Using the Radmind Command Line Tools to

Maintain Multiple Mac OS X Machines

Version 0.8.1

This document describes how to install, configure and use the radmind client and server tools to maintain a small lab of machines running Mac OS X. The methodology described can also be applied to using radmind on other platforms.

Client – Creating a Base Load

Prior to installing and using the radmind tools, you need to create a model machine that will be used to create the base load. A base load contains the operating system and other common files that every client will have. This might include the fully updated operating system, applications and any configurations to the system you want standardized. Once this machine is ready, you can install the radmind tools. It is a good idea to remove all removable media and unmount all network drives before performing the following steps.

1. Download and install the radmind tools from http://rsug.itd.umich.edu/software/radmind/

   The installer adds:
   - The radmind tools into /usr/local/bin/
     - fsdiff - compares file system to transcripts
     - lcreate - uploads a creatable transcript and its corresponding files to a radmind server
     - ktcheck - verifies and downloads a client's command file and transcripts
     - lapply - modifies a client's file system to match an appliable transcript
   - The radmind man pages into /usr/local/share/man
   - /var/radmind/client/command.K - an example command file used in this tutorial. (The command file is a list of transcripts that collectively describe a client's load set.)
   - /var/radmind/client/negative.T - a default negative transcript. (A transcript contains a list of file system objects e.g. directories, files, etc.)

2. Since the radmind tools are run from the command line, you will need to run them with the Terminal application.

3. The radmind tools need access to the entire file system, so they must be run as root. In the Terminal application, type:

   ```
   [client:~] user% sudo -s
   ```

   Enter your password when prompted. You must be an administrator of the machine to run this command. The –s option causes sudo to start a root shell, so at this point you have access to every file on your computer.

4. Using fsdiff with the –C option, you are going to make a createable transcript for the file system. A creatable transcript is one that can be used to create a load, in our case, the base load.

```
[client:~] root# fsdiff -C -c sha1 –o /var/radmind/client/base.T /
```

fsdiff reads the command file, /var/radmind/client/command.K by default, to get a list of transcripts and their individual type, either positive or negative. If a transcript is indicated as positive in the command file, fsdiff checks all attributes of each file system object (i.e. file, directory, link, etc) listed in that transcript. If a transcript is indicated as negative, fsdiff checks only some of the attributes depending on the type of file system object listed. For example, if a directory is listed in a negative transcript, fsdiff will check its UID, GID and permissions, but will not read the directory itself.

The radmind package installs an example command file with only one entry for the negative transcript negative.T, which is also installed by the package.

fsdiff then reads the entire file system starting at / (the "root" of the file system) and compares the local file system against the transcripts listed in the command file. fsdiff will write any difference between the transcripts and the local file system to the file, base.T.

The –c sha1 option tells fsdiff to calculate the sha1 checksum for all files. A file's checksums in conjunction with its size can be used to verify that the file has not been modified. Without checksums, fsdiff will only use a file's modification time and size to determine if a file's contents has changed.

5. Open base.T with your favorite UNIX editor (pico is used in this tutorial):

```
[client:~] root# pico /var/radmind/client/base.T
```

Taking a look at base.T, you will notice that it is rather large. That is because it contains an entry for just about every file system object on your machine. The only file system objects not listed are located in directories listed in negative.T. When fsdiff encounters a directory listed in a negative transcript, it will check the directory's attributes, but will not enter the directory itself. The contents of such directories are not managed by radmind.

6. Looking further at base.T you might notice that some files have been included in the transcript that one would not want to have in a base load. For instance, all of the files in /private/tmp are listed. Since these files are temporary, there is no need to include them in the base load.

To eliminate these files from base.T, you can add /private/tmp to the negative transcript and repeat the previous step. Adding /private/tmp/ to the negative transcript will cause fsdiff to check the directory's mode, uid and gid but the directory itself will not be read.

Use fsdiff with the –1 option to get the transcript line for /private/tmp.

```
[client:~] root# fsdiff -1 -c sha1 /private/tmp
d /private/tmp                          1777     0     0
```

Copy the transcript line to the clipboard.

7. Paste the transcript line into /var/radmind/client/negative.T.

   ```
   [client:~] root# pico /var/radmind/client/negative.T
   ```

   Transcripts are sorted in depth first order, which means subdirectories have precedence over files in the same directory. This means that "/" has special precedence. For example, /etc/passwd would be listed before before /etc.old even though "." comes before "/" alphabetically.

   With this in mind, insert the transcript line for /private/tmp into negative.T. When sorting transcripts, you should use the second argument of the transcript line for sorting. The first argument indicates the type of file system object and does not affect sort order.

   If you get the sorting wrong, fsdiff will give you an error with the offending line number.

8. Remake the creatable transcript for the system

   ```
   [client:~] root# fsdiff -C -c sha1 –o /var/radmind/client/base.T /
   ```

   Taking a look at this new base.T, you will notice that there are no listings for file system objects in /private/tmp.

   Every time you modify your negative transcript, you should recreate your base load. This way, any modifications to the negative transcript will be reflected in your base load.

Server – Installation and Configuration

The radmind client can connect to a radmind server running on any supported platform. These directions describe how to install the server on a Macintosh running Mac OS X. Mac OS X server is not required to run radmind.

1. Download and install the radmind tools from http://rsug.itd.umich.edu/software/radmind.

   To install the server tools you will need to perform a customized install. During the install process, click the "Customize" button, select the three server packages and press install.

   In addition to those items mentioned in the previous section, the installer adds:
   - lcksum - verifies a transcript's checksums and file sizes
   - /Library/StartupItems/RadmindServer - radmind server startup script
   - /usr/local/sbin/radmind - the radmind server

2. Since the radmind tools are run from the command line, you will need to run with the Terminal application.

3. The radmind server is installed as root, so to configure and start the server you must have root access. In the Terminal application, type:

   ```
   [server:~] user% sudo -s
   ```

   Enter your password when prompted. At this point you have access to every file on your computer, so be careful what you do.

4. The radmind server uses /var/radmind/config to determine which clients can connect to the server. Each line contains a client's DNS name or IP address and the client's command file separated by any amount of white space. Lines that are blank or begin with '#' are ignored. '*' is a wildcard and will match any string. A number range can be given by "<MIN-MAX>" where MIN is the lower bound and MAX is the upper bound. '\' can be used to escape any character.

   The following example defines four known clients, each using one of three different command files. Also, any client that ends with ".lab.umich.edu" will get lab.K as its config file and clients in the IP range 212.12.243.1 through 212.12.243.50 will get solaris8.K as their config file.

   ```
   #Client                 command file
   #
   amber.umich.edu         apple.K
   josh.umich.ede          apple.K
   ben.umich.edu           apple-test.K
   oreo.umich.edu          solaris8.K
   *.lab.umich.edu         lab.K
   212.12.243.<1-50>       solaris8.K
   ```

   Create the config file and add a line for your client. For now, use apple.K as its command file.

```
[server:~] root# pico /var/radmind/config
```

5. Create an empty command file.

```
[server:~] root# touch /var/radmind/command/apple.K
```

6. Start the radmind server:

```
[server:~] root# /usr/local/sbin/radmind
```

On future reboots, the RadmindServer startup script will start the server if /var/radmind/config exists.

Client – Storing the Base Load to the Radmind Server

With the radmind server running, you are now ready to store base.T and its associated files to the radmind server.

1. In the client's terminal window type:

   ```
   [client:~] root# lcreate -h your.radmind.server /var/radmind/client/base.T
   ```

   This stores base.T and all the files listed in the transcript on the radmind server.

2. You also must store the negative transcript on the radmind server.

   ```
   [client:~] root# lcreate –N –h your.radmind.server /var/radmind/client/negative.T
   ```

   This stores negative.T and all the files listed in the transcript on the radmind server. The –N option indicates that negative.T is a negative transcript and that all files should be stored as zero length.

Server - Verifying the Loads and Create the Command File

We are now ready to verify the loads and make them available from the radmind server. Files and transcripts stored on the server are initially saved in /var/radmind/tmp/file/ and /var/radmind/tmp/transcript/ respectively. This is a staging area where loads can be verified before being made available via radmind. Transcripts and files available from the radmind server are stored in /var/radmind/transcript and /var/radmind/file respectively.

1. In the servers terminal window type:

   ```
   [server:~] root# cd /var/radmind/tmp/transcript
   ```

   This places you in the temporary storage area.

2. Before you make the loads available with the radmind server, you need to check that the files were successfully stored on the server.

   ```
   [server:~] root# lcksum -c sha1 -n base.T
   ```

   lcksum reads the transcript base.T and compares the checksum and file size listed in the transcript with the actual checksum and size of the corresponding file on the server. With the –n option, lcksum will warn you if the transcript does not match the files stored on the server. Without the –n option, lcksum will update the transcript with the checksum and size of the file located on the server.

3. Since the negative transcript was stored using the –N option, all of the file sizes and checksums listed in the stored transcript are wrong. To fix it, type:

   ```
   [server:~] root# lcksum -c sha1 negative.T
   ```

   This will update negative.T with the correct checksums and sizes of the zero length files stored on the server.

4. Now that both the base load and negative overload have been verified, you are ready to make them available.

   ```
   [server:~] root# mv /var/radmind/tmp/transcript/base.T /var/radmind/transcript/
   [server:~] root# mv /var/radmind/tmp/transcript/negative.T /var/radmind/transcript/
   [server:~] root# mv /var/radmind/tmp/file/base.T /var/radmind/file/
   [server:~] root# mv /var/radmind/tmp/file/negative.T /var/radmind/file/
   ```

   This moves the transcripts and their associated files from the temporary directory where all loads are first stored into the working radmind directories.

5. Now you are ready to edit the command file to describe the load set consisting of the base load, base.T and the negative overload, negative.T.

   ```
   [server:~] root# pico /var/radmind/command/apple.K
   ```

6. Add these lines to the command file:

```
p base.T
n negative.T
```

The 'p' indicates that base.T is a positive transcript while the 'n' indicates that negative.T is a negative transcript.

Transcripts are listed in order of increasing precedence in a command file. The first transcript listed has the lowest precedence and the last the highest. If a file system object is listed in multiple transcripts, the transcript with the highest precedence is used.

For example, say you have a command file that looks like this:

```
p base.T
p overload.T
n negative.T
```

Where base.T has /Applications listed as:

```
d /Applications                         0775     0    80
```

And overload.T has /Application listed as:

```
d /Applications                         0755     0    0
```

The ordering of the command file tells us that overload.T has a higher precedence, so it's listing for /Applications will be used.

Client – Testing the Load Set

With both loads successfully stored, a command file to describe the load set, and a working config on the server, we are ready to test the load set.

1. In the client's terminal window type:

   ```
   [client:~] root# ktcheck –c sha1 -h your.radmind.server
   ```

   ktcheck checks the version of the command file and related transcripts. This tool connects to the radmind server and verifies that the client has the current version of its command file. The radmind server determines which command file a client should use from its config file. The client's command file is stored locally as /var/radmind/client/command.K.

   ktcheck then reads the client's command file to determine which transcripts the client needs and verifies with the radmind server that the current versions are present.

   If the client already has the current command file and related transcripts, ktcheck will not download any files. It will only transfer a few lines of text used for verification.

2. Using fsdiff with the –A option, you are going to create an appliable transcript

   ```
   [client:~] root# fsdiff -A -c sha1 /
   ```

   fsdiff reads the command file to get a list of the client's transcripts. fsdiff then reads the entire file system starting at / and compares the local file system against the transcripts listed in the command file. fsdiff will write any discrepancies between the transcripts and the local file system to the screen.

   This appliable transcript can be used by lapply to make the client match the load set as described by the command file and related transcripts. Transcript lines beginning with "+" indicate that a file must be download because it is missing or it's checksum or file size does not match the transcript listing; lines beginning with "-" indicate that the local file system object is not listed in a transcript and must be removed; while lines with no prefix indicate the files system object can be created or modified locally without download anything.

   Since this client was used to create the loads, there should not be many lines listed.

3. To apply this transcript to your client, type:

   ```
   [client:~] root# fsdiff -A -c sha1 / | lapply -h your.radmind.server
   ```

   Once lapply successfully finishes your client matches your transcripts exactly.

4. To show that there are no difference run fsdiff again

   ```
   [client:~] root# fsdiff -A -c sha1 /
   ```

Since you client is already up-to-date, fsdiff will not find any difference between the local file system and the load set.

Loading Other Clients

Once you have tested the load, you can easily load other clients with the same load set in a few steps.

1. Download and install the radmind tools on the new client.
2. Add the client to the server's config file using apple.K as its command file.
3. As root on the new client run ktcheck to get the command file and related transcripts:

   ```
   [client:~] root# ktcheck –c sha1 -h your.radmind.server
   ```

4. run fsdiff and lapply to update the machine.

   ```
   [client:~] root# fsdiff -A -c sha1 / | lapply -h your.radmind.server
   ```

Adding New Software - Creating an Overload

With a working base load and negative transcript, you are now ready to create an overload. An overload can be used to add software to a client without storing it in the base load. This allows a system administrator to have a single base load used on every client and specialized overloads containing software and/or configurations shared among a group of clients. For instance, all clients with scanners might have an additional overload containing the needed scanning software and configurations.

In this example, we create an overload for office, but the same steps can be followed for any software.

1. In order to capture the new software, you need to start with a clean client.

   ```
   [client:~] root# ktcheck –c sha1 -h your.radmind.server
   [client:~] root# fsdiff -A -c sha1 / | lapply -h your.radmind.server
   ```

   This will assure you that your client has the most recent load set.

2. Install office and make any configurations to the software you want to save in the overload.

3. Run fsdiff to make a creatable transcript for the overload.

   ```
   [client:~] root# fsdiff -C -c sha1 –o /var/radmind/client/office.T /
   ```

   fsdiff reads the command file to get a list of transcripts. fsdiff then reads the entire file system starting at / and compares the local file system against the transcripts listed in the command file. All Office files will be listed as additions to the local file system since they are not listed in any existing transcripts.

4. Review the new transcript.

   ```
   [client:~] root# pico /var/radmind/client/office.T
   ```

   While looking through the transcript you can see exactly what changes office made to your file system. You can check the permissions and remove any unwanted files from the overload.

   To remove an unwanted file, just delete its transcript line. Since lcreate stores files listed in the transcript, only files listed in the transcript will be included in the overload.

5. Store the overload to the radmind server

   ```
   [client:~] root# lcreate -h your.radmind.server /var/radmind/client/Office.T
   ```

Verifying the Overload and Adding It To A Command File

We are now ready to verify the overload and make it available from the radmind server.

1. In the servers terminal window type:

   ```
   [server:~] root# cd /var/radmind/tmp/transcript
   ```

2. Check that the files were successfully stored on the server.

   ```
   [server:~] root# lcksum -c sha1 -n Office.T
   ```

3. Once overload has been verified, you can check it in.

   ```
   [server:~] root# mv /var/radmind/tmp/transcript/office.T /var/radmind/transcript/
   [server:~] root# mv /var/radmind/tmp/file/office.T /var/radmind/file/
   ```

4. Now you are ready to create a new command file that describes the load set consisting of the base load, the negative overload and Office overload.

   ```
   [server:~] root# pico /var/radmind/command/graphics.K
   ```

   Add these lines to the command file:

   ```
   p base.T
   p Office.T
   n negative.T
   ```

   This indicates that any client using the graphics.K command file should have the base load, Office and negative overloads.

5. Edit the config file:

   ```
   [server:~] root# pico /var/radmind/config
   ```

   Change one of your client's command file listing in the config file from apple.K to graphics.K.

Test the Graphics Load Set

With the Office overload successfully stored, a command file to describe the load set and a modified config file on the server, we are ready to test the new load set. Be sure the client you are testing has graphics.K listed as its command file on the radmind server.

1. In the client's terminal window type:

   ```
   [client:~] root# ktcheck –c sha1 -h your.radmind.server
   ```

   ktcheck connects to the radmind server and verifies that the client has the current version of its command file. Since the command file has changed on the server, ktcheck downloads the new version to /var/radmind/client/command.K.

   ktcheck then reads the client's command file to determine which transcripts the client needs. If office.T is missing or is out-of-date, it will download the current version.

2. Use fsdiff and lapply to update the client:

   ```
   [client:~] root# fsdiff -A -c sha1 / | lapply -h your.radmind.server
   ```

   Once you have tested the load set, you can easily give other clients Office by changing their config file listing to have graphics.K as their command file.

   Other software can be added to the graphics load set in a similar fashion. Just create an over load for the software, store it to the radmind server and add it to the graphics command file.

Merging Overloads

In order to reduce the number of overloads listed in a clients command file, you can merge multiple overloads into one. For example, instead of having a base load with version 10.1 of the operating system and an overload for 10.1.1, 10.1.2, 10.1.3 and so on, you could merge the updates into the base load. The radmind server installation includes a tool that does this called lmerge.

In this example, we will merge two over loads that already exist on the server, Office, and Office-update, each of which are listed in graphics.K

1. On the server, move into the radmind transcript directory:

   ```
   [server:~] root# cd /var/radmind/transcript
   ```

2. Use lmerge to combine the two over loads.

   ```
   [server:~] root# lmerge office-update.T office.T office-new.T
   ```

   lmerge takes a list of transcripts in precedence order and created a new overload from them. In this case, the highest precedence transcript it office-update.T, the lowest precedence is office.T and the new overload will be call office-new.T

   lmerge writes the transcript line from each transcript to the new overload. If a file is listed in both transcripts, the transcript listing from office-update.T, the higher precedence transcript, will be written to the new overload.

   Files are hard linked from their original location into the new overload to minimize disk usage.

3. Modify the command file:

   ```
   [server:~] root# pico /var/radmind/command/graphics.K
   ```

   Remove office.T and office-update.T and replace them with office-new.T

4. Once you have tested this new overload, you can remove both office.T and office-update.T from the radmind server since they have been replaced with office-new.T.