

FirstClass Scripts

Jason Healy, Director of Networks and Systems

Last Updated Mar 18, 2008

Contents

1	FirstClass Scripts	5
1.1	Introduction	5
1.2	Preflight Scripts	5
1.2.1	The Preflight Scripts	6
1.3	Postflight Scripts	7
1.3.1	The Postflight Scripts	7
1.4	FirstClass Network Store Backup	8
1.4.1	Background	8
1.4.2	Requirements	8
1.4.3	Design	8
1.4.4	Usage	9
1.4.5	Recovering from Zombie Processes	11
1.4.6	Restoration	12
1.5	Internet Services Log Rotate	13
1.5.1	Background	13
1.5.2	Requirements	13
1.5.3	Design	13
1.5.4	Usage	14
1.5.5	Viewing Old Logs	14

Chapter 1

FirstClass Scripts

Last updated 2008/03/18

1.1 Introduction

Suffield Academy uses **FirstClass 8.0** as it's e-mail server. We currently run it on Mac OS X, which has improved stability and reliability over First Class 7 under Mac OS 9.

First Class 8 (FC8) supports the ability to script certain actions via the command line (start, stop, restart, mirror pause, etc). Additionally, the standard control scripts support additional hooks for custom behaviors when the scripts are run.

Suffield uses both of these features to add custom behaviors to our FC8 server. We use the scripting to execute regular unattended backups of the First Class Network Store, and we use the script hooks for automatic maintenance tasks (such as log rotation).

1.2 Preflight Scripts

FC8 has hooks to execute preflight shell scripts whenever the user invokes the standard server control scripts `fcscctl` or `fcisctl`. These control scripts look in a specific location for a files called `prefcsd` and `prefcisd`, respectively. If they are found, the control scripts call them using the same arguments passed to the control script. When the preflight script returns, the control script resumes control and performs the operation requested by the user.

Because these preflight scripts execute entirely before the regular control script, they can be used to perform additional checks or setup for the FC8 daemons.

We have written a set of generic hook scripts, which can be used to call other scripts as needed. This additional layer of abstraction helps to find and diagnose problems, and also consolidates common code in external scripts.

1.2.1 The Preflight Scripts

The **preflight scripts** are called directly by the FC8 control scripts. They must have a specific name, and reside in a specific location, in order for FC8 to find and execute them.

All preflight scripts must live in the **Documents** folder in the home directory of the FC8 admin user (**fcadmin**). The scripts' names must start with "pre", and end with the name of the service that it ties in to.

Therefore, there are two preflight scripts: **prefcsd**, which goes with the FC8 Core Services daemon (the main server), and **prefcisd**, which goes with the FC8 Internet Services daemon (for SMTP, HTTP, IMAP, and other processing).

Our preflight scripts basically contain a large case statement to determine which scripts to run based on the action requested by the user. Because of this, both **prefcsd** and **prefcisd** scripts are nearly identical, except in name.

You may download the current version of both **prefcsd** and **prefcisd** from our [scripts directory](#).

You'll notice that the scripts look for one of the three supported actions: **start**, **stop**, and **restart**. Under each action, you may add any commands that you wish to execute when that action is requested. In most cases, the FC8 control script ensures that the requested daemon is running (or not running, depending on the action) before invoking the preflight script. Thus, you do not usually need to check on the status of the daemons before running your commands.

We have chosen to enclose all commands in external scripts (described below). For example, in the **start** stanza of the **prefcisd** script, we call our log rotation script. Every time the Internet Services daemon starts, the log rotation script executes. (For more information on this script, please see [the section on our log rotator script](#).)

Following our example, all you need to do is write a script that performs the functionality you need, and add it to the proper stanza in the preflight scripts.

1.3 Postflight Scripts

FC8 has hooks to execute postflight shell scripts whenever the user invokes the standard server control scripts `fcscctl` or `fciscctl`. First, the control scripts perform the operations requested by the user. After the normal operation has completed, they look in a specific location for a files called `postfcsd` and `postfcisd`, respectively. If they are found, the control scripts call them using the same arguments passed to the control script.

Because these postflight scripts execute entirely after the regular control script, they can be used to perform cleanup or notification on the FC8 daemons.

We have written a set of generic hook scripts, which can be used to call other scripts as needed. This additional layer of abstraction helps to find and diagnose problems, and also consolidates common code in external scripts.

1.3.1 The Postflight Scripts

The **postflight scripts** are called directly by the FC8 control scripts. They must have a specific name, and reside in a specific location, in order for FC8 to find and execute them.

All postflight scripts must live in the **Documents** folder in the home directory of the FC8 admin user (`fcadmin`). The scripts' names must start with "post", and end with the name of the service that it ties in to.

Therefore, there are two postflight scripts: `postfcsd`, which goes with the FC8 Core Services daemon (the main server), and `postfcisd`, which goes with the FC8 Internet Services daemon (for SMTP, HTTP, IMAP, and other processing).

Our postflight scripts basically contain a large case statement to determine which scripts to run based on the action requested by the user. Because of this, both `postfcsd` and `postfcisd` scripts are nearly identical, except in name.

You may download the current version of both `postfcsd` and `postfcisd` from our [scripts directory](#).

You'll notice that the scripts look for one of the two supported actions: **start** or **stop**. Under each action, you may add any commands that you wish to execute when that action is requested. In most cases, the FC8 control script ensures that the requested daemon is running (or not running, depending on the action) before invoking the postflight script. Thus, you do not usually need to check on the status of the daemons before running your commands.

At the moment, we do not make use of any postflight scripts (we use preflight scripts instead). However, to add postflight commands, all you need to do is write a script that performs the functionality you need, and add it to the proper

stanza in the postflight scripts.

1.4 FirstClass Network Store Backup

1.4.1 Background

FirstClass has a built-in mirroring capability that allows for a "warm" backup of the FirstClass system to exist at all times. We have chosen to mirror our FirstClass Network Store (FCNS) onto a second hard drive inside the server. The mirroring is quick, and does not suffer from external factors such as network availability.

That said, we do wish to guard against catastrophic failure of the entire server. We have written a script, `fcnsbackup`, which copies the entire FCNS off of the server and onto a backup server of our choosing. We use `rsync` for this task, so copies are quite fast (usually less than 15 minutes for a full synchronization).

In the sections below, we discuss the design of the script, its use, and how to recover using the backups it creates.

1.4.2 Requirements

This script synchronizes a **mirror** of the FCNS to a remote server. Therefore, the FirstClass server must have mirroring turned on (you cannot sync the main FCNS directly or data corruption may occur). Additionally, our script is written with the assumption that the mirror is locally attached to the machine running the FirstClass core server (if it isn't, then you probably don't need our script...)

You will need a remote host that can be reached over the network. This host must have `rsync` and an SSH server running on it (we tunnel the `rsync` connection over SSH for security). The remote host must have a directory that is writable by the username that we connect over SSH with.

The script is written for the **Bash** shell. At this writing, the script has been written and tested on machines running Mac OS X 10.3 (Panther). With the exception of file locations, the scripts should work for any Unix-like setup, and may even run under Windows with some minor modifications.

1.4.3 Design

If you'd like, you may view [the current version of `fcnsbackup`](#) from our [scripts directory](#) to see how it works.

Overall, what the script does is very simple:

1. Pause mirroring in FirstClass
2. Use rsync to transfer any changed files to the remote host
3. Resume mirroring in FirstClass

The script has some extra sanity-checking to prevent most basic problems. Briefly:

1. It creates a lockfile to prevent multiple copies of the script from running.
2. If interrupted or killed, the script attempts to "clean up" and restart mirroring in FirstClass.
3. If the synchronization process fails, diagnostic output is printed for analysis.

We use `rsync` to copy files to the remote host. `rsync` is great because it compares the files on both machines and only sends the bits that have changed. Due to the size of the FCNS, this comparison stage can take a while (10-20 minutes), but it is much faster than blindly copying all the files every time.

For security, we tunnel the `rsync` connection over `ssh`. To prevent the script from having to enter a password, we recommend setting up a password-less auth key (see the next section for more information).

The script is written such that user-customizable variables appear at the top, where they can easily be edited. The variables are commented, and should be familiar to anyone who has used `rsync` (if you are unfamiliar with `rsync`, we suggest running `man rsync` on the command line to view the manual for the program).

1.4.4 Usage

To use the script, copy it onto the server. We've chosen to place it in the **Documents** folder in the **fcadmin** account, as that's where FirstClass likes some of its other scripts to be kept.

SSH Keys

You'll need to set up a password-less SSH key to use for connecting to the remote server. You can do this using a command like this:

```
ssh-keygen -t dsa -N "" -C "Auto-Login Key" -f fcadmin_backup_autologin
```

You can change the part in quotes after `-C` to whatever description you want. Additionally, you can call the file whatever you want (in the example above, we call it `fcadmin_backup_autologin`).

When the key has been generated, there will be two files: one with the name you specified (the *secret* key), and one with `.pub` appended to the name (the *public* key). To use the keys, copy only the public key onto the remote host, and add it to the file `.ssh/authorized_keys` (create the file if it doesn't already exist). At this point, you should be able to log in to the remote host by typing:

```
ssh -i private_key_file user@remotehost
```

Where `private_key_file` is the name of your private key. For more information on using keyed auth with SSH, read the manual page for SSH or search the web for SSH key logins. Many extra options are available (for example, to restrict to the key to certain programs or IP addresses), but those topics are beyond the scope of this document.

Customization

Before running the script, you'll need to edit it and change the variables that appear at the beginning of the file. They are commented to describe their function, and are all self-explanatory. The `REMOTE_KEY` variable should be the path to the secret key file you created above.

Testing

At this point, you should be able to run the script from the command line to see if it works. Run the script, and make sure that you are not prompted for a password when you connect to the remote host.

You may wish to add `-nv` to the line that invokes `rsync` before you test. Doing so will print verbose information about what the script does, and additionally `//will not actually change the remote machine//`. Thus, you can see what would have happened, without worrying about messing something up (and without waiting for the data to transfer!).

When the script finishes, make sure that mirroring has turned back on in the FirstClass server.

When you're done, remove any debugging information from the script, and proceed to the next step.

Running Automatically

To have the script run automatically, you should add a line to `/etc/crontab` similar to the following:

```
7 0 * * * fcadmin /Users/fcadmin/Documents/fcnsbackup
```

If you're not familiar with `cron`, type `man crontab` on the command line for the manual page. The line above says to run the `fcnsbackup` script under the username `fcadmin`. The script is run every day at 12:07am (e.g., hour 0 and minute 7).

You should choose a time when your server is not under a heavy load, as the backup script requires a fair amount of processing power. Additionally, because the mirror is paused during backup, you want to try to backup during off-peak hours to reduce the amount of "catch-up" that the mirror must perform.

1.4.5 Recovering from Zombie Processes

The script takes steps to prevent a failed backup from causing problems down the line. However, it is possible that a badly timed disruption could prevent the script from running properly.

If you get a message like:

```
fcnsbackup: unable to get lock file from user ...
```

It means that another version of the script is still running, or that it didn't get a chance to clean up properly. To fix this problem, check to see if `rsync` is running on the machine:

```
ps auxwww | grep rsync
```

If no processes are listed, then the script was probably interrupted the last time it was run (perhaps due to a crash or power outage). You can remove the file `/tmp/lock.fcnsbackup` and the script should run once again.

If an active `rsync` process is running, you should investigate why. Are you running your backups too close together? Does it normally take this long? If you feel that the process has stalled, you can kill it using the normal `kill` semantics. The backup script should notice that it was interrupted, and exit cleanly. Confirm that the `/tmp/lock.fcnsbackup` file has been removed after you kill the process.

1.4.6 Restoration

Should the unthinkable happen, the instructions below should help you get the server up and running again.

First, you'll need to have a working server with the correct version of FirstClass Core Services installed on it. We keep a system image of the server in the **Network Administrators** share on the file server, along with directions for getting the server software reinstalled.

Once the OS and server are installed, you're ready to restore the FCNS. Because the backup script uses `rsync` to back up its files, we can just use `rsync` "in reverse" to copy the files back down.

To do this, you'll need the following:

1. The name of the remote machine you backed up to
2. The username on that machine that created the backup
3. The SSH key file or password for the account (if you don't have either, log in as an administrator and change the password to something you know).
4. The remote path where the files are backed up (e.g., `/Volumes/BigDisk/fcbackup/`)

With this information, you can restore the backup. On the FirstClass server, run a line similar to the following:

```
rsync -az --stats -e ssh "user@host:/path/to/fcns/" \  
"/Library/FirstClass Server/Volumes/Master/fcns/"
```

Note that you should replace `user`, `host`, and `/path/to/fcns` with the appropriate values. When you run the command, you will be prompted for the password to the remote account.

You may add `-v` to the command line (right after the `-az`) if you'd like more verbose output. This will show you each file as it is copied, but will slow down the transfer somewhat.

We have split the command over two lines, and used a backslash (`\`) to tell the terminal to split the command. You may type it all on one line, if you wish.

At this point, you will have to wait a while (as of this writing, a full restore for 10GB of data takes 1-2 hours). If you did not add `-v` to the command, you will not see any status of the file transfer, so you just need to be patient. Your best bet is to run `top` in another window and keep an eye on the load average and running processes.

When the copy is complete, your FCNS should be completely restored. At this time, you **must** run the following command on the FirstClass server:

```
"/Library/FirstClass Server/fcfixvol all all"
```

This repairs the permissions on the FCNS and ensures that it's ready to run. When this is complete, you're ready to start the server. Take a deep breath, and then go for it!

1.5 Internet Services Log Rotate

1.5.1 Background

Under Mac OS 9, the FirstClass Internet Services module displayed a running event log as it processed mail. Under OS X, this functionality is no longer turned on by default. We had grown used to the log information, so we turned it on when we switched to OS X.

Unfortunately, doing so created a log file on the machine that, if left unchecked, would grow without limit. While we had plenty of space remaining on the disks, we opted to write a small script to rotate this log file on a regular basis.

We called this script `fcisdlogrotate`, and we describe its design and functionality in the sections below.

If you wish, you may download the [the current version of `fcisdlogrotate`](#) from our [scripts directory](#).

1.5.2 Requirements

As written, the script requires that you run FC8 Internet Services with logging turned on. Additionally, you must have the `date` and `bzip2` utilities installed on your system (these come standard with Mac OS X 10.3, and most other Unix variants).

You may modify the script to use a different compression program (such as `gzip`) simply by modifying a global variable.

1.5.3 Design

The script simply looks for the current log file generated by Internet Services (`fcisd.log`). If found, it copies the log file to another directory, gives it a name

based on the current date and time, and compresses it to save space.

You may alter most of the script's behavior by modifying the global variables at the top of the script. You may specify which compression program to use, the initial location of the log file, and the destination directory to save the compressed files to.

1.5.4 Usage

To run the script automatically, copy it to the **Documents** folder of the **fcadmin** user, and add a line to the **start** stanza of the **prefcisd** script. See our [section on preflight scripts](#) for more information on the **prefcisd** script.

The script should run whenever you start or restart the daemon, and the logfile should rotate accordingly.

To rotate your logfiles regularly, just restart the Internet Services daemon on a regular basis. You can do this using the **cron** utility that comes built in to Unix.

Edit the file `/etc/crontab` and add the following line:

```
29 4 * * * fcadmin /usr/sbin/fcisctl restart
```

That tells the machine to restart **fcisd** at 4:29am every day. You may customize the time by modifying the fields at the beginning of the entry; see [man 5 crontab](#) for more information.

1.5.5 Viewing Old Logs

Old logs are stored in the destination directory specified in the script (`/var/log/fcisd/` by default). If your system has a copy of the **bzless** utility installed, you may use this to view the logs directly.

Otherwise, you should decompress the logs using the **bunzip2** utility, and then view the uncompressed logs with any text viewer.