Reqtool Documentation

Version 0.11

By Chris Debenham

Table of Contents

Introduction

Reqtool is a perl application that both handles email requests and provides a web interface to track, edit and resolve these email requests. It requires no external DB or additional apps other than a web server.

It includes a command line interface which processes incoming emails and a web front-end for dealing with the tickets.

Requirements:

Reqtool requires the following applications to be installed:
Perl 5.6 or later
Perl modules - CGI, MIME::Parser, Sys::Hostname and Date::Parse
A web server capable of running perl CGI
Net::LDAP is optional, needed only if you want LDAP authentication

Running:

When run from a terminal reqtool takes standard in and
processes it to be added to the tracking system. It will
mainly be used from a web browser where it provides an
interface to work with the individual requests in which case just point your prowser at the cgi page.

To create a new queue simply run make_new_queue from the reqtool directory and answer its' questions

Contact Details:

Any complaints, changes, suggestions, praise or whatever then e-mail me:
cjdebenh@users.sourceforge.net

MAINIFEST:

reqtool - Main program
install - Installation script
make_new_queue - Script to create a new queue

docs/Makefile - Makefile to create ps/html documentation
docs/reqtool.1 - Man page
docs/INSTALL - Installation documentation
docs/README - This file
docs/CHANGELOG - Changelog file
docs/TODO - List of ideas/things to do

default.conf - Default reqtool configuration file

mime.types - Example mime.types file
forwards.html - Example forwards file
config - Example default configuration
images/*.gif - Images for buttons

Developer notes

The source has been tidied by perltidy with the options:
--warning-output --check-syntax --tabs --paren-tightness=2 --square-bracket-tightness=2 --closing-side-comments --cuddled-else

HTML, ps, rtf and tex documentation can be made by ROBODOC. A makefile is provided to automate this. The man page is created via pod2man

The images used for the buttons were created using GIMP with the following settings
* Script-Fu/Buttons/Round Button
* UC = (.75,.75,.75) LC = (.5,.5,.5) TC = (0,0,0)
* Font Size=16 Font=HELVETICA
* Padding X,Y = 4,4
* Bevel=2
* Round Ratio=1

When running reqtool uses the following directories and files(based in GLOBAL_DIR which is normally /usr/local/reqtool)

reqtool - The main program
mime.types - mime-type to extension mapping file
images - Directory containing the images that will be used
- for the side buttons

default - Default queue.
- If you have another queue then this directory
- structure will be replicated for it
default/reqtool.count - Next request number
default/valid.users - Who is allowed to use this queue
default/config - Queue configuration file

default/active - Active queue directory
default/active/indexfile - Index of active requests
default/active/<req> - where <req> is a request number
- this contains a single request
default/active/<req>/request.contents - The tracking of a single request
default/active/<req>/msg-* - Individual emails in request

default/resolved - Resolved queue directory
- contents in same form as active directory

default/forward - Directory containing all forwards
default/forward/.forwards.html - HTML describing all forwards
default/forward/* - Individual files describing seperate forwards

Installation

For installation of Reqtool follow these 7 easy steps.

1. Make sure you have a web server and perl installed
Reqtool is written in perl and is accessed via a web server so you will require perl/apache (or similar) to be installed.
To download these go to http://www.perl.com and http://www.apache.org or just install them from your distributions install disks.
LDS also requires a few perl modules such as CGI and MIME-tools. These may already be installed or you can install them from CPAN or your install disks.

2. Download the main Reqtool program
LDS is available at http://prdownloads.sourceforge.net/reqtool/
reqtool-0.11.tar.gz

3. Extract the files
To untar the main program run tar xvfz reqtool-0.11.tar.gz

4. Run the install script
The install script checks for the existance of certain perl modules, asks a few
questions then installs the app

5. Edit reqtool for your site
Open up default/config in the editor of your choice
Edit this section as required

6. Setup email parsing
Next you need all mail going to the mailing-list to pass through reqtool.
A good way of doing this is to add a line to your /etc/mail/aliases (or similar) that looks like the following line (changing paths/addresses as needed)
reqtool-test: |"/usr/local/reqtool/reqtool default -"
If you don't wish to do it this way, just add a user to your machine, subscribe it to the list etc and add a . forward file which pipes the mail through /usr/local/reqtool/reqtool default -

7. Done
If all has gone well you should be able to send email to the mailing-list/email address and it will come up in the web interface which can be accessed as http://<hostname>//cgi-bin/reqtool

Manual Page

NAME

reqtool - Request Tracking Tool

SYNOPSIS

cat I<email> | B<reqtool> I<queue> -

or reqtool [I<queue> C<-r> | C<-v>]

or call directly as a CGI script

DESCRIPTION

When run from a terminal B<reqtool> takes standard in and processes it to be added to the tracking system.

It will mainly be used from a web browser where it provides an interface to work with the individual requests.

When called from the web the queue is set by calling it as B<reqtool/I<queue>>

OPTIONS

There is are three command line options

-

Take email from standard in

Queue name needs to be set on commandline

-r

Rebuild indexfiles from request directories

Use this is your indexfiles get corrupted or you manually change the contents of the active/resolved directories

Queue name needs to be set on commandline

-v

Get reqtool version

CONFIGURATION

All configuration is done by editing the config file in the directory of each queue

REQUIRES

Perl 5.6 or later, CGI, MIME::Parser, Sys::Hostname, Date::Parse, Net::LDAP is optional

AUTHOR

Chris Debenham <Chris.Debenham@Sun.Com>

COPYRIGHT

This program is released under the GPL (http://www.gnu.org/copyleft/gpl.html)

VERSION

Reqtool version 0.11

THANKS

This was based on the great work by the authors of Reqng which was inturn based on req but contains no code from either of those original programs.
Thanks go to Remy Evard and other members of the systems group at Northeastern University's College of Computer Science (http://www.ccs.neu.edu) for thier work on the original req.
Thanks also go to the authors of ReqNG (http://reqng.sycore.net/reqng) and WWWREQ (http://www.cs.ucr.edu/~cvarner/wwwreq/) which the interface was based on.

Source Files

reqtool

Methods

admin_forward admin_forward_do admin_main admin_users
admin_users_do display_buttons display_frames display_login
display_stats display_welcome load_config lock_append
lock_read lock_write login_check main_code
print_error print_page print_usage read_email
rebuild_indexes req_comments req_comments_do req_give
req_give_do req_mail req_mail_do req_merge
req_merge_do req_parameters req_rescan req_resolve
req_search req_search_do req_status request_stats
send_attachment send_image take_request view_request

Structures

buttons gconfig

Functions

date_diff display_comments display_email display_give
display_merge display_resolve search_file section_header
un_base64 untaint_num untaint_str

Variables

ACTIVE_DIR ACTIVE_INDEX ADMIN_PASSWORD CGI_NAME CHARSET
COUNTFILE FORWARD_DIR FORWARD_FILE IMAGE_PATH LDAP
LDAP_BASE LDAP_BIND LDAP_SERVER LISTNAME MAILING_LIST
MIME_TYPES QUEUE_DIR READ_USER RESOLVED_DIR RESOLVED_INDEX
TMP_DIR VALID_USERS VERSION button_order

Generic

pod setup

reqtool/setup

NAME

```
Setup section
```

DESCRIPTION

```
Loads required modules, sets some global variables,
and other global things
```

SOURCE

```
#
# Modules we use.
#
use strict;
use CGI qw(:standard :html3 :netscape :cookie :escapeHTML);
use CGI::Cookie;
use MIME::Parser;
use Sys::Hostname;
use Date::Parse;
use Fcntl ':flock';    # import LOCK_* constants

# un-taint environment
$ENV{'PATH'} = '/bin:/usr/bin';
delete @ENV{ 'IFS', 'CDPATH', 'ENV', 'BASH_ENV' };
```

---

reqtool/qconfig

NAME

%qconfig - reqtool instance configuration

DESCRIPTION

```
This is a large hash structure that contains all configuration
for this instance of reqtool
The majority of its values are set in load_config based
on values read from the queue configuration file
```

SOURCE

```
#
# Site Configuration.
# You should only have to edit the entry stating GLOBAL_DIR
# if you moved reqtool after installation
```

```
#
my %qconfig;

# Where this script will be in absolute directory terms
%qconfig->{'GLOBAL_DIR'} = "/usr/local/reqtool";
```

---

reqtool/VERSION

NAME

$**VERSION**

DESCRIPTION

Version number of this installation of reqtool

SOURCE

```
my $VERSION        = "0.11";
```

---

reqtool/buttons

NAME

%**buttons**

DESCRIPTION

This structure contains a list of the available **buttons**
The hash index is appended with 'req_' to give the function the button
will call, the first parameter is the text of the button
'.gif' is added to this text to detirmine the image.
The final parameter to each entry is the frame which it will target

SOURCE

```
my (%buttons) = (
        'rescan'     => ['Rescan',     'req_top'],
        'take'       => ['Take',       'req_status'],
        'give'       => ['Give',       'req_top'],
        'mail'       => ['Mail',       'req_top'],
        'merge'      => ['Merge',      'req_bottom'],
        'resolve'    => ['Resolve',    'req_status'],
        'drop'       => ['Drop',       'req_status'],
        'search'     => ['Search',     'req_top'],
        'comments'   => ['Comments',   'req_top'],
        'parameters' => ['Parameters', '_new'],
```

```
        'statistics' => ['Statistics', '_new'],
);
```

---

reqtool/button_order

NAME

@button_order

DESCRIPTION

This is a list of the buttons to be displayed
and the order they will be in

SOURCE

```
my (@button_order) = (
        'rescan',   'take',       'give',     'mail',
        'merge',    'resolve',    'drop',     'search',
        'comments', 'parameters', 'statistics'
);
```

---

reqtool/lock_write

NAME

lock_write -- Lock a file for overwriting

SYNOPSIS

lock_write ($filename)

FUNCTION

Create an exclusive lock on given file and blanks the file

INPUTS

$filename – filename to lock

OUTPUT

No output

SOURCE

```
sub lock_write {
        my $FILE = shift;
        flock($FILE, LOCK_EX);
        seek($FILE, 0, 0);
        truncate($FILE, 0);
} ## end sub lock_write
```

---

reqtool/lock_append

NAME

lock_append -- lock a file for appending

SYNOPSIS

lock_append ($filename)

FUNCTION

Create an exclusive lock on given file

INPUTS

$filename - filename to lock

OUTPUT

No output

SOURCE

```
sub lock_append {
        my $FILE = shift;
        flock($FILE, LOCK_EX);
}
```

---

reqtool/lock_read

NAME

lock_read -- lock a file for reading

SYNOPSIS

lock_read ($filename)

FUNCTION

```
Create a shared lock on given file
```

INPUTS

```
$filename – filename to lock
```

OUTPUT

```
No output
```

SOURCE

```
sub lock_read {
        my $FILE = shift;
        flock($FILE, LOCK_SH);
}
```

---

reqtool/untaint_num

NAME

```
untaint_num -- make a number untainted
```

SYNOPSIS

```
$num = untaint_num ( $num )
```

FUNCTION

```
Returns a blank if the string contains anything other than a number or 'R'
The 'R' is allowed since request numbers may contain one
```

INPUTS

```
$num – a possibly tainted number
```

OUTPUT

```
$num – a clean number or blank if original was tainted
```

SOURCE

```
sub untaint_num {
        my $num = shift;
        $num =~ /^([\dR]+)$/;
```

```
        return $1;
}
```

---

reqtool/untaint_str

NAME

untaint_str -- untaint a string

SYNOPSIS

```
$string = untaint_str ( $string )
```

FUNCTION

Checks if a string is tainted and returns it otherwise exits

INPUTS

$string – a possibly tainted string

OUTPUT

$string – a clean string

SOURCE

```
sub untaint_str {
        my $string = shift;
        if ($string eq "") {
                return $1;
        } elsif ($string =~ /^([-\@\w=\.\/ \,]+)$/) {
                return $1;
        } else {
                print_error("Tainted data in $string");
        }
} ## end sub untaint_str
```

---

reqtool/section_header

NAME

section_header -- Return nicely formatted section header

SYNOPSIS

```
$string = section_header ( $title, $bgcolor, $req )
```

FUNCTION

```
Return nicely formatted section header containing the title given, at
the color given and with a link to reload the request given
```

INPUTS

```
$title   - title of header
$bgcolor - color of section header
$req     - request number
```

OUTPUT

```
$text - string containing header as html source
```

SOURCE

```
sub section_header {
        my ($title, $bgcolor, $req) = @_;
        my $header =
          "<table border=0 cellpadding=0 cellspacing=0 width=100%><tr bgcolor="
          . $bgcolor
          . ">\n<td>";
        $header .= $title . "</td>\n<td align=right>";
        $header .= "<a href=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=view&req="
          . $req
          . "\" target=req_bottom><img src=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=getimage&image=refresh\" border=0></a>";
        $header .= "</td>\n</tr></table><br>\n";
        return $header;
} ## end sub section_header
```

---

reqtool/read_email

NAME

read_email -- Read in a email and turn into a request

SYNOPSIS

read_email ()

FUNCTION

```
Read in a email and turn into a request.
If the request exists, add to it
If the request was closed, re-open it
```

INPUTS

```
STDIN - email message passed via stdin
```

OUTPUT

```
New/modified request to queue
```

SOURCE

```
sub read_email {
        my ($from)     = "";
        my ($count)    = 0;
        my ($boundary) = "";
        my ($OUTFILE)  = "";
        my ($entity, $head, $subject, $preexists, $wasresolve, $oldpath, $bodyfile,
                $type);
        my ($attachpath, $request_dir);

        # Create a new parser object
        my $parser = new MIME::Parser;
        $parser->output_under(%qconfig->{'TMP_DIR'});
        $parser->extract_nested_messages(0);

        $entity = $parser->read(\*STDIN)
          or print STDERR "Couldn't parse MIME; continuing...\n";

        my $results = $parser->results;
        $head = $results->top_head;

        # Find subject and check if it relates to an existing request
        # If it does then set request number accordingly, otherwise
        # set request number to counter
        chomp($subject = $head->get('Subject', 0));
        $subject =~ s/\n//m;    # remove embedded newline
        if ($subject =~ /\[\s+%qconfig->{'LISTNAME'}\s+#(\d+).*\]/i) {
                $count     = $1;
                $preexists = 1;
        } else {
                open(REQ, "<" . %qconfig->{'COUNTFILE'}) or $count = 1;
                lock_read(\*REQ);

                if ($count == 0) {
                        lock_read(\*REQ);
                        $count = <REQ>;
                        close REQ;
                }

                if ($count == 1) {
                        open(REQ, "> " . %qconfig->{'COUNTFILE'});
                } else {
```

```
                open(REQ, "+< " . %qconfig->{'COUNTFILE'});
                lock_write(\*REQ);
        }
        print REQ ($count + 1);
        close REQ;
        $preexists = 0;
} ## end else

# Check if message is multi-part/alternative and handle
if ($entity->parts) {
        if ($entity->parts(0)->bodyhandle) {
                $oldpath = $entity->parts(0)->bodyhandle->path;
        } else {
                $oldpath = $entity->parts(0)->parts(0)->bodyhandle->path;
        }
} else {
        $oldpath = $entity->bodyhandle->path;
}

# Get temporary directory and parse
$bodyfile   = $oldpath;
$attachpath = $oldpath;
my $tmppath = %qconfig->{'TMP_DIR'};
$attachpath =~ s/$tmppath\/(.*)\/(.*)$/$1/g;
$bodyfile   =~ s/(.*)\/(.*)$/$2/g;
$oldpath    =~ s/(.*)\/(.*)$/$1/g;
$request_dir = %qconfig->{'ACTIVE_DIR'} . "/" . $count;

if ($request_dir =~ /^([-\@\w.\/]+)$/) {
        $request_dir = $1;
} else {
        print_error("Bad path data");
}

# If active directory doesn't exist, create it
if (!-e %qconfig->{'ACTIVE_DIR'}) {
        mkdir(%qconfig->{'ACTIVE_DIR'}, 0775);
}

# If the request was previously resolved, reopen it
if (($preexists)
        && (!-e $request_dir)
        && (-e %qconfig->{'RESOLVED_DIR'} . "/" . $count))
{
        system("/usr/bin/mv "
                  . %qconfig->{'RESOLVED_DIR'} . "/"
                  . $count . " "
                  . $request_dir);
        system("cp "
                  . %qconfig->{'RESOLVED_INDEX'} . " "
                  . %qconfig->{'RESOLVED_INDEX'}
                  . "~");
        open(IN, "<" . %qconfig->{'RESOLVED_INDEX'} . "~")
          || die "Cannot open: $!";
        lock_read(\*IN);
        open(OUT, "+<" . %qconfig->{'RESOLVED_INDEX'})
          || die "Cannot create: $!";
```

```
        lock_write(\*OUT);
        open(AINDEX, ">>" . %qconfig->{'ACTIVE_INDEX'})
          || die ("Couldn't open active index");
        lock_append(\*AINDEX);

        while ($_ = <IN>) {
                my @line = split (/\|/, $_, 6);
                if ($line[0] == $count . "R") {
                        $line[0] =~ s/R//g;
                        print AINDEX $line[0] . "|"
                          . $line[1] . "|"
                          . $line[2]
                          . "|open|"
                          . $line[4] . "|"
                          . $line[5];
                } else {
                        print OUT $_;
                }
        } ## end while ($_ = <IN>)
        close AINDEX;
        close OUT;
        close IN;
        unlink "%qconfig->{'RESOLVED_INDEX'}~";
} ## end if (($preexists) && (!...

# Make requisite directories for request, email and attachments
mkdir($request_dir, 0775);
mkdir($request_dir . "/" . $attachpath, 0775);

# Copy all attachments over
system("/usr/bin/cp " . $oldpath . "/* "
          . $request_dir . "/"
          . $attachpath
          . "/");

# Remove temporary files
system("/usr/bin/rm -rf " . $oldpath);

# Add section to request.contents recording event
open(REQTRACK, ">>$request_dir/request.contents")
  || print_error("Couldn't create $request_dir/request.contents");

lock_append(\*REQTRACK);
print REQTRACK "-- New Action --\n";
print REQTRACK "Action: Mail\n";
print REQTRACK "To: " . $head->get('To', 0);
print REQTRACK "Cc: ";

if ($head->get('Cc', 0)) {
        my $cc = $head->get('Cc', 0);
        $cc =~ s/\n//m;    # remove embedded newline
        print REQTRACK $cc;
}
print REQTRACK "\n";
if (!$head->get('Cc', 0)) { print REQTRACK "\n"; }
print REQTRACK "From: " . $head->get('From', 0);
if ($head->get('Subject', 0)) {
```

```
                my $subject = $head->get('Subject', 0);
                $subject =~ s/\n/ /g;
                print REQTRACK "Subject: " . $subject . "\n";
        } else {
                print REQTRACK "Subject: (no subject)\n";
        }
        print REQTRACK "Path: " . $attachpath . "\n";
        print REQTRACK "Body: " . $bodyfile . "\n";
        print REQTRACK "Date: " . localtime(time) . "\n";
        print REQTRACK "Content-Type: " . $head->get('Content-Type');
        print REQTRACK "-- End Action --\n";
        chmod 0755, $request_dir . "/" . $attachpath;

        # If this is new then add it to the index
        if (!$preexists) {
                open(INDEX, ">>" . %qconfig->{'ACTIVE_INDEX'})
                  || print_error("Couldn't open active index");
                chomp($from = $head->get('From', 0));
                if ($head->get('Subject', 0)) {
                        chomp($subject = $head->get('Subject', 0));
                } else {
                        $subject = "(no subject)";
                }
                lock_append(\*INDEX);
                $subject =~ s/\n/ /g;    # remove embedded newlines
                print INDEX $count . "|none|" . time . "|open|" . $from . "|" .
$subject
                  . "\n";
                close INDEX;
        } ## end if (!$preexists)
    } ## end sub read_email
```

---

reqtool/print_page

NAME

print_page -- Display text as html page

SYNOPSIS

print_page ($title, $contents, $header)

FUNCTION

```
Takes the page contents and outputs it to the web server (stdout)
Always called as last subroutine.
If header is not specified, default provided
```

INPUTS

```
$title    - Page title
```

```
$contents – Page contents
$header   – page header (optional) contains cookies etc
```

OUTPUT

```
HTML page to STDOUT
```

SOURCE

```
sub print_page {
        my ($title, $contents, $header) = @_;
        if ($header) {
                print $header;
        } else {
                print header (
                        -title => $title,
                        -type  => "text/html; charset=" . %qconfig->{'CHARSET'}
                );
        } ## end else

        print "<body bgcolor=#eeeeee>\n";
        print $contents;
        print "</body>\n</html>\n";
} ## end sub print_page
```

---

reqtool/print_error

NAME

print_error -- Display error as html page

SYNOPSIS

```
print_error ( $error_in )
```

FUNCTION

```
Creates a form html contents for delivery of error messages
```

INPUTS

```
$error_in – error message
```

OUTPUT

```
Calls print_page for display of HTML
```

SOURCE

```
    sub print_error {
            my ($error_in) = @_;
            my $error;

            $error = "<center><h1>You have encountered an error</h1></center><br>\n";
            . "<hr>\n";
            . "<font size=+1>Normally this is caused by trying to do an action on a
non-existant request<br>\n";
            . "For example, if you haven't chosen a request or the request has been
resolved since you last checked<br>\n";
            . "<br>\nPress <a href=\"javascript:history.go(-1)\">Here</a> to go back
and try again<br><br>\n";
            . "Error Was: <b><i>" . $error_in . "</i></b><br></font>\n";
            . "If, after trying again, it still fails please contact the";
            . "reqtool administrator and tell them the text of the error<br>\n";
            print_page("Error Encountered", $error);
            exit 1;
    } ## end sub print_error
```

---

reqtool/display_email

NAME

display_email -- Output an single email

SYNOPSIS

```
$email = display_email ( $message, $request_dir, $req )
```

FUNCTION

Takes an email message from a request and displays it

INPUTS

```
$message     - email as a MIME::Message object
$request_dir - directory where request is located
$req         - request number
```

OUTPUT

$email - HTML representation of message

SOURCE

```
    sub display_email {
            my ($message, $request_dir, $req) = @_;
            my ($email) = "";
            my ($attachments, $attachnum, $attach, $attach2, @message, $reqnum);
```

```
	# Email header fields
	$email = section_header("<h2>Message</h2>", "lightblue", $req);
	$email .= "<b>From: <i>" . escapeHTML($message->{'From'}) . "</i><br>\n";
	$message->{'Subject'} =~ s/(\?\=)\r?\n[\040\t]+(\=\?)/$1$2/sg;
	$message->{'Subject'} =~
	  s/\=\?([\w\-]+)\?B\?([A-Za-z0-9\+\/]+\=*)\?\=/&un_base64($2,$1)/gex;
	$email .= "Subject: <i>" . $message->{'Subject'} . "</i><br>\n";
	$email .= "To: <i>" . escapeHTML($message->{'To'}) . "</i><br>\n";
	$email .= "Cc: <i>";

	if ($message->{'Cc'}) {
		$email .= escapeHTML($message->{'Cc'});
	}
	$email .= "</i><br>\n";
	$email .= "Date: <i>" . $message->{'Date'} . "</i></b><br>\n";
	if (!($message->{'Content-Type'})) {
		$message->{'Content-Type'} = "";
	}

	# If the email is not a HTML mail then wrap it
	if (!($message->{'Content-Type'} =~ /text\/html/)) {
		$email .= "<pre wrap>\n";
	}

	# Find message boddy and display
	$request_dir = $request_dir . "/" . $message->{'Path'};

	if (!open(MESSAGE, $request_dir . "/" . $message->{'Body'})) {
		print_error(
			"Couldn't open " . $request_dir . "/" . $message->{'Body'}
. "\n");
	} else {
		lock_read(\*MESSAGE);

		while (<MESSAGE>) {
			if (!($message->{'Content-Type'} =~ /text\/html/)) {
				$email .= escapeHTML($_);
			} else {

				#$_=~ s/<?html.*>//g;
				#$_=~ s/<?body.*>//g;
				if (!((/\<.*html.*\>/) || (/\<.*body.*\>/))) {
					$email .= $_;
				}
			} ## end else
		} ## end while (<MESSAGE>)
		close MESSAGE;
	} ## end else
	if (!($message->{'Content-Type'} =~ /text\/html/)) {
		$email .= "\n</pre>";
	}
	$email .= "<br>\n";

	# Show any attachments
	$attachments = section_header("<h3>Attachments</h3>", "pink", $req);

	if (!opendir(DIR, $request_dir)) {
```

```
                print_error("can't opendir " . $request_dir . ": $!");
        } else {
                $attachnum = 0;
                while ($attach = readdir(DIR)) {
                        if ($attach =~ /^\./) {

                                #ignore
                        } elsif ($attach =~ /$message->{'Body'}/) {

                                # also ignore
                        } else {
                                $attachnum++;
                                $attachments .=
"<table bgcolor=#ccccdd border=1><tr><td bgcolor=#ddcccc>Attachment "
                                  . $attachnum
                                  . "</td></tr>\n";
                                $attachments .= "<tr><td>";
                                $attach2 = $attach;
                                $attach2 =~ s/ /\%20/g;
                                $reqnum = $req;
                                $reqnum =~ s/R//g;
                                $attachments .= "<a href=\""
                                  . %qconfig->{'CGI_NAME'}
                                  . "?command=download&req="
                                  . $req
                                  . "&file="
                                  . $reqnum . "/"
                                  . $message->{'Path'} . "/"
                                  . $attach2
                                  . "\" target=\"_new\">"
                                  . $attach;
                                $attachments .= "</a></td></tr>\n";
                                $attachments .= "</table>\n";
                        } ## end else
                } ## end while ($attach = readdir(...

                if ($attachnum) {
                        $email .= $attachments;
                }
                close DIR;
        } ## end else
        $email .= "\n";

        return scalar $email;
} ## end sub display_email
```

---

reqtool/display_comments

NAME

display_comments -- Output comments

SYNOPSIS

```
$text = display_comments ( $req, $message )
```

FUNCTION

```
Takes a comment from a request and outputs it
```

INPUTS

```
$req     – request number
$message – message containing comments
```

OUTPUT

```
$text – comments as HTML
```

SOURCE

```
 sub display_comments {
         my ($req, $message) = @_;
         my ($comments);

         $comments = section_header("<h2>Comments</h2>", "lightgreen", $req);
         $comments .= "<b>Comments added by <i>"
           . $message->{'Action-By'}
           . "</i> on <i>"
           . $message->{'Date'}
           . "</i><br><br>\n";
         $comments .= $message->{'Comments'};
         $comments .= "\n";

         return $comments;
 } ## end sub display_comments
```

---

reqtool/display_resolve

NAME

```
display_resolve -- Output resolved section
```

SYNOPSIS

```
$text = display_resolve ( $req, $message )
```

FUNCTION

```
Output a html section stating that the request was resolved
Different text is shown if the request was dropped
```

INPUTS

```
$req     – request number
$message – section describing resolve
```

OUTPUT

```
$text – HTML describing resolve action
```

SOURCE

```
sub display_resolve {
        my ($req, $message) = @_;
        my $comments;

        if ($message->{'Action-By'} eq "dropped") {
                $comments = section_header("<h2>Dropped</h2>", "red", $req);
                $comments .=
                  "<b>Dropped on <i>" . $message->{'Date'} . "</i><br><br>\n";
        } else {
                $comments = section_header("<h2>Resolved</h2>", "red", $req);
                $comments .= "<b>Resolved by <i>"
                  . $message->{'Action-By'}
                  . "</i> on <i>"
                  . $message->{'Date'}
                  . "</i><br><br>\n";
        } ## end else
        $comments .= "\n";

        return $comments;
} ## end sub display_resolve
```

---

reqtool/display_merge

NAME

display_merge -- Output merge

SYNOPSIS

```
$text = display_merge ( $req, $message )
```

FUNCTION

```
Outputs section saying request was merged with another
```

INPUTS

```
$req     – request number
```

```
$message – section describing merge
```

OUTPUT

```
$text – HTML describing merge
```

SOURCE

```
sub display_merge {
        my ($req, $message) = @_;
        my $comments;

        $comments = section_header("<h2>Merged</h2>", "red", $req);
        $comments .= "<b>Contents Merged from Request <i>"
          . $message->{'Request-Merged'}
          . "</i> on <i>"
          . $message->{'Date'}
          . "</i></b><br><br>\n";

        return $comments;
} ## end sub display_merge
```

---

reqtool/display_give

NAME

display_give -- Output a section showing that request was given

SYNOPSIS

```
$text = display_give ( $req, $message )
```

FUNCTION

INPUTS

```
$req     – request number
$message – section describing give
```

OUTPUT

```
$text – HTML describing 'give'
```

SOURCE

```
sub display_give {
        my ($req, $message) = @_;
        my ($give);
```

```
            $give = section_header("<h2>Give Request to User</h2>", "lightgreen",
$req);
            $give .= "<b>Given by <i>" . $message->{'Giver'} . "</i> to <i>" .
$message->{'Owner'} . "</i> on <i>" . $message->{'Date'} . "</i></b><br>\n";

            return $give;
    } ## end sub display_give
```

---

reqtool/view_request

NAME

view_request -- view a single request

SYNOPSIS

view_request ($req)

FUNCTION

Loads a request and then, by calling other subroutines, displays it as html

INPUTS

$req - request number

OUTPUT

HTML page display by calling print_page

SOURCE

```
    sub view_request {
            my ($req) = @_;
            my (
                    $request_dir, $submessage, @line,   @message, $sub,
                    $contents,    $cookie,     $header, $rest,    $reqtrack
            );

            $reqtrack = $req;

            # Check if this is a resolved request
            if ($req =~ /R/) {
                    $req =~ s/^(.*)R$/$1/g;
                    $request_dir = %qconfig->{'RESOLVED_DIR'} . "/" . $req;
            } else {
                    $request_dir = %qconfig->{'ACTIVE_DIR'} . "/" . $req;
            }
```

```
		# If we can't open the request see if it was recently resolved
		if (!open(REQTRACK, "$request_dir/request.contents")) {

			# try resolved
			$request_dir = %qconfig->{'RESOLVED_DIR'} . "/" . $req;
			$reqtrack    = $req . "R";
			open(REQTRACK, "$request_dir/request.contents")
			  || print_error("Couldn't open $request_dir/request.contents");
		} ## end if (!open(REQTRACK, "$request_dir/request.contents"...
		lock_read(\*REQTRACK);

		# Load the request into the $message structure
		$submessage = 0;
		while (<REQTRACK>) {
			if (/^-- New Action --/) {

				# New Reqtool action
				$submessage++;
				while (!/-- End Action --/) {
					$_ = <REQTRACK>;
					if (!/-- End Action --/) {
						@line = split (/: /, $_, 2);

						if ($line[1]) {
							chomp($line[1]);
							chomp($rest = $_);
							$rest =~ s/^(.*?): (.*)$/$2/g;
							$message[$submessage]->{ $line[0] }
= $rest;
						} else {
							$message[$submessage]->{'Comments'}
.= $_;
						}
					} ## end if (!/-- End Action --/...
				} ## end while (!/-- End Action --/...
			} ## end if (/^-- New Action --/...
		} ## end while (<REQTRACK>)
		close REQTRACK;

		# Set default type
		my $type = "text/html";
		$contents = "<center><h1>Request Number " . $reqtrack . "</h1></center>";

		# For each action on the request call the relevant function
		# to display the action
		foreach $sub (1 .. $submessage) {
			if ($message[$sub]->{'Action'} eq "Mail") {

				# Handle non-english emails
				if (($message[$sub]->{'Content-Type'})
					&& ($message[$sub]->{'Content-Type'} =~ /charset/))
				{
					$type = $message[$sub]->{'Content-Type'};
					$type =~ s/^.*charset=\"(.*)\".*/$1/g;
					$type = "text/html; charset=" . $type;
				} elsif ($message[$sub]->{'Subject'} =~ /^=\?ISO/) {
					$type = $message[$sub]->{'Subject'};
```

```
                                $type =~ s/^.*(ISO-[0-9]*-[A-Z]*).*$/$1/;
                                $type = "text/html; charset=" . $type;
                        }
                        $contents .= display_email($message[$sub], $request_dir,
$reqtrack);
                } elsif ($message[$sub]->{'Action'} eq "Give") {
                        $contents .= display_give($reqtrack, $message[$sub]);
                } elsif ($message[$sub]->{'Action'} eq "Resolve") {
                        $contents .= display_resolve($reqtrack, $message[$sub]);
                } elsif ($message[$sub]->{'Action'} eq "Merge") {
                        $contents .= display_merge($reqtrack, $message[$sub]);
                } elsif ($message[$sub]->{'Action'} eq "Comments") {
                        $contents .= display_comments($reqtrack, $message[$sub]);
                }
        } ## end foreach $sub (1 .. $submessage...

        $cookie = new CGI::Cookie(
                -name  => 'request',
                -value => $reqtrack,
                -path  => %gconfig->{'CGI_NAME'}
        );
        $header = header(
                -cookie => $cookie,
                -type   => $type
        );
        $contents .= "<script>\n";
        $contents .= "top.req_status.location.href=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=req_status\"\n";
        $contents .= "top.req_top.location.reload()\n";
        $contents .= "</script>\n";
        print_page("Request " . $req, $contents, $header);
    } ## end sub view_request
```

---

reqtool/send_attachment

NAME

send_attachment -- send a file to user

SYNOPSIS

send_attachment ($file)

FUNCTION

Sends a file to the user

INPUTS

$file - the filename of the file to send

OUTPUT

```
file as mime stream to stdout
```

SOURCE

```
sub send_attachment {
        my ($file) = @_;
        my ($att_header, $attach, $att_buffer, $extension, $mime_type);

        # Find attachment and read it into $att_buffer
        $attach = $file;
        $attach =~ s/(.*)\/(.*)$/$2/g;
        if (!open(ATTACH, $file)) {
                print_error("Couldn't open $file : $!\n");
        } else {
                lock_read(\*ATTACH);
                sysread(ATTACH, $att_buffer, (stat($file))[7]);
                close(ATTACH);
        }

        # Find the files mime-type
        if (open(MIME, %qconfig->{'MIME_TYPES'})) {
                lock_read(\*MIME);
                $extension = $attach;
                $extension =~ s/^(.*)\.(.*)$/$2/g;
                $mime_type = "";
                while ($_ = <MIME>) {

                        if (/$extension/) {
                                $_ =~ s/^(.*?)\s(.*)$/$1/g;
                                $mime_type = $_;
                                last;
                        }
                } ## end while ($_ = <MIME>)
                close MIME;

                if (!$mime_type) {
                        $mime_type = "application/octet-stream";
                }
        } else {
                if ($attach =~ /txt$/) {
                        $mime_type = "text/plain";
                } else {
                        $mime_type = "application/octet-stream";
                }
        } ## end else
        chomp $mime_type;

        # Send attachment
        $att_header = header(
                -disposition => 'attachment;filename=\"$attach\"',
                -type        => $mime_type
        );

        syswrite(STDOUT, $att_header, length($att_header));
```

```
        syswrite(STDOUT, $att_buffer, (stat($file))[7]);
} ## end sub send_attachment
```

---

reqtool/send_image

NAME

send_image -- Send an image to browser

SYNOPSIS

send_image ($image)

FUNCTION

Sends an image to browser which is used for command buttons

INPUTS

$image – name of image to send

OUTPUT

image as mime to stdout

SOURCE

```
sub send_image {
        my ($image) = @_;
        my ($file, $att_buffer, $att_header);

        $image = untaint_str ($image);

        # Convert image name to filename
        $file = %qconfig->{'IMAGE_PATH'} . "/" . $image . ".gif";

        # Read the image into buffer
        if (!open(ATTACH, $file)) {
                $att_buffer = "";
        } else {
                lock_read(\*ATTACH);
                sysread(ATTACH, $att_buffer, (stat($file))[7]);
                close(ATTACH);
        }

        # Send image to stdout
        $att_header = header(
                -expires     => '+3M',
                -disposition => 'attachment;filename=\"$image\".gif',
                -type        => 'image/gif'
```

```
        );

        syswrite(STDOUT, $att_header, length($att_header));
        syswrite(STDOUT, $att_buffer, (stat($file))[7]);
} ## end sub send_image
```

---

reqtool/date_diff

NAME

date_diff -- Return the difference between two dates

SYNOPSIS

```
$diff = date_diff ( $old, $new )
```

FUNCTION

Returns the difference between two dates in human readable form (ie 1 week)

INPUTS

```
$old - first date
$new - second date
```

OUTPUT

```
$diff - difference in human readable form
```

SOURCE

```
sub date_diff {
        my ($old, $new) = @_;
        my ($diff, $minute, $hour, $day, $week, $month, $year, $s, $string);

        $diff = $new - $old;

        # Define measures in seconds
        $minute = 60;
        $hour   = 60 * $minute;
        $day    = 24 * $hour;
        $week   = 7 * $day;
        $month  = 4 * $week;
        $year   = 356 * $day;

        # Convert difference in seconds to human readable form
        if ($diff < $minute) {
                $s      = $diff;
                $string = "sec";
        } elsif ($diff < $hour) {
```

```
                $s      = int($diff / $minute);
                $string = "min";
        } elsif ($diff < $day) {
                $s      = int($diff / $hour);
                $string = "hr";
        } elsif ($diff < $week) {
                $s      = int($diff / $day);
                $string = "day";
        } elsif ($diff < $month) {
                $s      = int($diff / $week);
                $string = "wk";
        } elsif ($diff < $year) {
                $s      = int($diff / $month);
                $string = "mth";
        } else {
                $s      = int($diff / $year);
                $string = "yr";
        }

        # Handle plurals
        if ($s > 1) {
                $string .= "s";
        }

        return "$s $string";
} ## end sub date_diff
```

---

reqtool/req_rescan

NAME

req_rescan -- Display a table of all active requests

SYNOPSIS

req_rescan ($req)

FUNCTION

Display a table of all active requests

INPUTS

$req   - current selected request number

OUTPUT

Calls print_page to display table

SOURCE

```
    sub req_rescan {
            my ($req) = @_;
            my ($contents, @request, $age, $header);

            # First make sure index exists
            if (!open(INDEX, %qconfig->{'ACTIVE_INDEX'})) {
                    $contents = "<center><h1>No Requests Exist</h1></center>\n";
            } else {
                    lock_read(\*INDEX);

                    # Table header
                    $contents = "<center>\n<table border=2 cellpadding=5>\n";
                    $contents .=
                      "<tr><th colspan=6 bgcolor=#ffffcc>Queue == active</th></tr>\n";
                    $contents .=
    "<tr
bgcolor=#99ffcc><td><b>Request</b></td><td><b>Owner</b></td><td><b>Age</b></td><td><b>S
tatus</b></td><td><b>From</b></td><td><b>Subject</b></td></tr>\n";

                    # First item is a form item to create a new request
                    $contents .= "<tr><td bgcolor=#ccccff align=center colspan=6><a
href=\""
                      . %qconfig->{'CGI_NAME'}
                      . "?command=req_mail&req=0"
                      . "\" target=\"req_bottom\">Create new request</a></td></tr>\n";

                    # Parse the index file and display all open requests
                    while (<INDEX>) {
                            @request = split (/\|/, $_, 6);
                            $contents .= "<tr>\n";
                            my $reqtrack = $req;
                            $reqtrack =~ s/R//g;
                            if ($request[0] == $reqtrack) {
                                    $contents .= "<td bgcolor=#ccccff align=right><a
href=\""
                                      . %qconfig->{'CGI_NAME'}
                                      . "?command=view&req="
                                      . $request[0]
                                      . "\" target=\"req_bottom\">"
                                      . $request[0]
                                      . "</a></td>\n";
                            } else {
                                    $contents .= "<td align=right><a href=\""
                                      . %qconfig->{'CGI_NAME'}
                                      . "?command=view&req="
                                      . $request[0]
                                      . "\" target=\"req_bottom\">"
                                      . $request[0]
                                      . "</a></td>\n";
                            } ## end else

                            if ($request[1] eq "none") {
                                    $contents .=
                                      "<td><font color=red>" . $request[1] .
"</font></td>\n";
                            } else {
                                    $contents .= "<td>" . $request[1] . "</td>\n";
```

```
                        }
                        $age = date_diff($request[2], time);
                        $contents .= "<td>" . $age . "</td>\n";
                        $contents .= "<td>" . $request[3] . "</td>\n";
                        $contents .= "<td>" . $request[4] . "</td>\n";
                        # Handle non-english subject
                        $request[5] =~ s/(\?\=)\r?\n[\040\t]+(\=\?)/$1$2/sg;
                        $request[5] =~
s/\=\?([\w\-]+)\?B\?([A-Za-z0-9\+\/]+\=*)\?\=/&un_base64($2,$1)/gex;

                        $contents .= "<td>" . $request[5] . "</td>\n";
                        $contents .= "</tr>\n";
                } ## end while (<INDEX>)
                $contents .= "</table>\n</center>\n";
        } ## end else

        # Make it refresh every 5 minutes
        $header = header(
                -Refresh => '300; URL='
                  . %qconfig->{'CGI_NAME'}
                  . '?command=req_rescan',
                -type => "text/html; charset=" . %qconfig->{'CHARSET'}
        );
        print_page("List Requests", $contents, $header);
} ## end sub req_rescan
```

---

reqtool/display_buttons

NAME

display_buttons -- Display the command buttons frame

SYNOPSIS

display_buttons

FUNCTION

Outputs the frame containing the command buttons

INPUTS

No output

OUTPUT

Calls print_page to output HTML page of buttons

SOURCE

```
sub display_buttons {
        my ($contents, $button);

        $contents = "<center>\n";
        foreach $button (@button_order) {
                $contents .= "<a href=\""
                  . %qconfig->{'CGI_NAME'}
                  . "?command=req_"
                  . $button
                  . "\" target="
                  . $buttons{$button}[1]
                  . "><img src=\""
                  . %qconfig->{'CGI_NAME'}
                  . "?command=getimage&image="
                  . $button
                  . "\" border=0 alt=\""
                  . $buttons{$button}[0]
                  . "\"></a><br>\n";
        } ## end foreach $button (@button_order...
        $contents .= "</center>\n";

        print_page("Commands", $contents);
} ## end sub display_buttons
```

---

reqtool/display_frames

NAME

display_frames -- Setup the site frames

SYNOPSIS

display_frames

FUNCTION

Output to STDOUT the frameset

INPUTS

No output

OUTPUT

HTML describing frameset

SOURCE

```
sub display_frames {
        print "Content-type: text/html\n\n";
```

```
        print "<html>\n";
        print "<head><title>Reqtool for "
          . %gconfig->{'LISTNAME'}
          . "</title></head>\n"
          . "<frameset rows=\"40%%,*\">\n";
          . "   <frame name=req_top src=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=list\">\n";
          . "   <frameset cols=\"125,*\">\n";
          . "           <frame name=req_command src=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=buttons\">\n";
          . "           <frameset rows=\"*,30\">\n";
          . "                   <frame name=req_bottom src=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=blank\">\n";
          . "                   <frame name=req_status src=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=req_status\">\n";
          . "           </frameset>\n";
          . "   </frameset>\n";
          . "</frameset>\n";
          . "</html>\n";
} ## end sub display_frames
```

---

reqtool/display_login

NAME

display_login -- Request Login information

SYNOPSIS

display_login

FUNCTION

Displays a login screen that varies depending on whether LDAP will be used

INPUTS

No output

OUTPUT

Calls print_page to display login screen

SOURCE

```
 sub display_login {
```

```
            my ($contents);

            $contents = "<center>\n";
            if (%qconfig->{'LDAP'}) {
                    $contents .= "<h2>To login enter your ldap
login/password</h2><br>\n";
            } else {
                    $contents .= "<h2>To login enter your unix
login/password</h2><br>\n";
            }
            $contents .= "</center>\n";
            $contents .=
              "<form action=\"" . %qconfig->{'CGI_NAME'} . "\" method=post>\n";
            $contents .= "<center><table>\n";
            $contents .=
              "<tr><td>Login:</td><td><input type=text size=10
name=login></td></tr>\n";
            $contents .=
    "<tr><td>Password:</td><td><input type=password size=10
name=password></td></tr>\n";
            $contents .= "</table>\n";
            $contents .= "<input type=submit name=loginSubmit value=\"Login\">\n";
            $contents .= "</center>\n";
            $contents .= "<input type=hidden name=command value=login>\n";
            $contents .= "</form>\n";
            print_page("Login to Reqtool", $contents);
    } ## end sub display_login
```

---

reqtool/login_check

NAME

login_check -- Check login/password and act accordingly

SYNOPSIS

login_check ($login, $password)

FUNCTION

Check login/password with passwd/LDAP and kickout or allow in

INPUTS

```
$login    - login name
$password - password
```

OUTPUT

Calls display_welcome or kicks person away

SOURCE

```
    sub login_check {
            my ($login, $password) = @_;
            my ($passwd, $salt);
            if (%qconfig->{'LDAP'}) {
                    my $LDAP = Net::LDAP->new(%qconfig->{'LDAP_SERVER'}) || die "$@";
                    my $mesg =
                      $LDAP->bind("uid=" . $login . ", " . %qconfig->{'LDAP_BIND'},
                            password => $password);
                    if ($mesg->code) {
                            print_error("Your Password/Login was incorrect");
                    } else {
                            $mesg = $LDAP->search(
                                    base   => %qconfig->{'LDAP_BASE'},
                                    attrs  => ['cn'],
                                    filter => "(uid=" . $login . ")"
                            );
                            display_welcome($login, $mesg->entry(0)->get_value("cn"));
                    } ## end else
                    $LDAP->unbind;
            } else {
                    (undef, $passwd, undef) = getpwnam($login);
                    if ($passwd) {
                            $salt = $passwd;
                            $salt =~ s/(..).*/$1/g;
                            if (crypt($password, $salt) eq $passwd) {

                                    #you're in
                                    display_welcome($login, $login);
                            } else {
                                    print_error("Your Password/Login was incorrect");
                            }
                    } else {
                            print_error("Your Password/Login was incorrect");

                    }
            } ## end else
    } ## end sub login_check
```

---

reqtool/display_welcome

NAME

display_welcome -- Display the welcome screen

SYNOPSIS

display_welcome ($login, $name)

FUNCTION

Temporary screen to welcome user and set required cookie

INPUTS

```
$login - login name
$name  - full name
```

OUTPUT

Calls print_page to display HTML

SOURCE

```
sub display_welcome {
        my ($login, $name) = @_;
        my ($contents, $cookie, $cookie2, $header);

        $contents =
          "<center><h1>Welcome " . $name . " to Reqtool</h1></center><br>\n";
        $contents .=
"<center><h2>The main screen will now load automatically</h2></center><br>\n";
        $contents .= "<center><h2>Click <a href=\""
          . %qconfig->{'CGI_NAME'}
          . "\">here</a> to go to main screen if it doesn't</h2></center><br>\n";
        $cookie = new CGI::Cookie(
                -name  => 'login',
                -value => $login,
                -path  => %qconfig->{'CGI_NAME'}
        );
        $header = header(
                -cookie  => $cookie,
                -Refresh => '1; URL=' . %qconfig->{'CGI_NAME'},
                -type    => "text/html; charset=" . %qconfig->{'CHARSET'}
        );
        print_page("Welcome to Reqtool", $contents, $header);
} ## end sub display_welcome
```

---

reqtool/take_request

NAME

take_request -- Take a request

SYNOPSIS

take_request ($req, $login)

FUNCTION

Calls req_give_do to give request to caller

INPUTS

```
$req   – request number
$login – callers name
```

OUTPUT

```
No output
```

SOURCE

```
sub take_request {
        my ($req, $login) = @_;
        req_give_do($req, $login, $login);
}
```

---

reqtool/req_give_do

NAME

req_give_do -- Give request to a person

SYNOPSIS

req_give_do -- ($req, $username, $login)

FUNCTION

Gives a request to a person, action by another
(may be same, see take_request

INPUTS

```
$req      – request number
$username – user to give to
$login    – user given by
```

OUTPUT

Calls print_page to display confirmation

SOURCE

```
sub req_give_do {
        my ($req, $username, $login) = @_;
        my ($request_dir, $contents, @line);
```

```
# Find fullname
if (%qconfig->{'LDAP'}) {
        my $LDAP = Net::LDAP->new(%qconfig->{'LDAP_SERVER'}) || die "$@";
        my $mesg = $LDAP->bind(%qconfig->{'LDAP_BIND'});
        $mesg = $LDAP->search(
                base   => %qconfig->{'LDAP_BASE'},
                attrs  => ['cn'],
                filter => "(uid=" . $username . ")"
        );
        $username = $mesg->entry(0)->get_value("cn");
        $mesg     = $LDAP->search(
                base   => %qconfig->{'LDAP_BASE'},
                attrs  => ['cn'],
                filter => "(uid=" . $login . ")"
        );
        $login = $mesg->entry(0)->get_value("cn");
        $LDAP->unbind();
} ## end if (%qconfig->{'LDAP'}...

$req         = untaint_num($req);

# Add entry to request contents
$request_dir = %qconfig->{'ACTIVE_DIR'} . "/" . $req;
open(REQTRACK, ">>$request_dir/request.contents")
  || print_error("Couldn't create $request_dir/request.contents");
lock_append(\*REQTRACK);
print REQTRACK "-- New Action --\n";
print REQTRACK "Action: Give\n";
print REQTRACK "Owner: " . $username . "\n";
print REQTRACK "Giver: " . $login . "\n";
print REQTRACK "Date: " . localtime(time) . "\n";
print REQTRACK "-- End Action --\n";
close REQTRACK;

# Update index by moving to tmp file then writing to new file
rename(%qconfig->{'ACTIVE_INDEX'}, %qconfig->{'ACTIVE_INDEX'} . "~")
  || print_error "Cannot rename: $!";
open(IN, "<" . %qconfig->{'ACTIVE_INDEX'} . "~")
  || print_error "Cannot open: $!";
lock_read(\*IN);
open(OUT, ">" . %qconfig->{'ACTIVE_INDEX'})
  || print_error "Cannot create: $!";
lock_append(\*OUT);

while ($_ = <IN>) {
        @line = split (/\|/, $_, 6);
        if ($line[0] == $req) {
                print OUT $line[0] . "|"
                  . $username . "|"
                  . $line[2] . "|"
                  . $line[3] . "|"
                  . $line[4] . "|"
                  . $line[5];
        } else {
                print OUT $_;
        }
} ## end while ($_ = <IN>)
```

```
        close OUT;
        close IN;

        # Output result of action to user
        $contents =
          "Request <font color=blue>" . $req
          . "</font> Given to <font color=blue>"
          . $username
          . "</font> by <font color=blue>"
          . $login
          . "</font>";
        $contents .= "<script>\n";
        $contents .= "top.req_top.location.href=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=req_rescan\"\n";
        $contents .= "top.req_bottom.location.reload()\n";
        $contents .= "</script>\n";
        print_page("Request Status", $contents);
} ## end sub req_give_do
```

---

reqtool/req_give

NAME

req_give -- Ask the user who to give request to

SYNOPSIS

req_give

FUNCTION

Displays a page to check who you want to give request to
List created from valid users file

INPUTS

No output

OUTPUT

Calls print_page to display combobox to choose user

SOURCE

```
 sub req_give {
        my ($contents, $username, $fullname);

        $contents = "<center><h1>Give Request</h1><br>\n";
        $contents .= "<form action=\""
```

```
              . %qconfig->{'CGI_NAME'}
              . "\" method=post target=req_status>\n";
            $contents .= "<h2>Who do you want to give it to?</h2><br>\n";
            $contents .= "<select name=username><br>\n";

            open(USERS, %qconfig->{'VALID_USERS'}) || print_error("No valid users");
            lock_read(\*USERS);

            my $LDAP;
            if (%qconfig->{'LDAP'}) {
                    $LDAP = Net::LDAP->new(%qconfig->{'LDAP_SERVER'}) || die "$@";
            }

            # Read list of valid users and convert login to fullname
            while ($_ = <USERS>) {
                    chomp($username = $_);
                    $username =~ s/
//g;
                    if (%qconfig->{'LDAP'}) {
                            my $mesg = $LDAP->search(
                                    base   => %qconfig->{'LDAP_BASE'},
                                    attrs  => ['cn'],
                                    filter => "(uid=" . $username . ")"
                            );
                            $fullname = $mesg->entry(0)->get_value("cn");
                    } else {
                            (undef, undef, undef, undef, undef, undef, $fullname) =
                              getpwnam($username);
                            if (!($fullname)) { $fullname = $username; }
                    }
                    $contents .= "<option value=\"" . $username . "\">" . $fullname .
"\n";
            } ## end while ($_ = <USERS>)
            if (%qconfig->{'LDAP'}) {
                    $LDAP->unbind();
            }
            close USERS;

            $contents .= "</select><br>\n";
            $contents .= "<input type=submit value=Give><br>\n";
            $contents .= "<input type=hidden name=command value=req_give_do>";

            print_page("Give Request", $contents);
    } ## end sub req_give
```

---

reqtool/req_comments

NAME

req_comments -- Get user comments

SYNOPSIS

req_comments

FUNCTION

Display a page requesting comments to add to request

INPUTS

No output

OUTPUT

Calls print_page to output HTML form

SOURCE

```
 sub req_comments {
        my ($contents);

        $contents = "<center><h1>Comments</h1>\n";
        $contents .= "<form action=\""
          . %gconfig->{'CGI_NAME'}
          . "\" method=post target=req_status>\n";
        $contents .= "<textarea cols=80 rows=10 name=comments></textarea><br>\n";
        $contents .= "<input type=submit value=\"Add Comments\">";
        $contents .= "<input type=hidden name=command value=req_comments_do>";

        print_page("Add Comments", $contents);
 } ## end sub req_comments
```

---

reqtool/req_comments_do

NAME

req_comments_do -- Add the comments to a request

SYNOPSIS

req_comments_do ($req, $login, $comments, $fullname)

FUNCTION

After receiving comments to add, actually add them to the request

INPUTS

```
$req      - request number
$login    - login name
```

```
$comments – comments to add
$fullname – fullname of user adding comments
```

OUTPUT

```
No output
```

SOURCE

```
sub req_comments_do {
        my ($req, $login, $comments, $fullname) = @_;
        my ($request_dir, $contents);

        $req         = untaint_num($req);
        $request_dir = %qconfig->{'ACTIVE_DIR'} . "/" . $req;
        open(REQTRACK, ">>$request_dir/request.contents")
          || print_error("Couldn't add comments");
        lock_append(\*REQTRACK);
        print REQTRACK "-- New Action --\n";
        print REQTRACK "Action: Comments\n";
        print REQTRACK "Date: " . localtime(time) . "\n";

        if ($fullname ne "") {
                print REQTRACK "Action-By: " . $fullname . "\n";
        } else {
                print REQTRACK "Action-By: " . $login . "\n";
        }
        print REQTRACK $comments . "\n";
        print REQTRACK "-- End Action --\n";
        close REQTRACK;

        $contents = "<script>\n";
        $contents .= "top.req_top.location.href=\""
          . %qconfig->{'CGI_NAME'}
          . "?command=req_rescan\"\n";
        $contents .= "top.req_bottom.location.href=\""
          . %qconfig->{'CGI_NAME'}
          . "?command=view&req="
          . $req . "\"\n";
        $contents .= "</script>\n";
        $contents .= "Comments Added";
        print_page("Comments sent", $contents);
} ## end sub req_comments_do
```

---

reqtool/req_mail_do

NAME

req_mail_do -- Send email

SYNOPSIS

```
   req_mail_do ( $req, $login, $address, $cc, $subject,
                 $mail, $filename, $tmpfile, $type, $encoding )
```

FUNCTION

```
   Send email to recipents as described in parameters
```

INPUTS

```
   $req      - request number
   $login    - login name
   $address  - email to send to
   $cc       - anyone to cc
   $subject  - subject of email
   $mail     - text of email
   $filename - sent filename of any attachment
   $tmpfile  - where the file is now
   $type     - what type of file is it
   $encoding - how is the file encoded
```

OUTPUT

```
   Email sent via sendmail, HTML displayed via print_page
```

SOURCE

```
    sub req_mail_do {
            my (
                    $req,  $login,    $address, $cc,   $subject,
                    $mail, $filename, $tmpfile, $type, $encoding
              ) = @_;
            my ($from, $top, $fullname, $contents);

            # If this is not a new request then automagically seyt subject
            if ($req ne "0") {
                    $subject =
                      "[ " . %qconfig->{'LISTNAME'} . " #" . $req . " ] " . $subject;
            }

            # Get fullname and email from LDAP if available
            # Otherwise use email as login@hostname
            if (%qconfig->{'LDAP'}) {
                    my $LDAP = Net::LDAP->new(%qconfig->{'LDAP_SERVER'}) ||
print_error($@);
                    my $mesg = $LDAP->bind(%qconfig->{'LDAP_BIND'});
                    $mesg = $LDAP->search(
                            base   => "ou=people, dc=sun, dc=com",
                            filter => "(employeenumber=" . $login . ")"
                    );
                    $LDAP->unbind;
                    $from =
                      $mesg->entry(0)->get_value("cn") . " <"
                      . $mesg->entry(0)->get_value("mail") . ">";
            } else {
```

```
                (undef, undef, undef, undef, undef, undef, $fullname) =
                  getpwnam($login);
                $from = $fullname . " <" . $login . "@" . hostname . ">";
        }

        $top = MIME::Entity->build(
                From       => $from,
                To         => $address,
                Cc         => $cc,
                Subject    => $subject,
                'Reply-To' => %qconfig->{'MAILING_LIST'},
                Data       => $mail
        );

        if ($tmpfile ne "") {
                $top->attach(
                        Path     => $tmpfile,
                        Type     => $type,
                        Filename => $filename,
                        Encoding => $encoding
                );
        } ## end if ($tmpfile ne "")

        # Pipe the email via sendmail to email it on
        open(SENDMAIL, "|/usr/lib/sendmail -oi -t")
          || print_error("Can't fork for sendmail: $!\n");
        $top->print(\*SENDMAIL);
        close(SENDMAIL) || warn("sendmail didn't close nicely");

        open(A, ">/tmp/out");
        $top->print(\*A);
        close A;
        $contents = "<script>\n";
        $contents .= "top.req_top.location.href=\""
          . %qconfig->{'CGI_NAME'}
          . "?command=req_rescan\"\n";
        $contents .= "</script>\n";
        $contents .= "Mail sent";
        print_page("Mail sent", $contents);
} ## end sub req_mail_do
```

---

reqtool/req_resolve

NAME

req_resolve -- Resolve a given request

SYNOPSIS

req_resolve ($req, $login, $fullname)

FUNCTION

Resolves a request by moving it to the resolved queue

INPUTS

```
$req      - request number
$login    - login name
$fullname - full name
```

OUTPUT

Calls print_page to display status of resolve action

SOURCE

```
sub req_resolve {
        my ($req, $login, $fullname) = @_;
        my ($request_dir, $contents, @line);

        # Don't allow this if user is read-only user
        if ($login ne %qconfig->{'READ_USER'}) {
                $request_dir = %qconfig->{'ACTIVE_DIR'} . "/" . $req;

                # Check for existance of resolved directory, otherwire create it
                if (!-e %qconfig->{'RESOLVED_DIR'}) {
                        mkdir(%qconfig->{'RESOLVED_DIR'}, 0775);
                }

                # Make sure the directory is a valid path
                if ($request_dir =~ /^([-\@\w.\/]+)$/) {
                        $request_dir = $1;
                } else {
                        print_error("Bad path data");
                }

                # Add action to request contents
                open(REQTRACK, ">>$request_dir/request.contents")
                  || print_error("Couldn't open request contents");
                lock_append(\*REQTRACK);
                print REQTRACK "-- New Action --\n";
                print REQTRACK "Action: Resolve\n";
                print REQTRACK "Date: " . localtime(time) . "\n";
                if ($fullname ne "") {
                        print REQTRACK "Action-By: " . $fullname . "\n";
                } else {
                        print REQTRACK "Action-By: " . $login . "\n";
                }
                print REQTRACK "-- End Action --\n";
                close REQTRACK;

                # Move the request to the resolved directory
                system("mv " . $request_dir . " " . %qconfig->{'RESOLVED_DIR'});

                # Update active/resolved indexes
                system("cp "
                          . %qconfig->{'ACTIVE_INDEX'} . " "
```

```
                . %qconfig->{'ACTIVE_INDEX'}
                . "~");
        open(IN, "<" . %qconfig->{'ACTIVE_INDEX'} . "~")
          || print_error "Cannot open: $!";
        lock_read(\*IN);
        open(OUT, "+<" . %qconfig->{'ACTIVE_INDEX'})
          || print_error "Cannot create: $!";
        lock_write(\*OUT);
        open(RINDEX, ">>" . %qconfig->{'RESOLVED_INDEX'})
          || print_error("Couldn't open resolved index");
        lock_append(\*RINDEX);

        while ($_ = <IN>) {
                @line = split (/\|/, $_, 6);
                if ($line[0] == $req) {
                        if ($line[1] eq "none") {
                                $line[1] = $login;
                        }
                        print RINDEX $line[0] . "R|"
                          . $line[1] . "|"
                          . $line[2]
                          . "|resolved|"
                          . $line[4] . "|"
                          . $line[5];
                } else {
                        print OUT $_;
                }
        } ## end while ($_ = <IN>)
        close RINDEX;
        close OUT;
        close IN;
        unlink "%qconfig->{'ACTIVE_INDEX'}~";

        # Display results of action to user
        if ($login eq "dropped") {
                $contents = "<script>\n";
                $contents .= "top.req_top.location.href=\""
                  . %qconfig->{'CGI_NAME'}
                  . "?command=req_rescan\"\n";
                $contents .= "top.req_bottom.location.reload()\n";
                $contents .= "</script>\n";
                $contents .= "Request " . $req . " dropped";

                print_page("Dropped", $contents);
        } else {
                $contents = "<script>\n";
                $contents .= "top.req_top.location.href=\""
                  . %qconfig->{'CGI_NAME'}
                  . "?command=req_rescan\"\n";
                $contents .= "top.req_bottom.location.reload()\n";
                $contents .= "</script>\n";
                $contents .= "Request <font color=blue>" . $req
                  . "</font> resolved by "
                  . $login;
                print_page("Resolved", $contents);
        } ## end else
} else {
```

```
                print_page("Error", "Sorry you are not allowed to do this
operation");
            }
    } ## end sub req_resolve
```

---

reqtool/req_search

NAME

req_search -- Display Search for a request screen

SYNOPSIS

req_search

FUNCTION

Displays a form to allow user to search for a request

INPUTS

No output

OUTPUT

Calls print_page to display form

SOURCE

```
    sub req_search {
            my ($contents);

            $contents = "<form action=" . %qconfig->{'CGI_NAME'} . " method=post>\n";
            $contents .= "<center><h1>Search for request</h1>\n";
            $contents .= "<b>Search by</b>\n";
            $contents .= "<select name=searchby>\n";
            $contents .= "<option value=all>All\n";
            $contents .= "<option value=num>Number\n";
            $contents .= "<option value=subject>Subject\n";
            $contents .= "<option value=email>eMail\n";
            $contents .= "</select>\n";
            $contents .= "<INPUT TYPE=checkbox NAME=active CHECKED>active\n";
            $contents .= "<INPUT TYPE=checkbox NAME=resolved CHECKED>resolved<br>\n";
            $contents .= "<b>Search for</b>\n";
            $contents .= "<input type=text size=20 name=query><br>\n";
            $contents .= "<input type=hidden name=command value=req_search_do>\n";
            $contents .= "</form>\n";
            print_page("Search", $contents);
    } ## end sub req_search
```

reqtool/req_search_do

NAME

req_search_do -- Search for a request and return results

SYNOPSIS

req_search_do ($query, $searchby, $active, $resolved)

FUNCTION

Search active/resolved queues for a request by calling search_file

INPUTS

```
$query    - search term
$searchby - what category to search by
$active   - boolean saying whether to search active queue
$resolved - boolean saying whether to search resolved queue
```

OUTPUT

Calls print_page to display search results

SOURCE

```
    sub req_search_do {
            my ($query, $searchby, $active, $resolved) = @_;
            my ($contents, $results);

            $contents = "<center>\n<table border=2 cellpadding=5>\n";
            $contents .=
              "<tr><th colspan=6 bgcolor=#ffffcc>Search for " . $query . " by "
              . $searchby
              . "</th></tr>\n";
            $contents .=
    "<tr
bgcolor=#99ffcc><td>Request</td><td>Owner</td><td>Age</td><td>Status</td><td>From</td><
td>Subject</td></tr>\n";

            # Check wheter we are searching resolved requests only
            if ($active eq "resolved") {
                    undef $active;
                    $resolved = "on";
            }

            if ($active) {
                    $results = search_file(%qconfig->{'ACTIVE_INDEX'}, $query,
$searchby);
```

```
                if ($results) {
                        $contents .= $results;
                        $results = "";
                }
        } ## end if ($active)

        if ($resolved) {
                $results = search_file(%qconfig->{'RESOLVED_INDEX'}, $query,
$searchby);
                if ($results) {
                        $contents .= $results;
                        $results = "";
                }
        } ## end if ($resolved)

        $contents .= "</table>\n</center>\n";

        print_page("Search", $contents);
} ## end sub req_search_do
```

---

reqtool/admin_forward

NAME

admin_forward -- Edit forwards

SYNOPSIS

admin_forward

FUNCTION

Displays a form that is used to edit the forwards

INPUTS

No output

OUTPUT

Calls print_page to display form

SOURCE

```
 sub admin_forward {
        my ($contents, $array_values, $count);

        $contents = "<script language=\"JavaScript\">\n";
        $contents .= "<!--\n";
```

```
            $array_values = "Forward[0] = \"\"\n";
            $array_values .= "ForwardEmail[0] = \"\"\n";
            $array_values .= "ForwardName[0] = new Option (\"Add New\",\"0\")\n";
            close REQTRACK;

            if (!open(FORWARD, %qconfig->{'FORWARD_FILE'})) {
                    $count = 0;
            } else {
                    lock_read(\*FORWARD);
                    $count = <FORWARD>;
                    chomp($count);

                    while (<FORWARD>) {
                            $array_values .= $_;
                    }
                    close FORWARD;
            } ## end else

            $contents .= "var Forward = new Array(" . $count . ")\n";
            $contents .= "var ForwardEmail = new Array(" . $count . ")\n";
            $contents .= "var ForwardName = new Array(" . $count . ")\n";
            $contents .= $array_values;
            $contents .= "function add_to_body() { // add selected contents to body\n";
            $contents .=
    "       document.email.current.value = document.email.addme.options[document.email.
addme.selectedIndex].text\n";
            $contents .=
    "       document.email.mail.value = Forward[document.email.addme.options[document.
email.addme.selectedIndex].value]\n";
            $contents .=
    "       document.email.address.value = ForwardEmail[document.email.addme.
options[document.email.addme.selectedIndex].value]\n";
            $contents .= "}\n";
            $contents .= "\n";
            $contents .= "// -->\n";
            $contents .= "</script>\n";
            $contents .= "\n";
            $contents .= "\n";
            $contents .= "<center><form action="
              . %qconfig->{'CGI_NAME'}
              . " method=get name=email>\n";
            $contents .= "<table border=0 cols=3>\n";
            $contents .=
    "<tr><td width=100><b>Send email to:</b></td><td colspan=2><input type=text
name=address size=60></td></tr>\n";
            $contents .=
    "<tr><td colspan=3><textarea cols=80 rows=20 name=mail></textarea></td></tr>\n";
            $contents .= "<tr><td><input type=submit value=\"Update Files\"></td>";
            $contents .= "<td></td>";
            $contents .=
    "<td><select name=addme onChange=\"javascript:add_to_body()
\"></select></td></tr>\n";
            $contents .= "</table>";
            $contents .= "<input type=hidden name=command value=admin>\n";
            $contents .= "<input type=hidden name=admin value=admin_forward_do>\n";
            $contents .= "<input type=hidden name=current value=\"Add New\">\n";
            $contents .= "</form></center>\n";
```

```
        $contents .= "\n";
        $contents .= "<script language=\"JavaScript\">\n";
        $contents .= "<!--\n";
        $contents .= "for (var i=0; i <= " . $count . " ; i++) {\n";
        $contents .= "  document.email.addme.options[i]=ForwardName[i]\n";
        $contents .= "  document.email.addme.selectedIndex=0\n";
        $contents .= "}\n";
        $contents .= "// -->\n";
        $contents .= "</script>\n";

        print_page("Admin - Update Forwards", $contents);
} ## end sub admin_forward
```

---

reqtool/admin_forward_do

NAME

admin_forward_do -- Update a forward

SYNOPSIS

admin_forward_do ($address, $body, $option)

FUNCTION

Update the forward file with data given

INPUTS

```
$address - email address of forward
$body    - text of forward
$option  - option number of forward
```

OUTPUT

Calls print_page to display result of update

SOURCE

```
 sub admin_forward_do {
        my ($address, $body, $option) = @_;
        my ($contents, $count, $output, $forward, $email);

        if ($option eq "Add New") {
                $contents =
                  "<form action=" . %qconfig->{'CGI_NAME'} . " method=post>\n";
                $contents .=
"<b>Enter Forward name</b><input type=text name=current size=20><br>\n";
                $contents .=
                  "<b>Address</b><input type=text name=address value=\"" . $address
```

```
                    . "\"><br>\n";
                $contents .=
                  "<textarea name=mail rows=10 cols=80>" . $body .
"</textarea><br>\n";
                $contents .= "<input type=submit value=\"Submit New Forward\">\n";
                $contents .= "<input type=hidden name=command value=admin>\n";
                $contents .= "<input type=hidden name=admin
value=admin_forward_do>\n";
                $contents .= "</form>\n";
                print_page("Add new forward", $contents);
        } else {

                # If no forward dir exists then create it
                if (!-e %qconfig->{'FORWARD_DIR'}) {
                        mkdir(%qconfig->{'FORWARD_DIR'}, 0775);
                }
                if ($option =~ /(.*)/) {
                        $option = $1;
                } else {
                        print_error("No name given");
                }

                # Create forward option
                open(FORWARD, ">" . %qconfig->{'FORWARD_DIR'} . "/" . $option)
                  || print_error("Couldn't create $option: $!");
                lock_append(\*FORWARD);
                print FORWARD $address . "\n";
                chomp $body;
                $body =~ s/
//g;
                print FORWARD $body . "\n";
                close FORWARD;

                opendir(DIR, %qconfig->{'FORWARD_DIR'})
                  || print_error("can't opendir " . %qconfig->{'FORWARD_DIR'} . ":
$!");
                if (!open(FORWARD, "+<" . %qconfig->{'FORWARD_FILE'})) {
                        open(FORWARD, ">" . %qconfig->{'FORWARD_FILE'});
                        lock_append(\*FORWARD);
                } else {
                        lock_write(\*FORWARD);
                }
                $count  = 0;
                $output = "";

                # Rebuild forward html from directory contents
                while ($forward = readdir(DIR)) {
                        if ($forward =~ /^\./) {
                                # ignore this as it is hidden file
                        } else {
                                $count++;
                                open(FILE, %qconfig->{'FORWARD_DIR'} . "/" .
$forward)
                                  || print_error("permission denied");
                                lock_read(\*FILE);
                                $output .= "ForwardName[" . $count
                                  . "] = new Option(\""
```

```
                                        . $forward . "\",\""
                                        . $count . "\")\n";
                                    $output .= "ForwardEmail[" . $count . "] = \"";
                                    $email = <FILE>;
                                    chomp($email);
                                    $output .= $email . "\"\n";
                                    $output .= "Forward[" . $count . "] = \"";

                                    while (<FILE>) {
                                            chomp;
                                            $_ =~ s/\"/\\\"/g;
                                            $output .= $_ . "\\n";
                                    }
                                    $output .= "\"\n";
                                    close FILE;
                            } ## end else
                    } ## end while ($forward = readdir...
                    print FORWARD $count . "\n";
                    print FORWARD $output;
                    close FORWARD;
                    close DIR;

                    # Return results to administrator
                    $contents = "<center><h1>Forward updated/Added</h1></center>\n";
                    $contents .= "<table border=1>\n";
                    $contents .=
                      "<tr><td><b>Forward name:</b></td><td>" . $option .
"</td></tr>\n";
                    $contents .=
                      "<tr><td><b>Address:</b></td><td>" . $address . "</td></tr>\n";
                    $contents .= "<tr><td colspan=2><pre wrap>" . $body .
"</pre></td>\n";
                    $contents .= "</table>\n";
                    $contents .= "<h2>Return to <a href=\""
                      . %qconfig->{'CGI_NAME'}
                      . "?command=admin\">Menu</a><h2>\n";
                    print_page("Added forward", $contents);
            } ## end else
    } ## end sub admin_forward_do
```

---

reqtool/req_mail

NAME

req_mail -- Send an email via reqtool

SYNOPSIS

req_mail ($req)

FUNCTION

```
   Displays a form to allow sending of an email
   Automatically fills some fields in from given request
   Can add attachment
```

INPUTS

```
   $req - request number ( 0 if new email )
```

OUTPUT

```
   Calls print_page to display form
```

SOURCE

```
    sub req_mail {
            my ($req) = @_;
            my (
                    $contents, $request_dir, $array_values,
                    @line,     $from,        $path,
                    $body,     $text,        $count
            );
            my $subject = "";
            $contents = "<script language=\"JavaScript\">\n";
            $contents .= "<!--\n";

            # Load request for automatic creation of forward
            if ($req =~ /R/) {
                    $req =~ s/^(.*)R$/$1/g;
                    $request_dir = %qconfig->{'RESOLVED_DIR'} . "/" . $req;
            } else {
                    $request_dir = %qconfig->{'ACTIVE_DIR'} . "/" . $req;
            }
            if ($req > 0) {
                    open(REQTRACK, "$request_dir/request.contents")
                      || print_error("Couldn't open $request_dir/request.contents");
                    lock_read(\*REQTRACK);

                    $array_values .= "Forward[0] = \"\\n--- Message Included ---\\n";

                    while (<REQTRACK>) {
                            chomp;
                            if ((/^To: /) || (/^From: /) || (/^Subject: /) || (/^Cc:
/)) {
                                    $_ =~ s/\"/\\\"/g;
                                    $array_values .= $_ . "\\n";

                                    if (/^From: /) {
                                            @line = split (/: /, $_, 2);
                                            $from = $line[1];
                                    } elsif (/^Subject: /) {
                                            @line = split (/: /, $_, 2);
                                            $subject = $line[1];
                                    }
                            } elsif (/^Path: /) {
                                    @line = split (/: /, $_, 2);
```

```
                                $path = $line[1];
                        } elsif (/^Body: /) {
                                @line = split (/: /, $_, 2);
                                $body = $line[1];
                                open(BODY, "$request_dir/$path/$body")
                                  || print_error("can't open body");
                                lock_read(\*BODY);

                                while (<BODY>) {
                                        chomp;
                                        $text = $_;
                                        $text =~ s/\"/\\\"/g;
                                        $text =~ s/$/\\n/g;
                                        $array_values .= $text;
                                } ## end while (<BODY>)
                                close BODY;
                                last;
                        } ## end elsif (/^Body: /)
                } ## end while (<REQTRACK>)
                $array_values .= "\"\n";
                $array_values .= "ForwardEmail[0] = \"" . $from . "\"\n";
                $array_values .=
                  "ForwardName[0] = new Option(\"Message Body\",\"0\")\n";
                close REQTRACK;
        } ## end if ($req > 0)

        # Add pre-defined forwards
        open(FORWARD, %qconfig->{'FORWARD_FILE'})
          || print_error("couldn't open " . %qconfig->{'FORWARD_FILE'});
        lock_read(\*FORWARD);
        $count = <FORWARD>;
        chomp($count);

        while (<FORWARD>) {
                $array_values .= $_;
        }
        close FORWARD;

        # HTML source from here on
        $contents .= "var Forward = new Array(" . $count . ")\n";
        $contents .= "var ForwardEmail = new Array(" . $count . ")\n";
        $contents .= "var ForwardName = new Array(" . $count . ")\n";
        $contents .= $array_values;
        $contents .= "function add_to_body() { // add selected contents to body\n";
        $contents .=
    "       document.email.mail.value = document.email.mail.value + Forward[document.
email.addme.options[document.email.addme.selectedIndex].value]\n";
        $contents .= "  if (document.email.address.value == \"\") {\n";
        $contents .=
    "               document.email.address.value = document.email.address.value +
ForwardEmail[document.email.addme.options[document.email.addme.selectedIndex].value]
\n";
        $contents .= "  } else {\n";
        $contents .=
    "               document.email.address.value = document.email.address.value + \",
\" + ForwardEmail[document.email.addme.options[document.email.addme.selectedIndex].
value]\n";
```

```
            $contents .= "  }\n";
            $contents .= "}\n";
            $contents .= "\n";
            $contents .= "// -->\n";
            $contents .= "</script>\n";
            $contents .= "\n";
            $contents .= "\n";
            $contents .= "<center><form action="
              . %qconfig->{'CGI_NAME'}
              . " method=post name=email enctype=\"multipart/form-data\"
target=req_status>\n";
            $contents .= "<table border=0 cols=3>\n";
            $contents .=
    "<tr><td width=100>To:</td><td colspan=2><input type=text name=address
size=60></td></tr>\n";
            $contents .=
              "<tr><td>Cc:</td><td colspan=2><input type=text name=cc value=\""
              . %qconfig->{'MAILING_LIST'}
              . "\" size=60></td></tr>\n";
            $contents .=
    "<tr><td>Subject:</td><td colspan=2><input type=text name=subject size=60 value=\""
              . $subject
              . "\"></td></tr>\n";
            $contents .=
    "<tr><td>Attachment:</td><td colspan=2><input type=file name=attach
size=50></td></tr>\n";
            $contents .=
    "<tr><td colspan=3><textarea cols=80 rows=5 name=mail></textarea></td></tr>\n";
            $contents .= "<tr><td><input type=submit value=\"Send Email\"></td>";
            $contents .=
    "<td align=right valign=top><input type=button name=contents value=\"Add to message
body\" onClick=\"javascript:add_to_body()\"></td>";
            $contents .=
    "<td align=left valign=top><select name=addme></select>Resolve<input type=checkbox
name=resolve value=Resolve></td></tr>\n";
            $contents .= "</table>";
            $contents .= "<input type=hidden name=command value=req_mail_do>\n";
            $contents .= "</form></center>\n";
            $contents .= "\n";
            $contents .= "<script language=\"JavaScript\">\n";
            $contents .= "<!--\n";
            $contents .= "for (var i=0; i <= " . $count . " ; i++) {\n";
            $contents .= "  document.email.addme.options[i]=ForwardName[i]\n";
            $contents .= "  document.email.addme.selectedIndex=0\n";
            $contents .= "}\n";
            $contents .= "// -->\n";
            $contents .= "</script>\n";

            print_page("Get Email", $contents);
    } ## end sub req_mail
```

---

reqtool/search_file

NAME

search_file -- search a given file regarding query

SYNOPSIS

```
$contents = search_file ( $file, $query, $searchby )
```

FUNCTION

Search the index for requests matching the query

INPUTS

```
$file     - index file
$query    - query text
$searchby - fields to search by
```

OUTPUT

$contents - HTML source describing results

SOURCE

```
    sub search_file {
            my ($file, $query, $searchby) = @_;
            my ($req, @request, $contents, $age);

            # Open the file we need to search and then search with reference
            # to the fields that have been requested
            open(FILE, $file) || return 0;
            lock_read(\*FILE);

            while (<FILE>) {
                    $req = 0;
                    @request = split (/\|/, $_, 6);
                    if ($searchby =~ /num/) {
                            if (($request[0] eq $query) || ($request[0] eq $query .
"R")) {
                                    $req = $request[0];
                            }
                    } elsif ($searchby =~ /subject/) {
                            if ($request[5] =~ /$query/i) {
                                    $req = $request[0];
                            }
                    } elsif ($searchby =~ /email/) {
                            if ($request[4] =~ /$query/i) {
                                    $req = $request[0];
                            }
                    } else {
                            if (/$query/i) {
                                    $req = $request[0];
                            }
                    }
```

```
                # This request matches search criteria so display it
                if ($req) {
                        $contents .= "<tr>\n";
                        $contents .= "<td align=right><a href=\""
                          . %qconfig->{'CGI_NAME'}
                          . "?command=view&req="
                          . $request[0]
                          . "\" target=\"req_bottom\">"
                          . $request[0]
                          . "</td>\n";
                        $contents .= "<td>" . $request[1] . "</td>\n";
                        $age = date_diff($request[2], time);
                        $contents .= "<td>" . $age . "</td>\n";
                        $contents .= "<td>" . $request[3] . "</td>\n";
                        $contents .= "<td>" . $request[4] . "</td>\n";
                        $contents .= "<td>" . $request[5] . "</td>\n";
                        $contents .= "</tr>\n";
                } ## end if ($req)
        } ## end while (<FILE>)
        return $contents;
} ## end sub search_file
```

---

reqtool/req_parameters

NAME

req_parameters -- Login for Administration section

SYNOPSIS

req_parameters

FUNCTION

Displays login screen for administration section

INPUTS

No output

OUTPUT

Calls print_page to display login screen

SOURCE

```
 sub req_parameters {
        my ($contents);

        $contents = "<form action=" . %qconfig->{'CGI_NAME'} . " method=post>\n";
```

```
            $contents .= "<center><h1>Login to Administration section</h1><br>\n";
            $contents .=
              "<b>Administrator Password</b><input type=password name=password><br>\n";
            $contents .= "<input type=hidden name=command value=admin>\n";
            $contents .= "<input type=hidden name=admin value=main>\n";
            $contents .= "</form></center>\n";

            print_page("Login Admin", $contents);
    } ## end sub req_parameters
```

---

reqtool/admin_main

NAME

admin_main -- Main administration menu/screen

SYNOPSIS

admin_main ($password)

FUNCTION

Displays admin menu and sets cookie saying you're allowed here

INPUTS

$password

OUTPUT

Calls print_page to display menu and set cookie

SOURCE

```
    sub admin_main {
            my ($password) = @_;
            my ($contents, $cookie, $header);

            $contents = "<center><h1>Reqtool Administration</h1><br>\n";
            $contents .= "<h2>Choose your options</h2><br>\n";
            $contents .= "<form action=" . %qconfig->{'CGI_NAME'} . " method=post>\n";
            $contents .=
    "<input type=hidden name=command value=admin><input type=hidden name=admin
value=admin_forward>\n";
            $contents .= "<input type=submit value=\"Update/Add Forward\">\n";
            $contents .= "</form>\n";
            $contents .= "<form action=" . %qconfig->{'CGI_NAME'} . " method=post>\n";
            $contents .=
    "<input type=hidden name=command value=admin><input type=hidden name=admin
value=admin_users>\n";
```

```
            $contents .= "<input type=submit value=\"Update users\">\n";
            $contents .= "</form>\n";
            $contents .= "<form action=" . %qconfig->{'CGI_NAME'} . " method=post>\n";
            $contents .=
    "<input type=hidden name=command value=admin><input type=hidden name=admin
value=admin_index>\n";
            $contents .= "<input type=submit value=\"Refresh index\">\n";
            $contents .= "</form>\n";

            $cookie = new CGI::Cookie(
                    -name   => 'admin',
                    -value  => $password,
                    -path   => %qconfig->{'CGI_NAME'},
                    expires => '+3h'
            );
            $header = header(
                    -cookie => $cookie,
                    -type   => "text/html; charset=" . %qconfig->{'CHARSET'}
            );

            print_page("Administator Section", $contents, $header);
    } ## end sub admin_main
```

---

reqtool/admin_users

NAME

admin_users -- Edit valid users

SYNOPSIS

admin_users

FUNCTION

Form to edit who is allowed to own requests

INPUTS

No output

OUTPUT

Calls print_page to display form

SOURCE

```
    sub admin_users {
            my ($contents);
```

```
        $contents = "<center><h1>Update Valid Users</h1><br>\n";
        $contents .= "<form action=" . %qconfig->{'CGI_NAME'} . " method=post>\n";
        $contents .= "<textarea cols=40 rows=10 name=users>";
        if (open(USERS, %qconfig->{'VALID_USERS'})) {
                lock_read(\*USERS);

                while (<USERS>) {
                        $contents .= $_;
                }
                close USERS;
        } ## end if (open(USERS, %qconfig...
        $contents .= "</textarea><br>\n";
        $contents .= "<input type=submit value=\"Submit Changes\">\n";
        $contents .= "<input type=hidden name=command value=admin>\n";
        $contents .= "<input type=hidden name=admin value=admin_users_do>\n";
        $contents .= "</form>\n";

        print_page("Update Valid Users", $contents);
} ## end sub admin_users
```

---

reqtool/admin_users_do

NAME

admin_users_do -- Update the valid users file

SYNOPSIS

admin_users_do ($users)

FUNCTION

Update VALID_USERS to value given from admin_users screen

INPUTS

$users - list of valid users

OUTPUT

Calls print_page to display status of action

SOURCE

```
sub admin_users_do {
        my ($users) = @_;
        my ($contents);

        if (!open(USERS, "+<" . %qconfig->{'VALID_USERS'})) {
                open(USERS, ">" . %qconfig->{'VALID_USERS'})
```

```
                  || print_error("Can't open %qconfig->{'VALID_USERS'}: $!");
                lock_append(\*USERS);

        } else {
                lock_write(\*USERS);
        }
        $users =~ s/ /\n/g;
        $users =~ s/,/\n/g;
        $users =~ s/^$//g;
        print USERS $users;
        close USERS;

        $contents = "<center><h1>Updated Valid-Users</h1></center>\n";
        $contents .= "<h2>Return to <a href=\""
          . %qconfig->{'CGI_NAME'}
          . "?command=admin\">Menu</a><h2>\n";
        print_page("Updated Users", $contents);
} ## end sub admin_users_do
```

---

reqtool/req_merge

NAME

req_merge -- Merge two requests

SYNOPSIS

req_merge ($req)

FUNCTION

Ask user which two requests to merge

INPUTS

$req - initial request number

OUTPUT

Calls print_page to display result

SOURCE

```
sub req_merge {
        my ($req) = @_;
        my ($contents, $requests, @line);

        # Open the active index to find available requests
        # Current request is automatically selected
        open(INDEX, %qconfig->{'ACTIVE_INDEX'})
```

```
          || print_error("Couldn't open active index: $!");
        lock_read(\*INDEX);
        $requests = "";
        while (<INDEX>) {
                @line = split (/\|/, $_, 6);

                if ($line[0] == $req) {
                        $requests .= "<option value=\""
                          . $line[0]
                          . "\" selected>"
                          . $line[0] . " - "
                          . $line[5] . "\n";
                } else {
                        $requests .= "<option value=\""
                          . $line[0] . "\">"
                          . $line[0] . " - "
                          . $line[5] . "\n";
                } ## end else
        } ## end while (<INDEX>)
        close INDEX;

        $contents = "<center><h1>Merge Requests</h1></center><br>\n";
        $contents .=
          "<form action=\"" . %qconfig->{'CGI_NAME'} . "\" target=req_status>\n";
        $contents .= "<input type=hidden name=command value=req_merge_do>\n";
        $contents .= "<h2>Merge request</h2>\n";
        $contents .= "<select name=from_req>\n";
        $contents .= $requests;
        $contents .= "</select><br>\n";
        $contents .= "<h2>Into request</h2>\n";
        $contents .= "<select name=to_req>\n";
        $contents .= $requests;
        $contents .= "</select><br>\n";
        $contents .= "<input type=submit value=\"Merge\">\n";
        $contents .= "</form>\n";
        print_page("Merge Requests", $contents);
} ## end sub req_merge
```

---

reqtool/req_merge_do

NAME

req_merge_do -- Do the actual merge

SYNOPSIS

req_merge_do ($req_from, $req_to)

FUNCTION

Merge two requests

INPUTS

```
$req_from - request to merge from
$req_to   - request to merge into
```

OUTPUT

```
Calls print_page to display result
```

SOURCE

```
sub req_merge_do {
        my ($req_from, $req_to) = @_;
        my ($req_from_dir, $req_to_dir, $contents, $req_to_index, $req_from_index,
                @line);

        $req_to   = untaint_num($req_to);
        $req_from = untaint_num($req_from);

        $req_from_dir   = %qconfig->{'ACTIVE_DIR'} . "/" . $req_from;
        $req_to_dir     = %qconfig->{'ACTIVE_DIR'} . "/" . $req_to;
        $req_to_index   = $req_to_dir . "/request.contents";
        $req_from_index = $req_from_dir . "/request.contents";

        # Open both requests and write merge action to req_to
        open(REQ_FROM, $req_from_index)
          || print_error("Couldn't open From request: $!");
        lock_read(\*REQ_FROM);
        open(REQ_TO, ">>$req_to_index")
          || print_error("Couldn't open To request: $!");
        lock_append(\*REQ_TO);
        print REQ_TO "-- New Action --\n";
        print REQ_TO "Action: Merge\n";
        print REQ_TO "Request-Merged: " . $req_from . "\n";
        print REQ_TO "Date: " . localtime(time) . "\n";
        print REQ_TO "-- End Action --\n";

        # Copy contents of old request to new request
        while (<REQ_FROM>) {
                print REQ_TO $_;
        }
        close REQ_TO;
        close REQ_FROM;
        unlink $req_from_index;
        # Move all request data across
        system("/usr/bin/mv " . $req_from_dir . "/* " . $req_to_dir . "/");
        rmdir $req_from_dir;

        # Update active index
        system("cp "
                  . %qconfig->{'ACTIVE_INDEX'} . " "
                  . %qconfig->{'ACTIVE_INDEX'}
                  . "~");
        open(IN, "<" . %qconfig->{'ACTIVE_INDEX'} . "~")
          || print_error "Cannot open: $!";
```

```
        lock_read(\*IN);
        open(OUT, "+<" . %gconfig->{'ACTIVE_INDEX'})
          || print_error "Cannot create: $!";
        lock_write(\*OUT);

        while ($_ = <IN>) {
                @line = split (/\|/, $_, 2);
                if ($line[0] != $req_from) {
                        print OUT $_;
                }
        } ## end while ($_ = <IN>)
        close OUT;
        close IN;
        unlink "%gconfig->{'ACTIVE_INDEX'}~";

        # Display results of merge to user
        $contents =
          "Request <font color=blue>"
          . $req_from
          . "</font> merged into Request <font color=blue>"
          . $req_to
          . "</font>\n";
        $contents .= "<script>\n";
        $contents .= "top.req_top.location.href=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=req_rescan\"\n";
        $contents .= "top.req_bottom.location.href=\""
          . %gconfig->{'CGI_NAME'}
          . "?command=blank\"\n";
        $contents .= "</script>\n";

        print_page("Merge done", $contents);
} ## end sub req_merge_do
```

---

reqtool/request_stats

NAME

request_stats -- Get dates for statistics

SYNOPSIS

request_stats

FUNCTION

Displays a form for user to choose date range for statistics generation

INPUTS

No output

OUTPUT

Calls print_page to display form

SOURCE

```
    sub request_stats {
            my ($contents, $i);
            $contents = "<h1><center>Statistics</center></h1>\n";
            $contents .=
              "<form action=\"" . %qconfig->{'CGI_NAME'} . "\" method=post>\n";
            $contents .= "<input type=hidden name=command value=display_stats>\n";
            $contents .=
    "<center><b>Choose the start and end dates for which you wish to view statistical
information</b></center><br>\n";
            $contents .= "<center><table>";
            $contents .= "<tr><td><b>Start Date: </b></td><td>";
            $contents .= "<select name=sDay>";
            $contents .= "<option value=1 selected>1\n";

            for ($i = 2 ; $i <= 31 ; $i++) {
                    $contents .= "<option value=" . $i . ">" . $i . "\n";
            }
            $contents .= "</select></td>";
            $contents .= "<td><select name=sMonth>";
            $contents .= "<option value=Jan selected>Jan\n";

            foreach (
                    'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
                    'Aug', 'Sep', 'Oct', 'Nov', 'Dec'
              )
            {
                    $contents .= "<option value=" . $_ . ">" . $_ . "\n";
            } ## end foreach ('Feb', 'Mar', 'Apr'...
            $contents .= "</select></td>";
            $contents .= "<td><select name=sYear>";
            $contents .= "<option value=2000>2000\n";
            $contents .= "<option value=2001 selected>2001\n";

            for ($i = 2002 ; $i <= 2010 ; $i++) {
                    $contents .= "<option value=" . $i . ">" . $i . "\n";
            }
            $contents .= "</select></td></tr>";

            $contents .= "<tr><td><b>End Date: </b></td><td>";
            $contents .= "<select name=eDay>";
            $contents .= "<option value=1 selected>1\n";
            for ($i = 2 ; $i <= 31 ; $i++) {
                    $contents .= "<option value=" . $i . ">" . $i . "\n";
            }
            $contents .= "</select></td>";
            $contents .= "<td><select name=eMonth>";
            $contents .= "<option value=Jan selected>Jan\n";

            foreach (
                    'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
```

```
                'Aug', 'Sep', 'Oct', 'Nov', 'Dec'
          )
        {
                $contents .= "<option value=" . $_ . ">" . $_ . "\n";
        } ## end foreach ('Feb', 'Mar', 'Apr'...
        $contents .= "</select></td>";
        $contents .= "<td><select name=eYear>";
        $contents .= "<option value=2000>2000\n";
        $contents .= "<option value=2001 selected>2001\n";

        for ($i = 2002 ; $i <= 2010 ; $i++) {
                $contents .= "<option value=" . $i . ">" . $i . "\n";
        }
        $contents .= "</select></td></tr>";
        $contents .= "</table></center>";
        $contents .=
          "<center><input type=submit value=\"View Stats\"></center><br>\n";
        $contents .= "</table>\n";
        print_page("Statistics", $contents);
} ## end sub request_stats

# Quick substitutions to make stats section more readable
```

---

reqtool/display_stats

NAME

display_stats -- Generate and display statistics

SYNOPSIS

```
display_stats ( $start, $end )
```

FUNCTION

Generate statistics for date range given and display as table

INPUTS

```
$start - start date
$end   - end date
```

OUTPUT

Calls print_page to display table of stats

SOURCE

```
sub display_stats {
        my ($start, $end) = @_;
```

```
            my (@line, $req, $request, $owner, %stats, $reqline, $state, $time);
            my ($start_time, $taken_time, $resolve_time, $contents, $fullname);

            # Define some values to increase readability
            my $state_mail = 1;
            my $state_resolve = 2;
            my $state_other = 3;

            # Open resolved index
            open(INDEX, %qconfig->{'RESOLVED_INDEX'})
              || print_error("Couldn't open resolved index so no stats could be
created");
            lock_read(\*INDEX);
            while (<INDEX>) {
                    $start_time = $taken_time = $resolve_time = 0;
                    @line = split (/\|/, $_, 3);

                    # If the request is outside requested dates ignore it
                    if (($line[2] < $start) || ($line[2] > $end)) {
                            next;
                    }

                    # Read the request and add the stats for it to global stats
                    $req = $line[0];
                    $req =~ s/R//g;
                    $owner   = $line[1];
                    $request =
                      %qconfig->{'RESOLVED_DIR'} . "/" . $req . "/request.contents";
                    open(REQ, $request) || print_error("Couldn't open request $req");
                    lock_read(\*REQ);

                    while (<REQ>) {
                            chomp $_;
                            @line = split (/: /, $_, 3);
                            if (/^Action: /) {
                                    if ($line[1] eq "Mail") {
                                            $state = $state_mail;
                                    } elsif ($line[1] eq "Resolve") {
                                            $state = $state_resolve;
                                    } else {
                                            $state = $state_other;
                                    }
                            } elsif (/^Date: /) {
                                    $time = str2time($line[1]);

                                    if (!$time) {
                                            $time = $line[1];
                                            $time =~ s/.*\((.*)\).*/$1/;
                                    }
                                    if (($state == $state_mail) && ($start_time == 0))
{
                                            $start_time = $time;
                                    } elsif ($state == $state_resolve) {
                                            $resolve_time = $time;

                                            if ($taken_time == 0) {
                                                    $taken_time = $time;
```

```
                                        }
                                } elsif ($taken_time == 0) {
                                        $taken_time = $time;
                                }
                        } ## end elsif (/^Date: /)
                } ## end while (<REQ>)
                close REQ;

                if (($taken_time > 0) && ($resolve_time > 0) && ($start_time > 0))
{
                        if (($start_time > $start) && ($start_time < $end)) {
                                %stats->{$owner}->{reaction} += ($taken_time -
$start_time);
                                %stats->{$owner}->{resolve}  += ($resolve_time -
$taken_time);
                                %stats->{$owner}->{count}    += 1;
                                %stats->{"ze Total"}->{reaction} += ($taken_time -
$start_time);
                                %stats->{"ze Total"}->{resolve}  +=
                                  ($resolve_time - $taken_time);
                                %stats->{"ze Total"}->{count} += 1;
                        } ## end if (($start_time > $start...
                } ## end if (($taken_time > 0) ...
        } ## end while (<INDEX>)
        close INDEX;

        $contents = "<table border=2 cellpadding=5>\n";
        $contents .=
          "<tr><th colspan=4 bgcolor=#ffffcc>Reqtool Statistics</th></tr>\n";
        $contents .= "<tr><th colspan=4 bgcolor=#ffffcc>"
          . localtime($start) . " to "
          . localtime($end)
          . "</th></tr>\n";
        $contents .=
    "<tr bgcolor=#99ffcc><td><b>Login</b></td><td><b>Count</b></td><td><b>Reaction
time</b></td><td><b>Resolve time</b></td></tr>\n";

        # Sort hash by login and display stats for each login
        foreach (sort keys(%stats)) {
                (undef, undef, undef, undef, undef, undef, $fullname) =
getpwnam($_);
                if ($_ eq "ze Total") {
                        $contents .= "<tr bgcolor=#ffcccc>";
                        $contents .= "<td bgcolor=#ffcccc><b>" . $_ . "</b></td>";
                } elsif ($fullname) {
                        $contents .= "<tr>";
                        $contents .= "<td bgcolor=#ccccff>" . $fullname . "</td>";
                } else {
                        $contents .= "<tr>";
                        $contents .= "<td bgcolor=#ccccff>" . $_ . "</td>";
                }
                $contents .= "<td>" . %stats->{$_}->{count} . "</td>";
                $contents .= "<td>"
                  . date_diff(0, int(%stats->{$_}->{reaction} / %stats->{$_}-
>{count}))
                  . "</td>";
                $contents .= "<td>"
```

```
                        . date_diff(0, int(%stats->{$_}->{resolve} / %stats->{$_}-
>{count}))
                        . "</td>";
                    $contents .= "</tr>\n";
            } ## end foreach (sort keys(%stats))
            $contents .= "</table>\n";

            print_page("Statistics", $contents);
    } ## end sub display_stats
```

---

reqtool/rebuild_indexes

NAME

rebuild_indexes -- Rebuild an index file from a directory

SYNOPSIS

rebuild_indexes ($dir, $indexfile, $status)

FUNCTION

Rebuild an index file from a directory

INPUTS

```
$dir       - directory to re-create from
$indexfile - index to re-create
$status    - status of requests (open/resolved)
```

OUTPUT

No output

SOURCE

```
    sub rebuild_indexes {
            my ($dir, $indexfile, $status) = @_;
            my ($req, $date, $subject, $owner, $from, @line, @reqs);

            # Open the directory to find all available requests
            opendir(DIR, $dir) || die ("Couldn't opendir " . $dir . ": $!");
            open(INDEXFILE, "+<" . $indexfile) || die ("couldn't open $indexfile");
            lock_write(\*INDEXFILE);

            # We only care about numbered requests
            @reqs = grep { /[0-9]/ } readdir(DIR);
            @reqs = sort { $a <=> $b } @reqs;
            # Parse each request and add to indexfile
            foreach (@reqs) {
```

```
                    $req = $_;
                    if (!open(REQ, $dir . "/" . $req . "/request.contents")) {
                            print STDERR
    "Couldn't open request.contents for $req\nLooks like directory damaged...
ignoring\n";
                    } else {
                            lock_read(\*REQ);
                            $owner = $from = $subject = $date = "";

                            while (<REQ>) {
                                    chomp;
                                    @line = split (/: /, $_, 2);
                                    if ((/^Date: /) && ($date eq "")) {
                                            $date = str2time($line[1]);
                                            if (!$date) {
                                                    $date = $line[1];
                                                    $date =~ s/.*\((.*)\).*/$1/;
                                            }
                                    } elsif (/^Owner: /) {
                                            $owner = $line[1];
                                    } elsif ((/^From: /) && ($from eq "")) {
                                            $from = $line[1];
                                    } elsif ((/^Subject: /) && ($subject eq "")) {
                                            $subject = $_;
                                            $subject =~ s/^Subject: //g;
                                    }
                            } ## end while (<REQ>)

                            if ($owner eq "") {
                                    $owner = "none";
                            }
                            if ($status eq "resolved") {
                                    $req .= "R";
                            }
                            print INDEXFILE $req . "|" . $owner . "|" . $date . "|" .
$status
                              . "|"
                              . $from . "|"
                              . $subject . "\n";
                            close REQ;
                    } ## end else
            } ## end foreach (@reqs)
            close INDEXFILE;
            close DIR;
    } ## end sub rebuild_indexes
```

---

reqtool/req_status

NAME

req_status -- Display reqtool status area

SYNOPSIS

req_status ($req, $status)

FUNCTION

Display what is the current request

INPUTS

$req   – current request number
$status – not used currently

OUTPUT

Calls print_page to display request number

SOURCE

```
 sub req_status {
        my ($req, $status) = @_;
        my ($contents);
        if ($req eq "") {
                $req = "Not selected";
        }
        $contents .=
          "<b>Current Request is <font color=blue>" . $req . "</font></b>\n";
        print_page("Status", $contents);
} ## end sub req_status
```

---

reqtool/print_usage

NAME

print_usage -- Print command-line usage

SYNOPSIS

print_usage

FUNCTION

Displays reqtool command-line usage to terminal

INPUTS

No output

OUTPUT

Usage to STDOUT

SOURCE

```
sub print_usage {
        print "Usage: reqtool [- | -r]\n";
        print "supported options:\n";
        print "   -     Read an email from STDIN and create a new open request\n";
        print "   -r    Rebuild active/resolved indexfiles\n";
        print "\n";
} ## end sub print_usage
```

---

reqtool/load_config

NAME

load_config -- Load configuration of given queue

SYNOPSIS

load_config

FUNCTION

Loads configuration of queue given in QUEUE into %qconfig

INPUTS

No output

OUTPUT

No output

SOURCE

```
sub load_config {

        # Set some initial defaults. Will be overridden by config values
        %qconfig->{'QUEUE_DIR'} =
          %qconfig->{'GLOBAL_DIR'} . "/" . %qconfig->{'QUEUE'};
        %qconfig->{'CGI_NAME'}       = "/cgi-bin/reqtool";
        %qconfig->{'LISTNAME'}       = "Reqtool Request";
        %qconfig->{'MAILING_LIST'}   = "reqtool-test\@bogus.com";
        %qconfig->{'TMP_DIR'}        = "/tmp";
        %qconfig->{'ADMIN_PASSWORD'} = "reqtool";
        %qconfig->{'READ_USER'}      = "sales";
        %qconfig->{'CHARSET'}        = "ISO-2022-JP";
        %qconfig->{'LDAP'}           = 0;
```

```
        %qconfig->{'LDAP_SERVER'}    = "";
        %qconfig->{'LDAP_BASE'}      = "";
        %qconfig->{'LDAP_BIND'}      = "";

        open(CONFIG, %qconfig->{'QUEUE_DIR'} . "/config")
          || die ("Could not open config file : $!");
        while (<CONFIG>) {
                my @line = split (/=/, $_, 2);
                $line[0] =~ /^\s*(.*?)\s*$/;
                $line[0] = $1;
                $line[1] =~ /^\s*(.*?)\s*$/;
                $line[1] = $1;
                %qconfig->{ $line[0] } = untaint_str($line[1]);
        } ## end while (<CONFIG>)
        close CONFIG;

        # Setting based of config values
        %qconfig->{'FORWARD_DIR'}  = %qconfig->{'QUEUE_DIR'} . "/forward";
        %qconfig->{'FORWARD_FILE'} = %qconfig->{'FORWARD_DIR'} . "/.forwards.html";
        %qconfig->{'MIME_TYPES'}   = %qconfig->{'GLOBAL_DIR'} . "/mime.types";
        %qconfig->{'COUNTFILE'}    = %qconfig->{'QUEUE_DIR'} . "/reqtool.count";
        %qconfig->{'ACTIVE_DIR'}   = %qconfig->{'QUEUE_DIR'} . "/active";
        %qconfig->{'ACTIVE_INDEX'} = %qconfig->{'ACTIVE_DIR'} . "/indexfile";
        %qconfig->{'IMAGE_PATH'}   = %qconfig->{'GLOBAL_DIR'} . "/images";
        %qconfig->{'VALID_USERS'}  = %qconfig->{'QUEUE_DIR'} . "/valid.users";
        %qconfig->{'RESOLVED_DIR'} = %qconfig->{'QUEUE_DIR'} . "/resolved";
        %qconfig->{'RESOLVED_INDEX'} = %qconfig->{'RESOLVED_DIR'} . "/indexfile";

        # Load LDAP module if required
        if (%qconfig->{'LDAP'}) {
                use Net::LDAP;
        }

        # rename program for ps listing
        $0 = "reqtool - " . %qconfig->{'LISTNAME'};

} ## end sub load_config
```

---

reqtool/un_base64

NAME

un_base64 -- Decode base64

SYNOPSIS

```
$text = un_base64 ( $str, $icode )
```

FUNCTION

Decode base64 for email headers when in other language

INPUTS

```
$str   – string to decode
$icode – unsure
```

OUTPUT

```
$text – decoded string
```

SOURCE

```
sub un_base64 {
        my ($str, $icode) = @_;
        my $len = length($str) * 3 / 4;
        $len -= length($1) if ($str =~ /(=+)$/);
        $icode =~ tr/A-Z/a-z/;
        $str   =~ tr#A-Za-z0-9+/=#`!-_`#;
        $str = chr($len + 32) . $str . "\n";
        $str = unpack("u", $str);
        my $alias = {
                "iso-2022-jp" => "jis",
                "euc-jp"      => "euc",
                "shift_jis"   => "sjis",
        };
        $icode = $alias->{$icode} if $alias->{$icode};
        return $str;
} ## end sub un_base64
```

---

reqtool/main_code

NAME

main_code – main code section, not in subroutine

SYNOPSIS

```
No output
```

FUNCTION

```
Figure out where to go
```

INPUTS

```
Commandline, cgi parameters
```

OUTPUT

```
   Everything
```

SOURCE

```
    my (%cookie, $req, $login, $image, $password);

    # set umask
    umask 0002;

    # Much selection goes on here
    # It will branch to the correct section based on command line and values
    # passed via the CGI interface

    # Check for command-line args and set queue dependant on argv[0]
    # Or grab queue from environment when running via CGI
    if ($ARGV[0]) {
            if ($ARGV[0] eq "-v") {
                    print "Reqtool Version: " . $VERSION . "\n";
                    exit;
            } else {
                    %qconfig->{'QUEUE'} = $ARGV[0];
                    %qconfig->{'QUEUE'} = untaint_str(%qconfig->{'QUEUE'});
            }
    } else {
            if ($ENV{PATH_INFO}) {
                    %qconfig->{'QUEUE'} = $ENV{PATH_INFO};
                    %qconfig->{'QUEUE'} =~ s/\///g;
                    %qconfig->{'QUEUE'} = untaint_str(%qconfig->{'QUEUE'});
            } else {
                    %qconfig->{'QUEUE'} = "default";
            }
    } ## end else

    load_config();

    # If there is a second argument then branch on it
    if ($ARGV[1]) {
            if ($ARGV[1] eq '-') {
                    read_email();
            } elsif ($ARGV[1] eq "-r") {
                    rebuild_indexes(
                            %qconfig->{'RESOLVED_DIR'}, %qconfig->{'RESOLVED_INDEX'},
                            "resolved"
                    );
                    rebuild_indexes(%qconfig->{'ACTIVE_DIR'}, %qconfig-
>{'ACTIVE_INDEX'},
                            "open");
            } else {
                    print_usage();
                    exit;
            }
    } elsif (param('command')) {  # Check if we are called to do specific thing

            # Fetch cookies
            %cookie = fetch CGI::Cookie;

            # Grab request number
```

```
        if ($cookie{'request'}) {
                $req = $cookie{'request'}->value;
        } else {
                $req = 0;
        }

        # Grab login name, or display login screen
        if ($cookie{'login'}) {
                $login = $cookie{'login'}->value;
        } else {
                if (param('command') eq "login") {
                        login(param('login'), param('password'));
                } else {
                        display_login();
                }
        } ## end else

        # Find fullname if we have LDAP
        my $fullname = "";
        if (%qconfig->{'LDAP'}) {
                my $LDAP = Net::LDAP->new(%qconfig->{'LDAP_SERVER'}) || die "$@";
                my $mesg = $LDAP->search(
                        base   => %qconfig->{'LDAP_BASE'},
                        attrs  => ['cn'],
                        filter => "(uid=" . $login . ")"
                );
                $fullname = $mesg->entry(0)->get_value("cn");
        } ## end if (%qconfig->{'LDAP'}...

        # Branch based on contents of command parameter
        if (param('command') eq "blank") {
                print_page("", "");
        } elsif (param('command') eq "buttons") {
                display_buttons();
        } elsif (param('command') eq "display_stats") {
                my $start =
                  param('sDay') . " " . param('sMonth') . " " . param('sYear');
                my $end = param('eDay') . " " . param('eMonth') . " " .
param('eYear');
                display_stats(str2time($start), str2time($end));
        } elsif (param('command') eq "download") {
                $req = param('req');

                if ($req =~ /R/) {
                        send_attachment(%qconfig->{'RESOLVED_DIR'} . "/" .
param('file'));
                } else {
                        send_attachment(%qconfig->{'ACTIVE_DIR'} . "/" .
param('file'));
                }
        } elsif (param('command') eq "getimage") {
                send_image(param('image'));
        } elsif (param('command') eq "list") {
                req_rescan($req);
        } elsif (param('command') eq "view") {
                $req = param('req');
                view_request($req);
```

```
} elsif (param('command') eq "req_comments") {
        req_comments();
} elsif (param('command') eq "req_comments_do") {
        req_comments_do($req, $login, param('comments'), $fullname);
} elsif (param('command') eq "req_drop") {
        req_resolve($req, "dropped", $fullname);
} elsif (param('command') eq "req_give") {
        req_give();
} elsif (param('command') eq "req_give_do") {

        if ($req) {
                req_give_do($req, param('username'), $login);
        } else {
                print_page("Error", "<h1>No Request Selected</h1>");
        }
} elsif (param('command') eq "req_mail") {

        if ($req) {
                req_mail($req);
        } else {
                req_mail(0);
        }
} elsif (param('command') eq "req_mail_do") {

        # If we have an attachment then call req_mail_do differently
        if (param('attach')) {
                req_mail_do(
                        $req,
                        $login,
                        param('address'),
                        param('cc'),
                        param('subject'),
                        param('mail'),
                        param('attach'),
                        tmpFileName(param('attach')),
                        uploadInfo(param('attach'))->{'Content-Type'},
                        uploadInfo(param('attach'))->{'Encoding'}
                );
        } else {
                req_mail_do(
                        $req,             $login,
                        param('address'), param('cc'),
                        param('subject'), param('mail'),
                        param('attach'),  "",
                        "",               ""
                );
        } ## end else

        if (param('resolve')) {

                # Wait for 20 sec to allow mail to be processed
                sleep 20;
                req_resolve($req, $login);
        } ## end if (param('resolve'))
} elsif (param('command') eq "req_merge") {
        req_merge($req);
} elsif (param('command') eq "req_merge_do") {
```

```
                req_merge_do(param('from_req'), param('to_req'));
        } elsif (param('command') eq "req_parameters") {
                req_parameters();
        } elsif (param('command') eq "req_rescan") {
                req_rescan($req);
        } elsif (param('command') eq "req_resolve") {

                if ($req) {
                        req_resolve($req, $login);
                } else {
                        print_page("Error", "<h1>No Request Selected</h1>");
                }
        } elsif (param('command') eq "req_search") {
                req_search();
        } elsif (param('command') eq "req_search_do") {
                req_search_do(
                        param('query'),  param('searchby'),
                        param('active'), param('resolved')
                );
        } elsif (param('command') eq "req_statistics") {
                request_stats();
        } elsif (param('command') eq "req_status") {
                req_status($req);
        } elsif (param('command') eq "req_take") {

                if ($req) {
                        take_request($req, $login);
                } else {
                        print_page("Error", "<h1>No Request Selected</h1>");
                }
        } elsif (param('command') eq "admin") {

                # Enter administration section
                if ($cookie{'admin'}) {
                        $password = $cookie{'admin'}->value;
                } else {
                        if (param('password')) {
                                $password = param('password');
                        } else {
                                req_parameters();
                        }
                } ## end else

                # Check that admin password is correct
                if ($password eq %qconfig->{'ADMIN_PASSWORD'}) {
                        if (param('admin')) {
                                if (param('admin') eq "admin_forward") {
                                        admin_forward();
                                } elsif (param('admin') eq "admin_forward_do") {
                                        admin_forward_do(param('address'),
param('mail'),
                                                param('current'));
                                } elsif (param('admin') eq "admin_users") {
                                        admin_users();
                                } elsif (param('admin') eq "admin_users_do") {
                                        admin_users_do(param('users'));
                                } elsif (param('admin') eq "admin_index") {
```

```
                                        rebuild_indexes(
                                                %qconfig->{'RESOLVED_DIR'},
                                                %qconfig->{'RESOLVED_INDEX'},
                                                "resolved"
                                        );
                                        rebuild_indexes(
                                                %qconfig->{'ACTIVE_DIR'}, %qconfig-
>{'ACTIVE_INDEX'},
                                                "open"
                                        );
                                        print_page("Updated Indexes", "Updated
Indexes");
                                } else {
                                        admin_main($password);
                                }
                        } else {
                                admin_main($password);
                        }
                } else {
                        req_parameters();
                }
        } ## end elsif (param('command') eq...
    } else {
        %cookie = fetch CGI::Cookie;

        if ($cookie{'login'}) {
                $login = $cookie{'login'}->value;
                display_frames();
        } else {
                display_login();
        }
    } ## end else
```

---

reqtool/QUEUE_DIR

NAME

QUEUE_DIR – Directory of queue

NOTES

GLOBAL_DIR/**QUEUE_DIR**

---

reqtool/CGI_NAME

NAME

CGI_NAME – Name of CGI script

NOTES

Given by config file

---

reqtool/LISTNAME

NAME

LISTNAME – Name of mailing list

NOTES

Given by config file

---

reqtool/MAILING_LIST

NAME

MAILING_LIST – address of mailing list

NOTES

Given by config file

---

reqtool/TMP_DIR

NAME

TMP_DIR – Temporary directory

NOTES

Given by config file

---

reqtool/ADMIN_PASSWORD

NAME

ADMIN_PASSWORD – Password to administration section

NOTES

Given by config file

---

reqtool/READ_USER

NAME

READ_USER – Username of read-only user

NOTES

Given by config file

---

reqtool/CHARSET

NAME

CHARSET – Character set

NOTES

Given by config file

---

reqtool/LDAP_SERVER

NAME

LDAP_SERVER – hostname of LDAP server

NOTES

Given by config file

---

reqtool/LDAP_BASE

NAME

LDAP_BASE – Base dn for bind

NOTES

Given by config file

---

reqtool/LDAP_BIND

NAME

LDAP_BIND – Full dn for bind

NOTES

Given by config file

---

reqtool/LDAP

NAME

LDAP – Boolean value say yay/nay to **LDAP**

NOTES

Given by config file

---

reqtool/FORWARD_DIR

NAME

FORWARD_DIR – Directory of forwards

NOTES

GLOBAL_DIR/forward

---

reqtool/FORWARD_FILE

NAME

FORWARD_FILE – Forward file

NOTES

FORWARD_DIR/.forwards.html

---

reqtool/MIME_TYPES

NAME

MIME_TYPES - List of mime types

NOTES

GLOBAL_DIR/mime.types

---

reqtool/COUNTFILE

NAME

COUNTFILE - Next request counter

NOTES

QUEUE_DIR/reqtool.count

---

reqtool/ACTIVE_DIR

NAME

ACTIVE_DIR - Directory of active requests

NOTES

QUEUE_DIR/active

---

reqtool/ACTIVE_INDEX

NAME

ACTIVE_INDEX - Index of active requests

NOTES

ACTIVE_DIR/indexfile

---

reqtool/IMAGE_PATH

NAME

IMAGE_PATH - Path to images for buttons

NOTES

GLOBAL_DIR/images

---

reqtool/VALID_USERS

NAME

VALID_USERS - List of valid users

NOTES

QUEUE_DIR/valid.users

---

reqtool/RESOLVED_DIR

NAME

RESOLVED_DIR - Directory of resolved requests

NOTES

QUEUE_DIR/resolved

---

reqtool/RESOLVED_INDEX

NAME

RESOLVED_INDEX - Index of resolved requests

NOTES

RESOLVED_DIR/indexfile

---

reqtool/pod

NAME

Pod documentation

DESCRIPTION

Documentation for reqtool that is displayed by perldoc

SOURCE

```
    =head1 NAME

    reqtool - Request Tracking Tool

    =head1 SYNOPSIS

    cat I<email> | B<reqtool> I<queue> -

    or reqtool [ I<queue> C<-r> | C<-v> ]

    or call directly as a CGI script

    =head1 DESCRIPTION

    When run from a terminal B<reqtool> takes standard in and processes it to be added
to the tracking system.
    It will mainly be used from a web browser where it provides an interface to work
with the individual requests.
    When called from the web the queue is set by calling it as B<reqtool/I<queue>>

    =head1 OPTIONS

    =over 4

    =item There is are three command line options

    =item B<->

    Take email from standard in
    Queue name needs to be set on commandline

    =item B<-r>

    Rebuild indexfiles from request directories
    Use this is your indexfiles get corrupted or you manually change the contents of
```

```
the active/resolved directories
    Queue name needs to be set on commandline

    =item B<-v>

    Get reqtool version

    =back

    =head1 CONFIGURATION

    All configuration is done by editing the config file in the directory of each queue

    =head1 REQUIRES

    Perl 5.6 or later, CGI, MIME::Parser, Sys::Hostname, Date::Parse, Net::LDAP is
optional

    =head1 AUTHOR

    Chris Debenham <Chris.Debenham@Sun.Com>

    =head1 COPYRIGHT

    This program is released under the GPL (http://www.gnu.org/copyleft/gpl.html)

    =head1 VERSION

    Reqtool version 0.11

    =head1 THANKS

    This was based on the great work by the authors of Reqng which was inturn based on
req but contains no code from either of those original programs.

    Thanks go to Remy Evard and other members of the systems group at Northeastern
University's College of Computer Science (http://www.ccs.neu.edu) for thier work on the
original req.

    Thanks also go to the authors of ReqNG (http://reqng.sycore.net/reqng) and WWWREQ
(http://www.cs.ucr.edu/~cvarner/wwwreq/) which the interface was based on.

    =cut
```