base*`/dict` directory and restart the `mshttpd` service. When the `mshttpd` service is restarted, the Messenger Express spell checker is enabled.

   Code Example 3-15 shows how to create a German hash file (`ge.hash`) by using the `buildhash` utility.

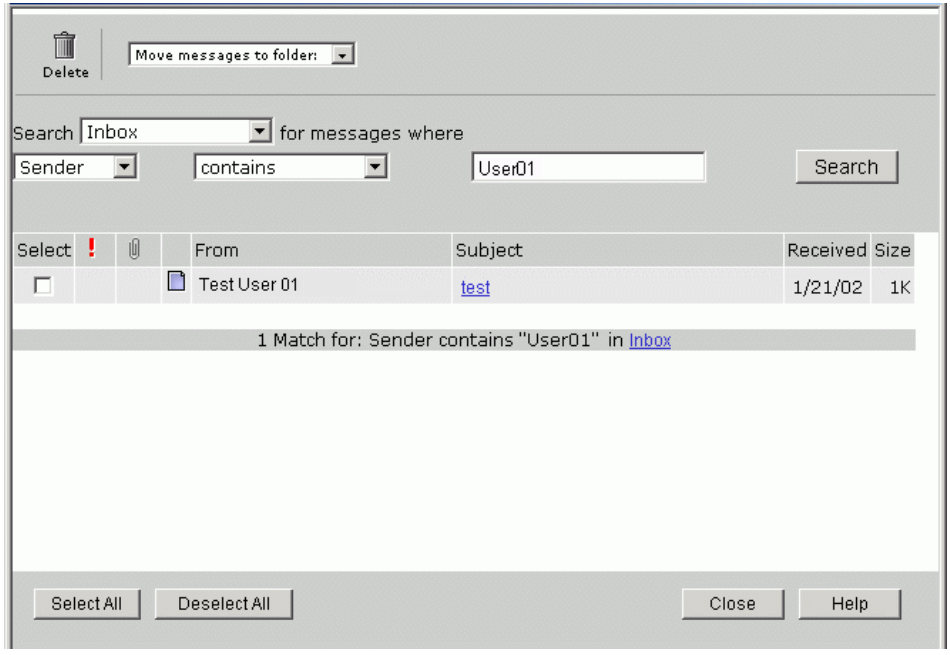
Code Example 3-15 Creating a German hash file by using the `buildhash` utility.

```
# cd /usr/Sun/server5/dict/bin
# ./buildhash german.dico german.aff ge.hash
# cp ge.hash ..
# /usr/Sun/server5/msg-budgie/start-msg http
```

Modifying Message Search Window

This section describes how to modify the Messenger Express Message Search window shown in Figure 3-13.

Figure 3-13 Message Search Window



You can modify the following on the Messenger Express Message Search window:

- Change the layout of the window
- Change the text associated with the tool

- Change the tool bar layout in the Message View window

To Modify the Message Search Window

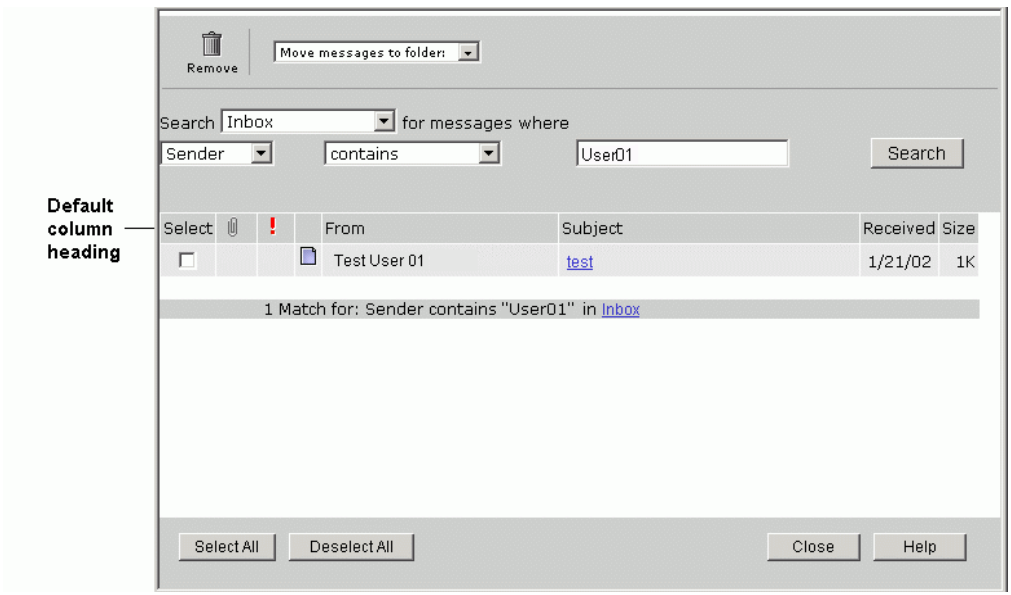
To modify the Message Search window, edit the appropriate files as follows:

- To alter the layout of the window, edit the `listFrameHTML()` function in the file `/en/searchMessage.html`.
- To change the text associated with the tool, edit the file `en/i18n.js`.
- To alter the tool bar in the Message View window, edit the `getToolbar()` function in the `seachmsg_fs.html` file.

Example—Message Search Window Modifications

The example shown in Figure 3-14 changes the text associated with the tool "Delete" to "Remove". It also shows how to interchange the appearance of the default column headings.

Figure 3-14 Message Search Window With Changes



Code Example 3-16 shows the changes to be made in file `en/i18n.js`, to change the text associated with the tool.

Code Example 3-16 Altering the Tool Text

```
// Tool Bars
i18n['delete'] = 'Remove'
```

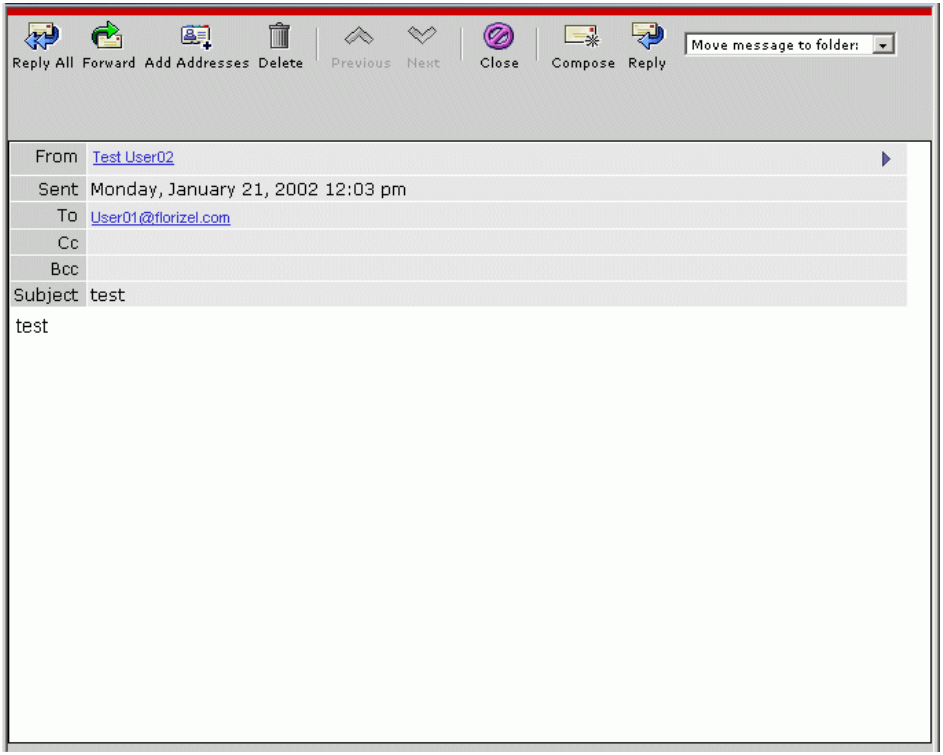
Code Example 3-17 shows the necessary changes to be made in the `searchMessage.html` file.

Code Example 3-17 Interchanging the Order of Appearance of the Default Column Headings

```
function listFrameHTML() {
  var i, msg
....
s += main.tableStart + '<form name="form1"><tr bgcolor=' +
main.chrome2 + '>\n'
s += '<td align="center" width=1% nowrap>' + main.font() +
    i18n['selectLabel'] + '</td>\n'
s += '<td align="center" width=5% nowrap><img
src="../imx/attach.gif"
alt="' + i18n['msg Attachment'] + '" width=16 height=16' +
 (IE >= 4 || NN > 0 ? ' hspace=2>' : '>') + getSortHeader('') +
'</td>\n'
s += '<td align="center" width=1% nowrap><img
src="../imx/high-0.gif"
alt="' + i18n['msg priority'] + '" width=16 height=16' +
 (IE >= 4 || NN > 0 ? ' hspace=2>' : '>') + getSortHeader('') +
'</td>\n'
  s += '<td align="left" width=1% nowrap>' + getSortHeader('') +
'</td>\n'
...
```

The example shown in Figure 3-15 alters the tool bar layout in the Message View window.

Figure 3-15 Message View Window with the Tool bar Modifications



Code Example 3-18 shows the necessary changes to be made in the file `seachmsg_fs.html` to alter the tool bar layout.

Code Example 3-18 Message View Window Tool Bar Modifications

```
function getToolbar() {
  var s
....
main.WMtoolbar(
....
i18n['close'], 'parent.closeMe()', 'imx/cancel.gif" alt="' +
i18n['msg close'], 27, 25, true,
null, null, 'imx/divider.gif', 2, 24, false,
i18n['compose'], 'main.compose("new")', 'imx/compose.gif" alt="'
+ i18n['msg compose'], 27, 25, true,
i18n['reply'], 'main.compose("reply")', 'imx/reply.gif" alt="' +
i18n['msg reply'], 27, 25, enable) +
    '<td nowrap>' + main.font(1) + main.nbsp +
 main.folderSelection('folderList',
```

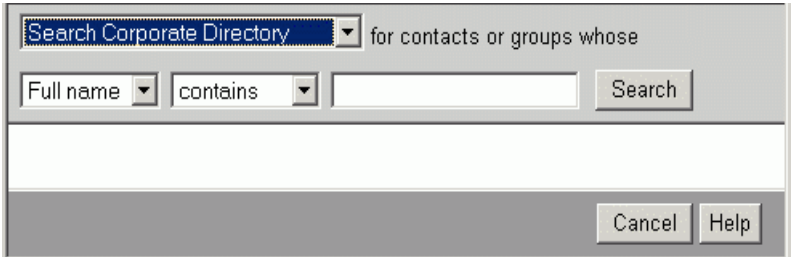
Code Example 3-18 Message View Window Tool Bar Modifications *(Continued)*

```
function getToolbar() {
    'parent.move(options[selectedIndex].value);selectedIndex=0',
i18n['file msg'], false,
main.mboxFrame.mbox,'',getsharedfolders())+ '</td>' +
....
```

Modifying the Address (Directory Lookup) Window

This section describes how to modify the Messenger Express Address (directory lookup) window shown in Figure 3-16.

Figure 3-16 Messenger Express Address (Directory Lookup) Window



You can modify the following in the Address (directory lookup) window:

- Change overall window appearance
- Alter search controls and their text
- Add additonal LDAP server search
- Alter all other texts
- Add User Defined Directory to Search

To Modify the Address (Directory Lookup) Window

To modify the Address window, edit the appropriate files as follows:

- To customize the overall window appearance, edit the `getSearchResult()` and `idxHTML()` functions in the `lookup.js` file.
- To customize other text, edit the `// LDAP Lookup` values in the *lang*`/lookup_fs.html` file.
- To add additional LDAP server search, add new entries specifying the LDAP host followed by the DN as the third parameter, edit the *lang*`/lookup_fs.html` file.

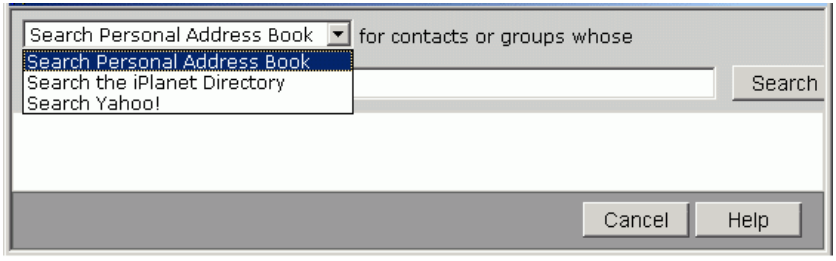
Functionally, `searchFrameHTML()` and `addFrameHTML()` assign the following functions to the buttons:

- Search: `parent.doSearch()`
- Cancel: `parent.cancel()`

Example—Address (Directory Lookup) Window Modifications

The example shown in Figure 3-17 changes “Search Corporate directory” to “Search the Sun ONE Directory” and adds the LDAP server search “Search Yahoo!” to the search list.

Figure 3-17 Example Address (Directory Lookup) Window Modifications



Code Example 3-19 shows the necessary changes to be made in `en/lookup_fs.html`.

Code Example 3-19 Altering Address Window Text

```
//Search Control
function s_SearchCtrl() {
....
'<option value="3 200">Search the Sun ONE Directory</option>\n' +
....
```

Code Example 3-20 shows how to add LDAP server search to the list. The file to edit is `en/lookup_fs.html`.

Code Example 3-20 Adding LDAP Server Search

```
//Search Control
function s_SearchCtrl() {
...
'<option value="2 25 ldap://ldap.yahoo.com/">' +
'Search Yahoo!</option>\n' +
'</select>\n' +
...
```

Adding a User Defined Directory to Search

You can add an additional user defined directory to the LDAP server search in the Address (Directory Lookup) Window. This can be accomplished by adding new entry for the LDAP host followed by the DN in the *lang*`/lookup_fs.html`.

To a Add User Defined Directory to Search to the Address (Directory Lookup) Window

Add a new entry specifying the LDAP host followed by the DN as the third parameter in the *lang*`/lookup_fs.html` file.

Example—Adding a User Defined Directory to Search to the Address (Directory Lookup) Window

The example shows the changes made to Address (Directory Lookup) Window to search for the user defined directory in the host `florizel.com` with the DN `ou=People, o=florizel.com`.

Code Example 3-21 shows how to add the user defined directory to the search list in the Address (Directory Lookup) Window. The file to edit is *lang*`/lookup_fs.html`.

Code Example 3-21 Adding user defined directory to search.

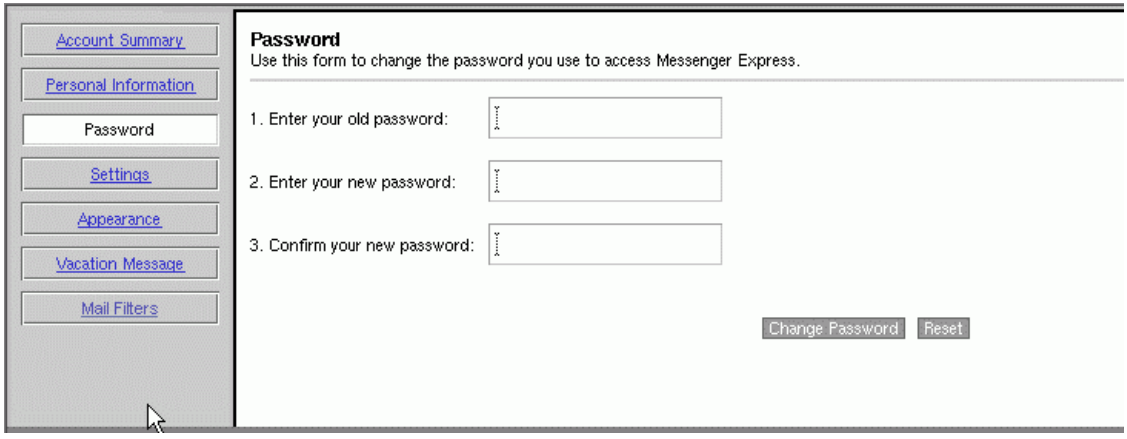
```
//Search Control
function s_SearchCtrl() {
...
'<option value="2 25
ldap://ldap.florizel.com/ou=People,o=florizel.com">' +
'Search Florizel!</option>\n' +
...
```

NOTE Code Example 3-21 will work only if anonymous reads are allowed on the DN `ou=People,o=florizel.com`. Otherwise to bind to the host `florizel.com`, you need to provide the values for `binddn` and `bindpwd` in the `ldap.msc` file.

Modifying the Options Window

This section describes how to modify the Messenger Express Options window shown in Figure 3-18.

Figure 3-18 Messenger Express Options Window



You can modify the following on the Messenger Express Options window:

- Change the layout of the window
- Change the default text

To Modify the Options Window

To modify the Options window, edit the appropriate files as follows:

- To customize the options and the layout of the options, edit the `toggleFrameHTML()` function in the `opts_fs.html` file.
- To customize default text, edit the `i18n[]` values under `// Options` in the *lang*`/i18n.js` file.

Example—Options Window Modifications

The example shown in Figure 3-19 moves “Vacation Message” between “Personal Information” and “Password,” and changes the text “Messenger Express” to “Mozilla Super Speedy Web Mail.”

Figure 3-19 Example Options Window Modifications

Code Example 3-22 shows the necessary changes to be made in the file `opts_fs.html` to move “Vacation Message” between “Personal Information” and “Password.”

Code Example 3-22 Altering Options Window Layout (`opts_fs.html`)

```
function toggleFrameHTML()  {
....
 getToggle(main.i18n['personal'], 'personal',
 'javascript:parent.toggle("personal")') +
 getToggle(main.i18n['vacation'], 'vacation',
 'javascript:parent.toggle("vacation")') +
 getToggle(main.i18n['password'], 'password',
 'javascript:parent.toggle("password")') +
....
}
```

Code Example 3-23 shows the necessary changes to be made in the file `en/i18n.js` to change the text "Messenger Express" to "Mozilla Super Speedy Web Mail.

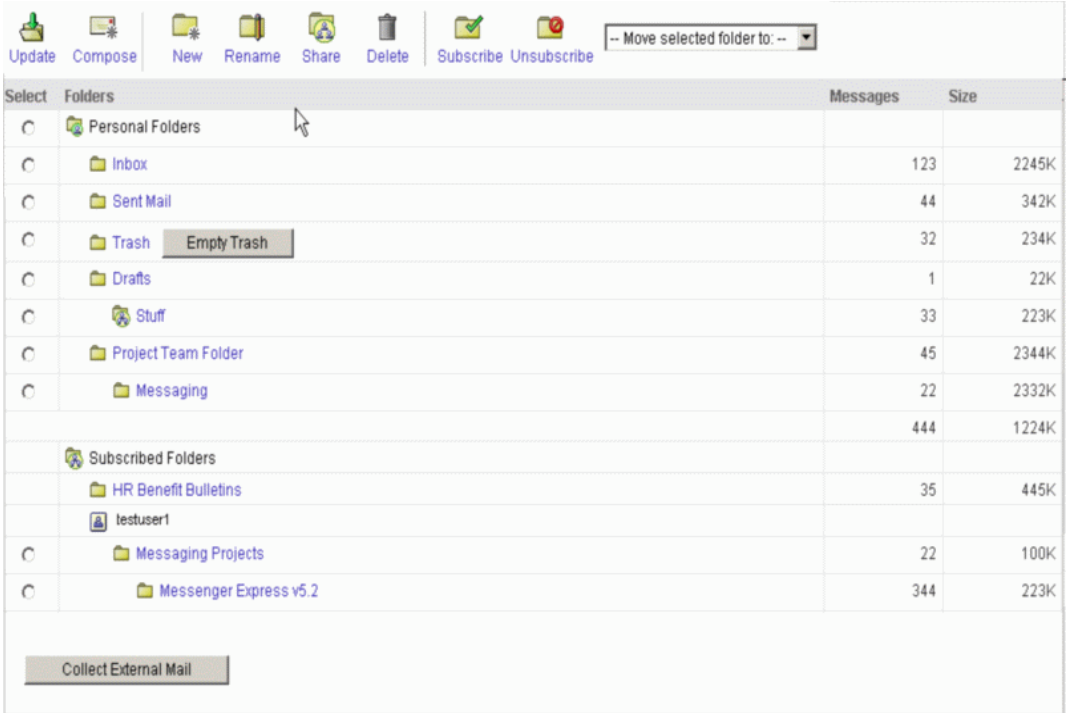
Code Example 3-23 Altering Options Window Text (`en/i18n.js`)

```
// Options
....
i18n['passwd exp'] = 'Use this form to change the password you use
to access Mozilla Super Speedy Web Mail.'
```

Modifying the Folders Window

This section describes how to modify the Messenger Express Folders window shown in Figure 3-20.

Figure 3-20 Messenger Express Folders Window



You can modify the following in the Messenger Express Folders window:

- Change the location of the tools
- Change the text associated with the tools or folders

To Modify the Folders Window

To modify the Folders window, edit the appropriate files as follows:

- To rearrange the tools on the folders toolbar, edit the `getToolbar()` function in the `fldr_fs.html` file.
- To customize the column headings, appearance of the buttons, and color of the window, edit the `listFrameHTML()` function in the `fldr_fs.html` file.
- To customize the text of the “Collect External Mail” button, edit `i18n['collect long]` in the *lang*`/i18n.js` file.

- To customize any other text, edit the `// Folders` section in *lang*`/i18n.js`.

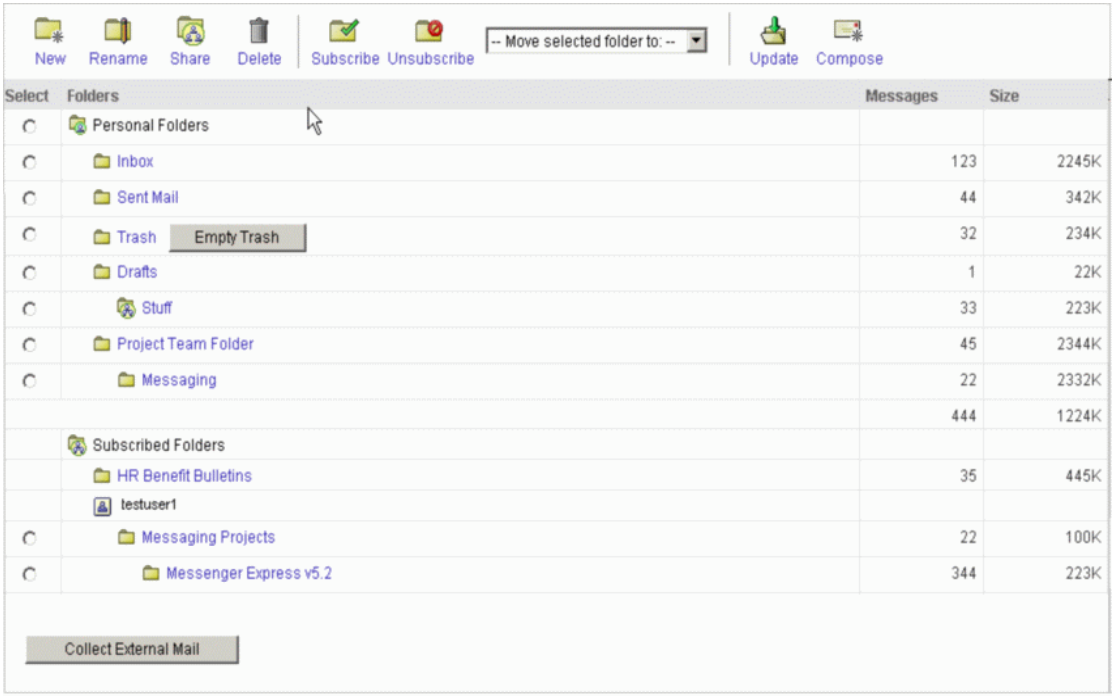
The functions assigned to the tools and links by `getToolbar()` and `listFrameHTML()` in `fldr_fs.html` are:

- Update: `main.refreshFolders()`
- Compose: `main.compose('new')`
- New: `parent.addFolder()`
- Rename: `parent.renFolder()`
- Share: `parent.setfolder()`
- Move Folder: `parent.moveFolder(options[selectedIndex].value)`
- Delete: `parent.delFolder()`
- Subscribe: `main.subscribeFolder()`
- Unsubscribe: `parent.unsubscribeFolder()`

Example—Folders Window Modifications

The example shown in Figure 3-21 moves “Update” and “Compose” tools to the end of the toolbar.

Figure 3-21 Example Folders Window Modifications



Code Example 3-24 shows the changes to be made in the file `fldr_fs.html`.

Code Example 3-24 Altering Folders Window Layout

```
function getToolbar() {
....
main.WMtoolbar(
i18n["new folder"], 'parent.addFolder()', 'imx/fldr_new.gif"
alt="' + i18n['folder new'], 24, 24, true,
i18n['rename'], 'parent.renFolder()', 'imx/fldr_edit.gif" alt="' +
i18n['folder rename'], 24, 24, true,
....
main.WMtoolbar(
null, null, 'imx/spacer.gif', 2, 24, false,
i18n['update'] ? i18n['update'] : i18n['get mail'],
'main.refreshFolders()', 'imx/Update_Folder.gif" alt="' +
i18n['folder update'], 24, 24, true,
i18n['compose'], 'main.compose("new")', 'imx/compose.gif',
alt="'+ i18n['msg compose'], 24, 24, true
....
}
```

Aligning the User Interface from Right to Left

By default the Menu tabs are aligned from left to right. When you have selected Arabic as the preferred language, then you need to customize Messenger Express to align the Menu tabs from right to left.

To align the User Interface from right to left

- To align the User Interface from right to left you need to remove a line from the `ar/i18n.js` file.

Code Example 3-25 shows the line that needs to be removed from the `ar/i18n.js` file.

Code Example 3-25 Aligning the User Interface from right to left

```
....
i18n['dir'] = 'ltr'
....
```

Disable Filtering of the HTML Tags

For security reasons the Messenger Express server filters the HTML tags that are used to encode multimedia in mails which are in HTML format. The user can disable the filtering of these HTML tags in the mails.

To disable the filtering of the HTML tags:

- Change the `/html/main.js` file to disable filtering the HTML tags in the mails.

Example — Disable the filtering of the HTML tags

The example shows how to disable filtering of JavaScript tags in the mails.

Code Example 3-26 shows the `/html/main.js` file before changing the file for disabling the filtering of Javascript tags in the mails.

Code Example 3-26 Before editing the `main.js` file.

```
....
load(msgFrame, msgHREF + 'msg.msc?sid=' + sid + '&security=' +
security + '&mbox=' + encode(selectmbox) + '&uid=' + num +
'&process=js,link,target,html' + binhex + maxtext +
get_charset())
....
```

Code Example 3-27 shows the changes to be made in the `/html/main.js` file for disabling the filtering of Javascript tags in the mails

Code Example 3-27 After altering the `main.js` file.

```
....
load(msgFrame, msgHREF + 'msg.msc?sid=' + sid + '&security=' +
security + '&mbox=' + encode(selectmbox) + '&uid=' + num +
'&process=link,target,html' + binhex + maxtext +
....
```

Supporting a New Locale

If the language is supported by the Messaging Server, you can add the language to the preferred language list and create the language-specific static webmail pages. These language-specific pages should be grouped in a subdirectory under the main document directory. The webmail code automatically detects the client's language preference and fetches the webmail pages from the appropriate subdirectory.

To support a new language in Messenger Express:

- Add the new language to the preferred language list. The new language can be added by editing the `i18n_preferredlanguagelist()` funtion in the `i18n.js` file.
- Copy the files from the directory `html/en` to this language directory `html/`*lang*.
- Change the charset to `iso-8859-5` in the lang/`i18n.js`

Example—Supporting a New Locale

The example shows how to support Russian language in Messenger Express.

Code Example 3-28 shows the changes to be made to the `i18n_preferredlanguagelist()` funtion in the `i18n.js` file for supporting Russian language.

Code Example 3-28 Adding a language to the `18n_preferredlanguagelist()` function in the `i18n.js` file

```
function i18n_preferredlanguagelist()
{
var s = '<select name="preferredLanguage">' +
'<option value="ar">Arabic</option>' +
'<option value="zh-CN">Chinese/Simplified</option>' +
'<option value="zh-TW">Chinese/Traditional</option>' +
'<option value="en">English</option>' +
'<option value="fr">French</option>' +
'<option value="de">German</option>' +
'<option value="it">Italian</option>' +
'<option value="ja">Japanese</option>' +
'<option value="ko">Korean</option>' +
'<option value="ru">Russian</option>' +
....
'</select>'
return s
}
```

Code Example 3-29 shows the changes to be made to the `ru/i18n.js` file for changing the charset in the i18N resource file.

Code Example 3-29 Changing the charset in the I18N Resource File.

```
// I18N Resource file
var i18n = new Array()
var fldr = new Array()
// DO NOT TRANSLATE AS STRINGS-JUST VALUES
i18n['client charset'] = 'iso-8859-5'
i18n['http charset'] = 'iso-8859-5'
i18n['fontface'] = 'PrimaSans BT,Verdana,sans-serif
i18n['fontface1'] = i18n['fontface']
i18n['fontface2'] = 'Times New Roman,Times,serif'
i18n['fontface3'] = 'Courier New,Courier,mono'
i18n['nbsp'] = ' '
```

Chapter 4

Customizing Advanced Features

This chapter describes how to perform advanced customizations. It also provides HTML files and their location and details on features that are fully customizable for the Messenger Express user interface.

This chapter contains the following sections:

Advanced Customization Overview

In addition to the Messenger Express features discussed in Chapter 2, "Customizing General Features", and Chapter 3, "Customizing User Interface Features", many others are fully customizable. However, to take advantage of these features, an in-depth knowledge of JavaScript is required. In addition, migration problems might be encountered, for example, when attempting to reconfigure JavaScript files.

NOTE This chapter does not provide code samples for the advanced customizations. Also, an advanced knowledge of JavaScript and HTML is assumed.

Messenger Express User Interface Customizable Features

Table 4-1 lists the features of the Messenger Express user interface that are fully customizable.

Table 4-1 Messenger Express User Interface Customizable Features

Features	Files
Attachments	`attach_fs.html`
Collection of mail from another server	`collect_fs.html`
Message composition	`comp_fs.html`
Folder management tab	`fldr_fs.html`
Addresses search	`ldap_fs.html`
Mailbox management tab	`mbox_fs.html`
Message management tab	`msg_fs.html`
Personalize option management	`opts_fs.html`
Return receipt	`receipt_fs.html`
Subscribe Folders	`subscribe_fs.html`
Search Messages	`searchmsg_fs.html`
Emoticon	`emoticons.html`

Attachments

You can modify the following attachments options:

- Browse button
- Attach, Cancel, Help buttons

HTML File Mapping

The HTML file that controls the attachment related features is `attach_fs.html`.

Collect Mail from Another Server

You can modify the following when collecting mail from another server:

- POP server name (text field)
- POP user ID (text field)
- Password (text field)
- Delete messages from server (select button)
- Save to folder (list box)
- Collect, Cancel, and Help (button)

HTML File Mapping

The HTML file that controls the mail collection feature is `collect_fs.html`.

Message Composition

The message composition feature enables basic mail functions. You can modify the following message composition options:

- Compose new message
- Reply to the sender
- Reply to all recipients of the message including the sender
- Forward message to others
- Move message to folder
- Delete message
- Navigate through messages using Prev or Next icon.

HTML File Mapping

The HTML file that controls the message composition feature is `msg_fs.html`.

Folder Management Tab

The folder management tab enables access to server-side folders. You can modify the following folder management tab options:

- Update content of folder
- Create new folder
- Rename folder
- Share folder
- Delete existing folder

HTML File Mapping

The HTML file that controls the folder management tab feature is `fldr_fs.html`.

Address Search

The address search feature enables the management of address search in LDAP directories. You can modify the following address search options:

- Search for people in the selected search directory (list box)
- Insert full name (text field)
- Contain field (text field)
- Search, Close (buttons)
- To, Cc, Bcc (buttons)

HTML File Mapping

The HTML file that controls the address search feature is `lookup_fs.html`.

Mailbox Management Tab

The mailbox management tab enables access to a mailbox. You can modify the following mailbox management tab options:

- Get new message
- Compose new message
- Search for message
- Move message to selected folder
- Delete message

- Undelete message
- Select message
- Select all messages
- Collect external messages

HTML File Mapping

The HTML file that controls the mailbox management tab feature is `mbox_fs.html`.

Personal Option Management (Options Tab)

You can modify the following Options tab options:

- Account summary
- Personal information - change personal information
- Password - change and reset password
- Settings
- Appearance
- Vacation message - set vacation message
- Mail Filters

HTML File Mapping

The HTML file that controls the options tab features is `opts_fs.html`.

Return Receipt

The return receipt feature enables the management of return receipts.

HTML File Mapping

The file that controls the return receipt feature is `receipt_fs.html`.

Shared Folders

A shared folder is folder in which you can store messages that can be accessed by other users. Messenger Express manages access to the shared folders by granting others access to the shared folders.

Sharing folders is a two-step process that involves sharing and subscribing:

1. A user shares a folder, specifying who has permissions to the folder.
2. The users who were given permissions to that folder then subscribe to it.

For more information on Shared Folders, see *Sun ONE Messenger Express Online Help*, that can be accessed by clicking the Help on Sun ONE Messenger Express.

Customization of Users' Default LDAP Attributes

The Messenger Express server loads a default set of LDAP attributes for a user at the start of a session. These attributes are as follows:

```
cn
givenName
mail
mailAlternateAddress
mailAutoReplyMode
mailAutoReplySubject
mailAutoReplyText
mailAutoReplyTextInternal
mailAutoReplyTimeout
mailDeliveryOption
mailForwardingAddress
mailQuota,mailMsgQuota
preferredLanguage
sn
uid
vacationEndDate
vacationStartDate
```

For more information on the attributes required or allowed by LDAP object classes for Sun ONE Messenger Express, see Chapter 3 Attributes in the *Sun One Messaging and Collaboration Schema Reference Manual* at:

`http://docs.sun.com/db/prod/s1msgsrv#hic`

You might want to obtain other customized LDAP attributes from the server. For example, an ISP might have a custom LDAP attribute assigned to all users called `myuserclass`. This attribute could denote different types of users that access services, including Messenger Express. Possible values for this attribute are "regular" and "vip".

NOTE Before adding custom attributes to the directory, ensure that the directory schema supports these new attributes. The global set of schema for Directory Server can be found in the entry named `cn=schema`.

For more information on extending directory schema, see *Sun ONE Directory Server Deployment Guide* at:

http://docs.sun.com/db/prod/s1dirsrv#hic

After extending the directory schema, you can add the new attribute to a User or a group entry. To add custom attribute to a User entry you will need to modify the entry in `servletsconf` section using the `ldapmodify` command. For more information on `ldapmodify` command, see *Sun ONE Directory Server Configuration, Command, and File Reference* at:

http://docs.sun.com/db/prod/s1dirsrv#hic

Depending on the type of user (that is, the value of the `myuserclass` LDAP attribute), different advertisement are presented to the user when they log into Messenger Express (Messenger Express is customized to display banner advertisement). If the customized client has access to the `myuserclass` LDAP attribute, the type of user can be determined and the relevant banner advertisement for that user type is displayed.

To obtain other customized LDAP attributes from the server, use `configutil` to modify the `service.http.extrauserldapattrs` configuration parameter. The attributes are read-only by default. If the user wants to modify an attribute using the Messenger Express code, that attribute needs to be marked read-write by appending the suffix `w`.

The example below assumes the user wants to display banner advertisements depending on the class of the user and that the client program allows the user to edit a link to a home page:

```
configutil -l -o service.http.extrauserldapattrs -v
"myuserclass,homepage:w "
```

Customizing Address Search to Return More LDAP Attributes

This section describes how to customize the Address Book Search to return more LDAP attributes.

You need to perform the following tasks to customize the Address Book Search to return more LDAP attributes:

- Obtain the additional LDAP attributes from the server using the `configutil` utility.
- Customize the client files to display the LDAP attributes obtained from the server.

For example, to customize the Address Book Search to return an additional LDAP attribute `pager`, perform the following steps:

1. Include the LDAP attribute `pager` to the list of LDAP attributes in the config attribute `local.service.http.ldapaddresssearchattrs` using the `configutil` utility:

```
configutil -o local.service.http.ldapaddresssearchattrs -v
"cn, mail, sn, telephoneNumber, pager"
```

NOTE Restart the Messaging server for the changes to take effect.

2. Add the following line to the `s_SearchCtrl()` function in the `lookup.fs` file:

```
....
' <select name="attr">\n' +
....
'<option value="'+ main.attr_list['telephonenumber']+'">Phone
#</option>\n' +
'<option value="'+ main.attr_list['pager']+'">Pager
#</option>\n' +
'</select>\n' +
....
```

3. Add the following lines to the `getSearchResults()` function in the `lookup.js` file:

```
if(forGroup) s = '<form name="form">\n' + main.tableStart +
....
'<td width=1% nowrap><nobr>' + main.font() + (pab ?
i18n_lu['work'] : i18n['ldap phone']) '</td>\n'+
'<td width=1% nowrap><nobr>' + main.font() + (pab ?
i18n_lu['pager'] : i18n['ldap pager']) +'</td>
else s = '</form><form name="form">\n' + main.tableStart +
'<td width=1% nowrap><nobr>' + main.font() + (pab ?
i18n_lu['work'] : i18n['ldap phone']) '</td>\n'+
'<td width=1% nowrap><nobr>' + main.font() + (pab ?
i18n_lu['pager'] : i18n['ldap pager']) +'</td>
....
```

4. Add the following lines to the `lookup.js` file:

```
....
if (list[i].telephoneNumber)
s += '<td nowrap>' + main.font() +
main.unescape_crlf(list[i].telephoneNumber) + '</td>\n';
else if (list[i].telephonenumber)
s += '<td nowrap>' + main.font() +
main.unescape_crlf(list[i].telephonenumber) + '</td>\n';
else
s += blank;
if (list[i].pager)
s += '<td nowrap>' + main.font() +
main.unescape_crlf(list[i].pager) + '</td>\n';
else if (list[i].pager)
s += '<td nowrap>' + main.font() +
main.unescape_crlf(list[i].pager) + '</td>\n';
else
s += blank;
s += '\n';
}
....
```

5. Add the following line to the `updatePabAttrList()` function in the `main.js` file:

```
....
attr_list['telephonenumber']=pabFrame.attrs?pabFrame.attrs.telep
honenumber.name:'telephonenumber';
attr_list['pager']=pabFrame.attrs?pabFrame.attrs.pager.name:'pag
er';
....
```

Domain Specific Customization

This section describes how to customize the Messenger Express client interface for each domain.

You can perform the following tasks to customize the Messenger Express client interface:

- Create a directory with the domain name under *msg_svr_base*/`html` directory.
- Populate this directory with the customized versions of the files from the original directory hierarchy.

  For example, assume that you have a domain called `siroe`. To change the icon for `siroe` domain, add a new icon in the `imx` directory of `siroe.com` and change the reference to it in the `main.js` file. The following is the directory structure for the domain `siroe.com`

Table 4-2 Directory Structure for the Domain `siroe.com`

`html/...`	/ / default interface
`html/imx/...`	/ / default interface
`html/en/...`	/ / default interface
`html/siroe.com/main.js`	/ / refers to `imx/bottle.gif`
`html/siroe.com/imx/bottle.gif`	

- After login, the server refers the user agent to pick the `main.html` file which is located in the *domain/lang* directory. The `main.html` file contains the relative references to the rest of the interface. The client requests all the files in the directory to make the interface. If these files exist in the *domain/lang* directory, they are displayed otherwise the default setup files from `html/en/` are displayed.

  If you have many domains and only a few distinct ‘brands’ then you can use links to make the server point to the correct brand:

Table 4-3 Linking multiple domains to few distinct brands

`html/...`	/ / default interface
`html/sesta.com/...`	/ / customized interface for brand 1

Table 4-3 *(Continued)*Linking multiple domains to few distinct brands

`html/...`	/ / default interface
`html/varrius.com -> sesta.com`	/ / default interface

Domain From URL

The server listens to all IP addresses and can present a customized interface before the authentication occurs. The server does this by looking at the URL and determines if it contains a known domain and presents the per domain Login screen for the domain.

For example, for the per domain Login screen: `http://webmail.sesta.com/`, the server presents the page from the location: `html/sesta.com/en/default.html`.

In this case a user does not have to suffix `@domain` to the username to login.

Chapter 5

Managing Authentication to the Messenger Express Service

This chapter describes how to integrate alternative authentication mechanisms with Messenger Express.

This chapter contains the following sections:

Introduction to Authentication

Some sites deploying Messenger Express might want to provide an alternative method of authenticating users. These sites might have portal service in which users sign on once at the “front door” to use various services without reauthentication. In this environment, the Messenger Express login mechanism has to recognize that the other service has already performed the authentication and recognize the user based on credentials provided by the other service.

This is known as *proxy authentication*. In most cases, users enter a user name and password to access the site. One of the services users can access from the site would be the Messenger Express. When users click on the link to open Messenger Express, it does not ask for a username or password again, because with proxy authentication the user name and password provided at the initial site login is referenced.

The Messenger Express authentication SDK (Software Development Kit) provides the following three components, which can be modified to accept the proxy authentication:

- Initialization
- Lookup
- Cleanup

SDK Files and Functions

To integrate the authentication SDK into the existing code, include the `expapi.h` header file into the calling code and link with the shared library (`.dll` or `.so` file). On some platforms, the authentication SDK may also require to link with other system libraries.

Table 5-1 lists the contents of *msg_svr_base*`/bin/msg/authsdk` (the install package).

Table 5-1 Contents of `authsdk`

Files	Function
`libexpress.so` or `DLL`	The SDK library
`cgiauth.c`	Source code for sample CGI using the API
`expapi.h`	Header file for API users
`login.html`	HTML source for the sample code
`nsapiauth.c`	Source code for the sample NSAPI plug-in using API
`README`	A readme documenting the API and use
`login.cgi`	Compiled CGI file
`Makefile.sample`	Example `makefile` to build `login.cgi`

SDK Configuration Initialization

The `EXP_Init` function initializes the SDK configuration information needed when calling other functions:

```
int EXP_Init(
char *pszLdapHost,
char *pszLdapMatchAttrib,
char *pszLdapDN,
unsigned int iLdapPort,
char *pszLdapBindUser,
char *pszLdapBindPass,
char *pszAdminUser,
char *pszAdminPassword);
```

Table 5-2 lists the SDK code variables and their description.

Table 5-2 SDK Code Variable Description

SDK Code Variables	Description
`pszLdapHost`	A null-terminated string containing the host name or IP address of the LDAP server in which the user search is performed.
`pszLdapMatchAttrib`	A null-terminated string specifying which LDAP attribute the `pszAdminUser` parameter should be matched against when searching the LDAP. The default is User ID (uid).
`pszLdapDN`	A null-terminated string specifying the Domain Name (DN) to use when searching for users.
`iLdapPort`	An integer specifying the port number in which the LDAP server is listening.
`pszLdapBindUser`	The string specifies the bind DN (Distinguish Name) and password for the directory server. If this is NULL, the SDK attempts to bind as anonymous user.
`pszLdapBindUser`	The string specifies the bind DN and password for the directory server. If this is NULL, the SDK attempts to bind as anonymous user.
`pszAdminUser`	The pointer to the string containing the “proxy” username and password is used when connecting to the messaging server. This is not NULL.

Table 5-2 SDK Code Variable Description

SDK Code Variables	Description
`pszAdminPass`	The pointer to the string containing the “proxy” username and password is used when connecting to the messaging server. This is not NULL.

A successful initialization, returns `0`. If initialization fails, a non-zero number is returned. If initialization fails, `errno` is set to the most appropriate value possible based on what failed (in most cases a system call). These codes then map to `standard errno` values.

SDK Lookup

The following functions are used to generate a session handle for a specific user and client IP address:

```
int EXP_GenerateLoginURL(
char *pszUser,
char *pszClientAddress,
char *pszMailHost,
char *pszURL);
```

Table 5-3 SDK Code Variable Description

SDK Code Variables	Description
`pszUser`	A null-terminated string containing the user ID (uid).
`pszClientAddress`	The string representation of the client’s IP address.
`pszMailHost`	A null-terminated string containing the hostname or IP address of the user’s mail server. If the third parameter is NULL, the LDAP server (from `EXP_init()`) is searched to determine this host. Otherwise, the mail host specified is used.
`pszURL`	The buffer allocated by the caller function in which URL is returned. The URL can have a maximum of 2048 characters long (including terminating NULL).

The function returns `0` on successful initialization, or a non-zero number on failure of initialization. On failure, `errno` is once again set to the most appropriate value.

The string returned by these functions is a login URL to be used when connecting to Messenger Express. Authentication applications (such as a login CGI) should call these functions after successfully authenticating the user based on the local authentication criteria. A typical CGI would use the resulting string to launch a URL or set a cookie on the client through HTTP headers or JavaScript.

SDK Cleanup

The following function is called to shut down and clean up any resources used by the SDK:

```
int EXP_Shutdown()
```

Typically it is not necessary to call the `EXP_Shutdown()` function in a simple CGI, but if the SDK is used as a plug-in in a continuous server running environment (such as Portal Server) the `EXP_Shutdown()` is called to reclaim the resources.

The function returns a `0` on successful initialization or a non-zero number indicating failure, and `errno` is then set to the most appropriate value based upon what failed. These codes also map to `standard errno` values.

The following function returns a `const` pointer to a null-terminated string identifying the version number of the SDK being used:

```
const char*EXP_GetVersion()
```

The value returned by the `EXP_GetVersion()` function should not be used in any dependant manner. In other words, do not write programs that expects this string to be in a certain format or contain a certain value. This string is only available to provide information on the version number of the SDK being used.

If no information on the version number is available, the function returns NULL.

The following function can be used to tell the SDK to contact a non-standard port when connecting to the Messenger Express service to generate a session:

```
void EXP_SetHttpPort
```

(init `iHttpPort`)

By default the SDK contacts the standard HTTP port, 80. This function is not thread safe and sets a global value. If you want to use it in a thread environment, lock the call and call the function `EXP_GenerateLoginURL`.

Example Deployment

The following example explains how the Proxy Authentication can be used. Use this example as a reference point to decide how to use the SDK in your environment.

- To test the Proxy Authentication API, see the `cgiauth.c` file. You need to edit certain `#define` statements appearing at the beginning of the file to suit your configuration:

```
#define HTML_SOURCE_FILE “login.html”
#define BUFFER_SIZE 1024
#define MAIL_SERVER “mail.yourdomain.com”
#define DIRECTORY_SERVER “directory.yourdomain.com”
#define DN “o=yourdomain.com”
#define ADMINNAME “admin”
#define ADMINPASS “admin”
```

- Change the values for `MAIL_SERVER`, `DIRECTORY_SERVER`, `DN`, `ADMINNAME`, and `ADMINPASS` to reflect your configuration. For example, if the mail server is `mail.siroe.com`, the directory server is `ldap.siroe.com`, and the administration username and password are `sysadmin`, then those lines should appear as follows:

```
#define HTML_SOURCE_FILE “login.html”
#define BUFFER_SIZE 1024
#define MAIL_SERVER “mail.siroe.com”
#define DIRECTORY_SERVER “ldap.siroe.com”
#define DN “o=siroe.com”
#define ADMINNAME “sysadmin”
#define ADMINPASS “sysadmin”
```

- Compile the `cgiauth.c` file into an executable CGI file and configure an HTTP server to execute the newly compiled CGI file.

If everything is set up correctly, upon running the script within a web browser, the CGI script presents the user with a simple Login screen based on the `login.html` file. If a user enters a valid username and password, the script launches a Messenger Express session without Messenger Express prompting the user to reauthenticate.

Index

F

G

H

I

J

L

M

O

P

R

S

T

U

V

V