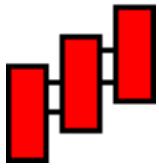


Steve Friedl's Unixwiz.net Tech Tips

Exporting MS Exchange 5.5 Users to Postfix



While configuring a **Postfix** mail server to relay inbound email from the outside world to a Microsoft Exchange server inside a customer's network, we wanted to populate the Postfix "relay_recipients" table with all the valid internal users. Though it's possible to just "relay everything", it means that the Postfix server will accept delivery for invalid accounts and then have them be refused by Exchange. This puts the burden of generating a bounce email onto Postfix. By populating the relay_recipients table, Postfix can reject this mail outright before taking delivery. It's just cleaner all around.



This Tech Tips documents how we built an automated system for exporting the user list from Microsoft Exchange, transferring it to the mail server, and specially processing the address list to build the proper table. This now runs on an automated basis on several customers and requires no human intervention.

The discussions here involve this environment:

- Microsoft Exchange 5.5 SP4
- PuTTY -or- Van Dyke Technologies "SecureCRT" 4.x
- Red Hat Linux 8.0
- Postfix 2.0.9
- OpenSSH 3.5

Note: We considered having Postfix make an LDAP query to the Exchange server, but we rejected it for several reasons. The main reason is that we wanted the mail relay machine to be as standalone as possible, not depending on Exchange to be available in realtime to decide to accept the message or not. We are looking into doing this "right" with LDAP, but for the time being we wanted the relay recipients listed locally.

Note further that here we are using Berkeley DB files for for storing the data even though there are plenty of other ways to do it (LDAP, MySQL, etc.). Adjust to your own environment.

Exporting users from Exchange 5.5

This proved to be the hardest part, and credit for figuring it out goes to Steve Gardiner of **Draper's & Damon's**. He waded through the bad and buggy Microsoft documentation to get it running on a completely unattended basis.

For this process we use the **ADMIN.EXE** (Exchange Administrator) command, but with command-line options that make it unattended. By way of example, our Exchange is installed at **D:\exchsrvr** and we're putting our custom files in **D:\userexport**. These of course can be relocated anywhere as needed.

One of the main difficulties was getting the *entire* list of email addresses in the system: all kinds of addresses were not showing up for one reason or another: this made the relay list incomplete.

We're creating several files in our **D:\userexport** directory:

■ **exportfields.txt**

The ADMIN program *reads* from the output file to learn what fields are being exported - which seems to us to be an odd arrangement - and we create this small template file to repopulate the file anew on each run. Otherwise, it's conceivable that a problem in the export process could lead to a trashed output file, losing the field list. Without a field list to start with, ADMIN chooses a default list that's not useful to us.

The file should contain:

```
Obj-Class      tab      E-mail addresses      tab      Secondary-
Proxy-Addresses
```

Once created, it's never touched again.

■ **userexport.ini**

This is a file that tailors the behavior of the ADMIN program for this export operation, and though it may be possible to use some "system" configuration file for this, we use a private config file that won't impact Exchange operations beyond the export.

```
[Export]
DirectoryService=servername here
Basepoint=
Container=Recipients
ExportObject=All
InformationLevel=Minimal
BasepointOnly=No
RawMode=No
HiddenObjects=Yes
Subcontainers=No
CodePage=0
; 09 = TAB
ColumnSeparator=09
; 37 = %
MVSeparator=37
; 34 = "
QuoteCharacter=34
```

■ **runexport.bat**

This batch file actually runs the export, and (later) will send the file to the Postfix mail server for addition to the relay-recipients database.

This batch file contains:

```
D:
cd \userexport
copy exportfields.txt exchusers.txt
\exchsrvr\bin\admin /e exchusers.txt /n /o userexport.
ini
```

The **/n** parameter suppresses a GUI progress display box, **/o** specifies the name of the options file, and **/e** shows where to export the data to.

Once these above files are created, give it a test run by launching the batch file. There won't be any meaningful output (remember that we used the `/n` switch to suppress progress reporting) only the final **exchusers.txt** file will be created to show success.

The file contains *all* email addresses for all users, and this includes addresses that aren't for the internet (X.500, CCMail, etc.). These are all removed later during file processing.

I Processing on the Exchange server

We are temporarily skipping the step of exactly how to get the data up to the Postfix system and just presume it somehow happened. This file has been conveyed to `/etc/postfix/exchusers.txt` and we'll touch on how we actually did that conveyance below.

The **exchusers.txt** file is in a form entirely unsuitable for use by Postfix, so we must do a bit of processing with a small perl program to make it useful. Though it's possible to do a direct one-to-one translation, in practice this is not very useful. The main reason is that most sites don't wish for *every* email address inside the network to be relayed from the outside.

In some cases each user has several addresses that account for previous email schemes, and in others there are users or distribution lists that should simply not be permitted from the outside: **everybody@unixwiz.net** would be a lousy address for a spammer to get. Finally, Exchange has some internal email addresses that don't look promising for external access, such as **schedule+freebusyinformation-ntserver@unixwiz.net**.

In addition, the directory can contain aliases for non-local addresses, such as "page-consultant" as an alias for an external pager email address. This is mainly for internal use: outside users shouldn't be able to use it. It's possible to exclude this specifically in the `--exclude` file, but it's easier to simply tell the parser to exclude *all* addresses that aren't in our interesting domain.

The `--domain=D` parameter adds **D** to the list of valid domains (it can be repeated), and if defined it ignores any addresses not in that list. If this option is not given at all, there are no domain-specific restrictions.

We normally put this rule in a makefile in the Postfix working area:

```

ALL = ...relay_recipients.db ...

all: $(ALL)

OPTS=--domain=unixwiz.net --exclude=exclude-users.txt

relay_recipients : exchusers.txt exclude-users.txt
tab    ./parse-exchange-users ${OPTS} < exchusers.txt > $@

%.db : %
tab    postmap $*

```

Now, typing "make" will build this file from scratch.

NOTE - those who have never used a makefile may wish to consult our other Tech Tip: [Using "make" for Postfix file maintenance](#).

Configuring Postfix to use the relay recipients is not really within the scope of this Tech Tip, but the relevant line in the **main.cf** file should be something like this:

```

relay_recipient_maps =
    hash:/etc/postfix/relay_recipients

```

In a more advanced environment, where one domain is on the "inside" but other domains are involved in relay, it may make sense to put the recipient lists in separate files:

```

relay_recipient_maps =
    hash:/etc/postfix/exchange_recipients
    hash:/etc/postfix/relay_recipients

```

Here, we presume that **exchange_recipients** is the dynamically built list, and **relay_recipients** is the one maintained by hand. We believe this does require two separate database queries, but we're not working in a high-volume environment. Those that are might concatenate two input files and create a single **relay_recipients** file as input to the database file.

As a final step we'll add a single command that's used to rebuild just the files

related to relay: it's used by the automated processes that follow. In the file `/etc/postfix/rebuild-relay-recipes` we include:

```
cd /etc/postfix
make relay_recipients.db
```

and the file must be made executable:

```
# chmod u+x /etc/postfix/rebuild-relay-recipes
```

Running this all by hand is very tedious, and in practice there is simply no way that anybody's going to be really religious about running this every time a user is added to Exchange.

So we've worked out a few ways to automatically copy the data from the Exchange machine to the Postfix machine using secure copies (we presume that Exchange is inside the corporate firewall and that Postfix may be outside or in the DMZ). Though we prefer the commercial software SecureCRT, we've also figure out how to use the freeware PuTTY tool for this.

Choose one of the two sections below.

■ Automating Using SecureCRT

We used **SecureCRT** from Van Dyke Software as our SSH client, and though it's commercial software (about \$100), we have used it for years and are very happy with it. It has a regular terminal emulation client, plus command-line copy and remote shell tools that work together.

After installing SecureCRT normally on the machine that runs Exchange, we next need to create an RSA public/private key pair to allow secure and unattended copies. This key should *not* be the one used for any other purpose!

Launch SecureCRT, then navigate this way through the menus:

1. Select **Tools:Create Public Key** from the top menu
2. Click **Next** after the introductory dialog box
3. Select a **RSA** key, then click **Next**

4. *do not select a passphrase*, then click **Next**
5. Select a 1024-bit key click **Next**
6. Move the mouse around as requested to provide random input, then click **Next**
7. Save the key file in **D:\userexport\exchupdate**, then click **Finish**
8. Click **No** when asked if you wish to use this as your global public key
9. Close SecureCRT

Somehow get the file **exchupdate.pub** to the Postfix server, and run these commands as root. One way is to ssh from the Exchange server to the Postfix server and actually paste the few ASCII lines from the pub file to the output place directly:

```
# cd /root/.ssh

# cat > exchupdate.pub
{paste exchupdate.pub here}
^D

# ssh-keygen -i -f exchupdate.pub >> authorized_keys2

# vi authorized_keys2
{add a comment "Exchange user update from NTSERVER"}
```

Now this key is allowed to run commands as root.

NOTE - there are all kinds of ways to add increased security to this arrangement, such as limiting which IP addresses this key can be used from, limiting which commands can run, and running this as a non-root user. This is all highly relevant, but we didn't want to bog down this Tech Tip with this detailed information. Feel free to give it a go.

Now we update our original batch file to reflect the added functions of "copy data to Postfix system" and "rebuild the relay recipients".

```
D:
cd \userexport
copy exportfields.txt exchusers.txt
\exchsrvr\bin\admin /e exchusers.txt /n /o userexport.ini
```

```
vcp -i exchupdate exchusers.txt root@servername:/etc/postfix  
vsh -i exchupdate -l root servername /etc/postfix/rebuild-  
relay-recipes
```

The last two lines do the real work, and it of course depends on having Van Dyke's **vcp** and **vsh** commands in the search path. Replace *servername* with the name of the *Postfix* server.

Now, running this script on the NT system will do a start-to-finish update of the relay recipients for this Exchange server, and this can be scheduled to run out of **WinAT** - the command scheduler - periodically. We typically run it once an hour during the workday. The command scheduler can be found in the Windows NT 4.0 Server Resource Kit.

■ Automating Using PuTTY

Though we have been fans (and paying customers) of SecureCRT for a very long time, we understand that others may wish for alternate solutions for getting the data from Exchange to the Postfix system. This section details the updates using the free solution PuTTY. Please note that this is the first time we've ever used PuTTY: those finding better ways to do this are encouraged to let us know.

1. Locate the three required PuTTY binaries: `pscp.exe` (secure copy), `plink.exe` (secure remote command execution), and `puttygen.exe` (the key generator). We normally put them right in the same directory with the other parts of this little system. We found PuTTY [here](#)
2. Create a PuTTY RSA public/private key pair
 - open command window, go to working directory (e.g., "**D:\userexport**")
 - run **puttygen.exe**
 - select the **SSH2 RSA Key** radio button
 - click the **Generate** button
 - move the mouse when requested to generate random data
 - when finished, enter anything you like for a key comment (we use "**Exchange User Update Key**")
 - do not enter a pass phrase!

- click "**Save Public Key**" and navigate to the directory you're working in: name it **exchupdate.pub**.
 - click "**Save Private Key**" and navigate to the directory you're working in: name it **exchupdate.ppk**. Approve the request to save without a passphrase.
 - exit the puttygen program
3. Somehow Convey the PuTTY public key file (**exchupdate.pub**) to the Postfix machine, put it in **/tmp** or other convenient place.
 4. As root, convert the key file from SSH2 format into OpenSSH format, appending it to the list of authorized keys:

```
# ssh-keygen -i -f /tmp/exchupdate.pub >> /root/.ssh/
authorized_keys2
# rm /tmp/exchupdate.pub
```

5. Edit the **/root/.ssh/authorized_keys2** file to make sure the key comment was entered - edit if necessary.
6. Update the **runexport.bat** batch file with the two secure commands:

```
D:
cd \userexport
copy exportfields.txt exchusers.txt
\exchsrvr\bin\admin /e exchusers.txt /n /o userexport.ini

pscp -2 -i exchupdate.ppk exchusers.txt root@servername:/etc/
postfix
plink -2 -i exchupdate.ppk root@servername /etc/postfix/rebuild-
exchusers
```

Download

- [parse-exchange-users.txt](#) - perl program (but named .txt for easy downloading)
- [userexport.ini](#) - sample options file for ADMIN.EXE
- [exportfields.txt](#) - export fields file

Navigate: [More Tech Tips](#)

[\[Home\]](#) ■ [Stephen J. Friedl](#) ■ Software Consultant ■ Tustin, California USA ■
steve@unixwiz.net