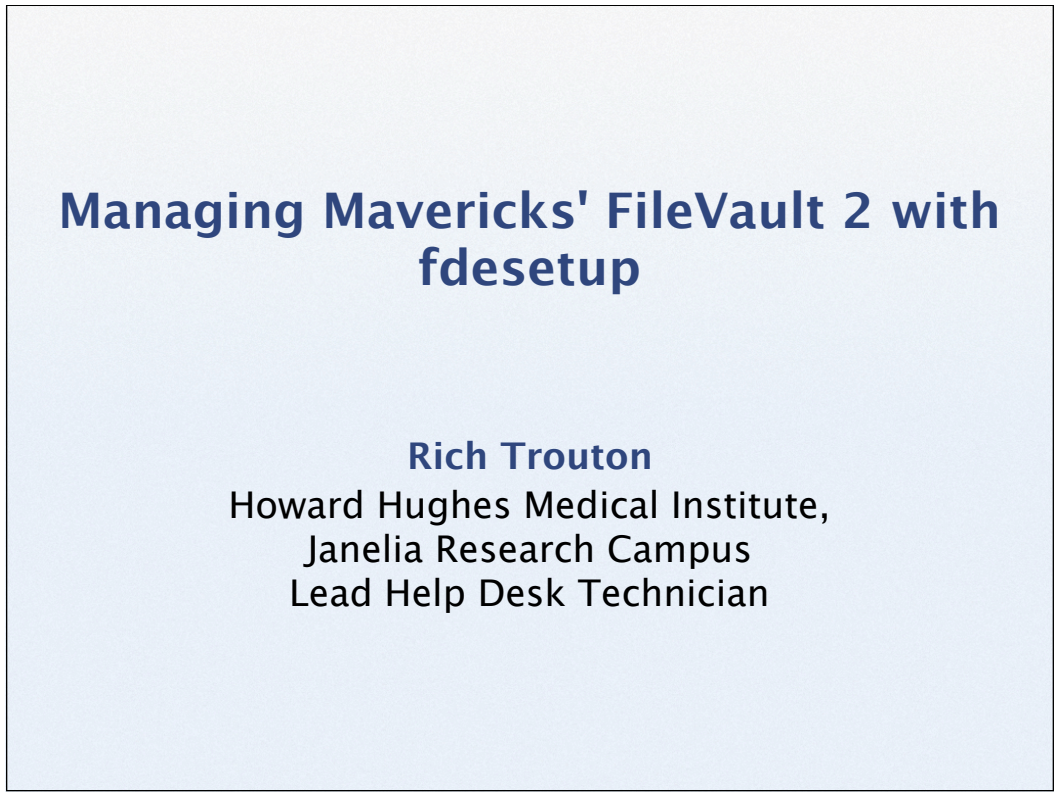


Before we get started, there's two things I'd like to mention. The first is that, all of the sides, speakers' notes and the demos are available for download and I'll be providing a link at the end of the talk. I tend to be one of those folks who can't keep up with the speaker and take notes at the same time, so for those folks in the same boat, no need to take notes. Everything I'm covering is going to be available for download.

The second is to please hold all questions until afterwards. If you've got questions, make a note of them and ask me at the end. With luck, I'll be able to answer most of your questions during the talk itself.

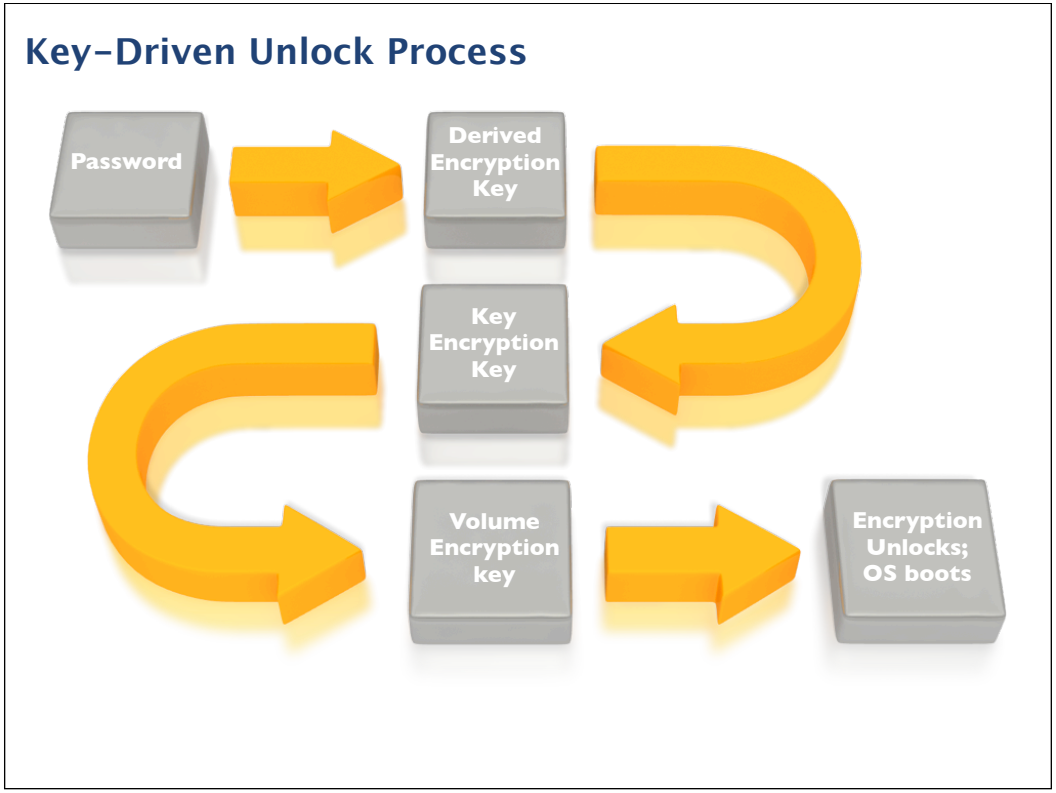
FileVault 2
Under The Hood

To better understand the capabilities of fdesetup and how FileVault management works on Mavericks, let's take a look underneath FileVault 2's hood to see how it handles authentication, unlocking and decryption.

Keys Used By FileVault 2

- Derived Encryption Key
- Key Encryption Key
- Volume Encryption Key

To begin with, passwords are almost irrelevant to FileVault 2's encryption. Instead, the system relies on a series of cryptographic keys granting access to two other layers of keys. These keys are the derived encryption key, the key encryption key and the volume encryption key.



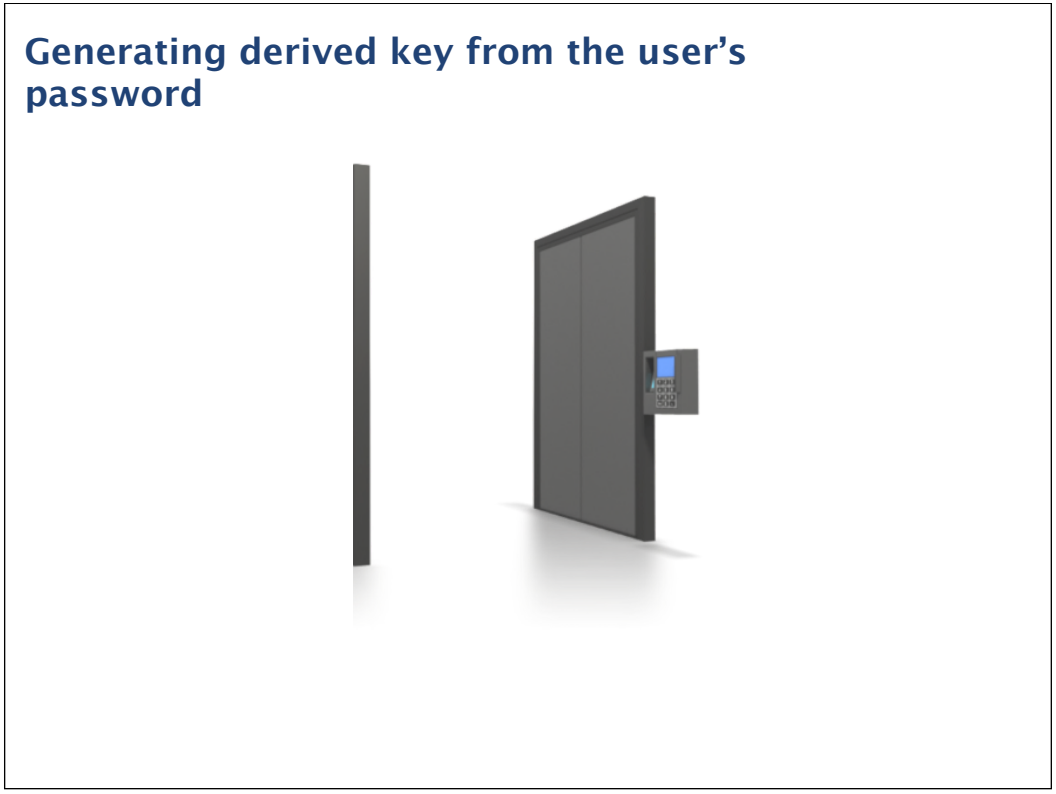
To give everyone an idea of how the keys are interacting with each other, here's a visual representation of what's happening when you log in at the pre-boot login screen.

To take them from the bottom layer up, let's first look at the Volume Encryption Key. This is the key that is interacting with the CoreStorage volume that the FileVault 2 encryption process has created. All cryptographic operations on an encrypted CoreStorage volume are unique to that volume because a different volume encryption key is randomly generated for each volume. This is the key that is actually unlocking the encrypted volume and it's also the key that's deleted when a wipe command is sent to a FileVault 2 encrypted Mac.

On the next level up, there's the key encryption key. This key is generated when FileVault 2 encryption is initialized on a particular volume. It is used to unlock the volume encryption key one layer down and acts as the middleman between the volume encryption key and the derived keys. This middle layer allows the derived keys to change without affecting the derived keys ability to unlock the encrypted volume. The key encryption key is the key that gets updated when derived keys are added, deleted or changed. With each change, addition or removal, the key encryption key gets re-wrapped to allow for the updated information.

On the top layer, there's the derived encryption keys. These keys begin the chain-reaction of unlocking the other keys below it, resulting in the unlocking or decryption of the encrypted volume. Any derived key can be independently changed without affecting its ability to unlock the other two layers of keys.

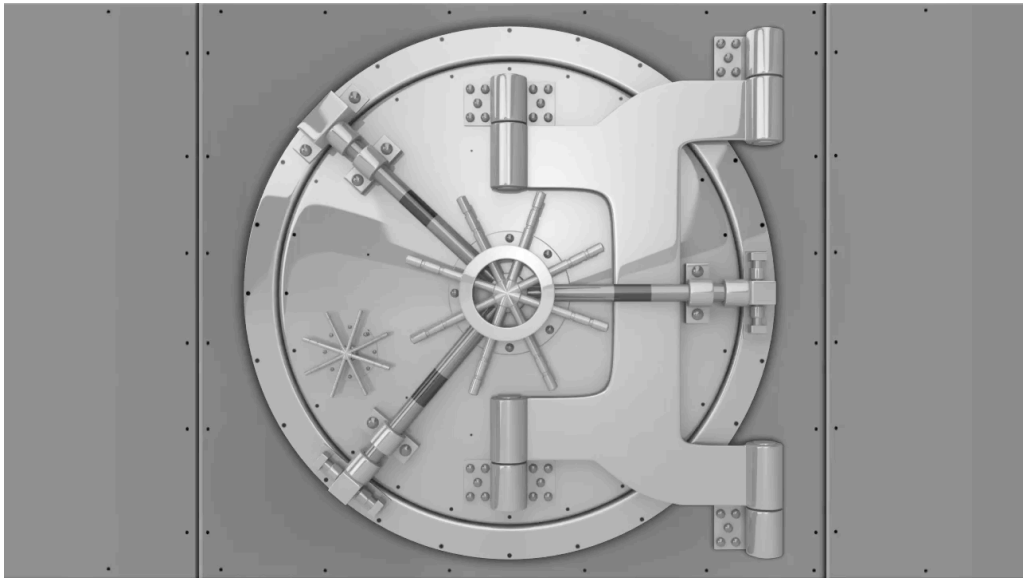
Any given CoreStorage volume must support multiple cryptographic users, each with their own derived key. This is important because it means that there can be multiple ways to access the encrypted volume. In the case of FileVault 2's encryption, it means that multiple user accounts can be enabled to unlock an encrypted Mac at the pre-boot login screen. Derived keys are also used for the FileVault 2 recovery keys.



To break it down further:

In this illustration, this process opens the door you're seeing above. You enter your password and the password is converted to a derived key with the RSA Password-Based Key Derivation Function (otherwise known as PBKDF2).

Key validation and volume unlock



The derived key unlocks the key encryption key, represented here by the vault door. Once the key encryption key has been unlocked, it grants access to the the volume encryption key.

The volume encryption key, represented here by the lock on the house, then unlocks the kernel and the OS boots.

Pre-encryption access

So how do you get a derived key in the first place? There's a couple of ways. The first is to have your access set up when FileVault 2 encryption is initialized. Since there are no pre-existing keys at this point, your password or other means of access get enabled at the same time that the encryption is initialized.

Post-encryption access

Once the encryption is turned on though, it's much tougher. The bouncer is standing by the door and he'll stop you from coming in unless you can properly show that you're legit. The only way to get in at this point is to have a friend who's already on the inside vouch for you.

Post-encryption access

An existing derived key can be used to enable new derived keys, so your friend who's already inside can use his derived key to enable your account. At that point, a new derived key is generated to give you access and you're good to go.

No key? No access.

Without an existing derived key available, there's no way to get a new derived key set up. To make a long story short, if you don't have a friend on the inside, you're not getting in.

Before we dive into fdesetup on Mavericks, let's take a look at what you can do with FileVault 2 when not using fdesetup.

You can monitor, unlock or decrypt a FileVault 2-encrypted boot drive using command line tools, but you can't start the encryption process from the command line using Apple's other native tools. Instead, the encryption needs to be enabled from System Preference's FileVault preference pane.

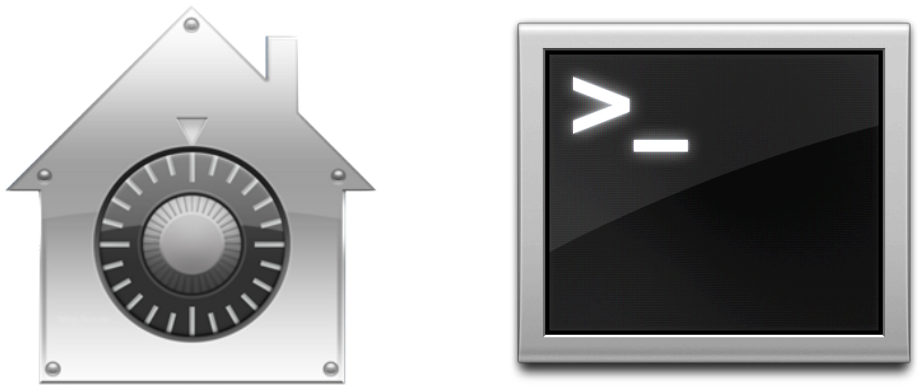
It is not possible to see who has FileVault 2-enabled accounts without looking at the pre-boot login screen.

It can be difficult to enable an account without using the FileVault preference pane.

It is not possible to remove an enabled account without deleting the account or setting the account password to be blank.

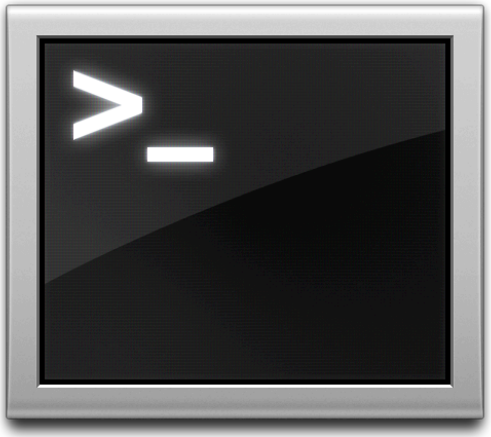
You have to choose between using the alphanumeric personal recovery key or using the institutional recovery key using FileVaultMaster.keychain.

fdesetup overview



fdesetup allows FileVault 2 administration from the command line and solves all of those problems with its various functions. It will turn on FileVault 2 encryption using a variety of options, disable encryption, allow addition and removal of FileVault 2 enabled users from the command line, supply a current list of authorized users, provide encryption status and much more.

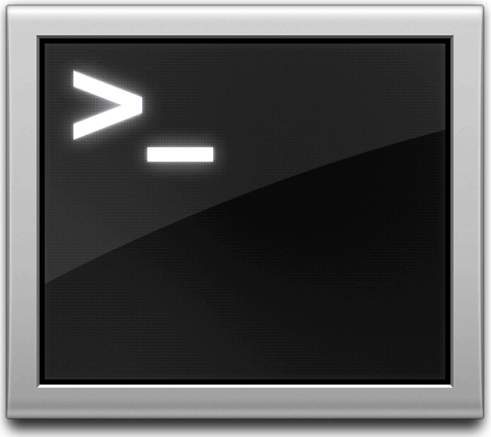
fdesetup commands



- fdesetup enable
- fdesetup disable
- fdesetup add
- fdesetup list
- fdesetup remove
- fdesetup changerecovery
- fdesetup removerecovery
- fdesetup sync
- fdesetup authrestart

fdesetup has a number of verbs associated with it. The ones that may be most commonly used in Mavericks are enable, disable, add, list, remove, changerecovery, removerecovery, sync and authrestart.

fdesetup enable

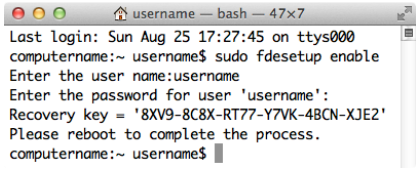


- Activates FileVault 2 Encryption
  - Can set FileVault 2 encryption to use:
    - Individual alphanumeric recovery key
    - Institutional recovery key using FileVaultMaster.keychain
    - Both kinds of recovery key simultaneously
  - Can enable multiple user accounts at time of encryption activation
  - Can import user and certificate information

fdesetup is amazingly flexible when it comes to enabling FileVault 2 encryption from the command-line.

fdesetup enable

sudo fdesetup enable

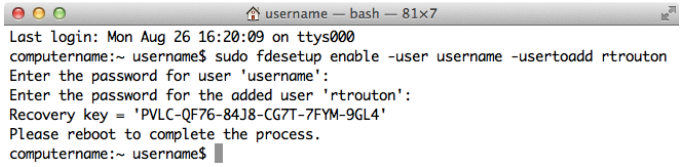


To start with the simplest method, you would run the command shown on the screen to enable FileVault 2 encryption. Next, you'll be prompted for the username and password of the user you want to authorize, which is the account you want to have appear at the FileVault 2 pre-boot login screen once the encryption is turned on. If everything's working properly, you'll next be given an alphanumeric personal recovery key and prompted to restart.

One thing that's very important to know is that the personal recovery key is not saved anywhere. You will need to make a record of it when it’s displayed or you will not have it later.

fdesetup enable -user

sudo fdesetup enable -user username -usertoadd username



You can also enable additional user accounts at the time of encryption, as long as the accounts are either local or mobile network accounts. You would run the command as shown on the screen and specify the accounts you want. As part of this, you will be prompted for the account passwords.

After that, you'll be given an alphanumeric personal recovery key and prompted to restart. All of the accounts specified should appear at the FileVault 2 pre-boot login screen.

fdesetup enable -inputplist

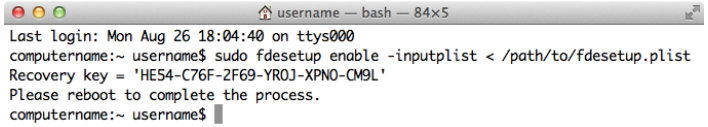
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Username</key>
<string>localadmin</string>
<key>Password</key>
<string>password</string>
<key>AdditionalUsers</key>
<array>
    <dict>
        <key>Username</key>
        <string>tom</string>
        <key>Password</key>
        <string>password</string>
    </dict>
    <dict>
        <key>Username</key>
        <string>harry</string>
        <key>Password</key>
        <string>password</string>
    </dict>
</array>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

For those who want to automate the process, fdesetup supports importing a property list file via standard input (stdin). The plist file needs to follow the format shown up on the screen and more users can be added as needed by appending new user information under the AdditionalUsers plist key.

fdesetup enable -inputplist

sudo fdesetup enable -inputplist < plistfile.plist



Once the plist has been set up, you would run the command shown on the screen to enable FileVault 2 encryption and reference the information in the plist file.

Since the accounts and passwords are in the plist file, fdesetup does not need to prompt for passwords. Instead, the alphanumeric personal recovery key is displayed and the user is prompted to restart. All of the accounts specified in the plist file should appear at the FileVault 2 pre-boot login screen.

fdesetup enable -defer

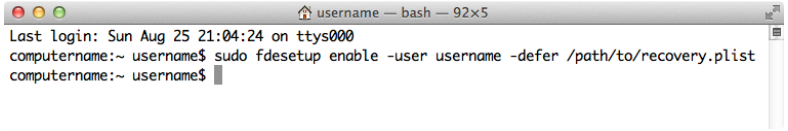
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>EnabledDate</key>
    <string>2012-07-08 08:05:54 -0400</string>
    <key>HardwareUUID</key>
    <string>00000000-0000-1000-8000-000C293DEFEC</string>
    <key>LVGUUID</key>
    <string>B7990442-91D1-4F5E-8F04-DDAC7B3610F2</string>
    <key>LVUUID</key>
    <string>FF837400-B781-496E-8992-D5A11B5A1139</string>
    <key>PVUUID</key>
    <string>E0FD9F00-5D41-4597-A10B-73F0A463CC65</string>
    <key>RecoveryKey</key>
    <string>8MGZ-C7BL-ML2M-J6B4-HN35-ZDOA</string>
    <key>SerialNumber</key>
    <string>VMWVk2fm2om0q0/em3zWsLr2g</string>
</dict>
</plist>
```

To avoid the need to enter a password, fdesetup also has a defer flag that can be used with the enable verb to delay enabling FileVault 2 until after the user logs out. With the defer flag, the user will be prompted for their password at their next logout. The recovery key information is not generated until the user password is obtained, so the defer option requires a file location where this information will be written to as a plist file.

The plist file will be created as a root-only readable file and contain information similar to what's shown on the screen. For security reasons, this plist file should not stay on the encrypted system. It should be copied to a safe location and then securely deleted from the system.

fdesetup enable -defer

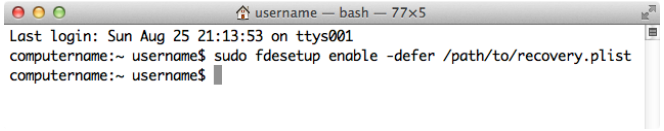
sudo fdesetup enable -user username -defer plistfile.plist



If you have a particular user account that you want to enable, you would run the command shown on the screen to defer enabling FileVault 2 and specify the account you want.

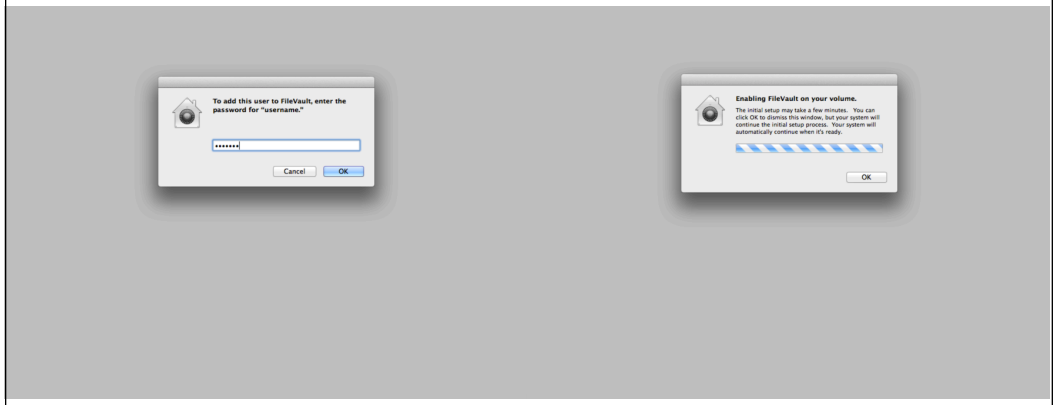
fdesetup enable -defer

sudo fdesetup enable -defer plistfile.plist



If you don't want to specify the account, you would use the command shown on the screen. If there is no account specified, then the current logged-in user will be enabled for FileVault 2. If there is no user specified and no users are logged in when the command is run, then the next user that logs in will be chosen and enabled.

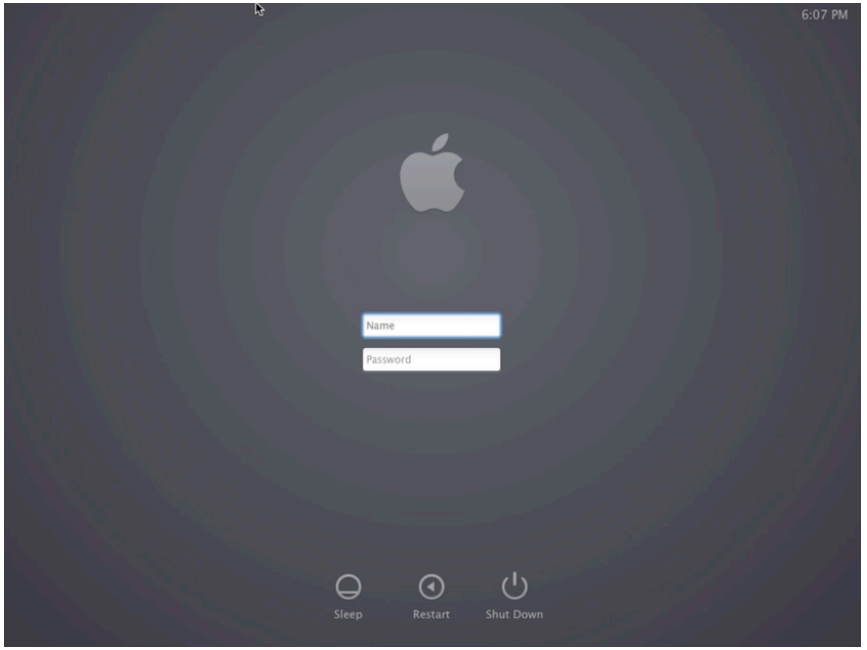
fdesetup enable -defer



On logout, the user will be prompted to enter their account password. Once entered, FileVault 2 will be enabled and the recovery information plist file will be created. Once the enabling process is complete, the Mac will restart.

An important thing to keep in mind about the defer option is that it enables one single user account at the time of turning on FileVault 2 encryption. The defer option does not enable multiple user accounts and cannot be used to enable accounts once FileVault 2 encryption has been turned on.

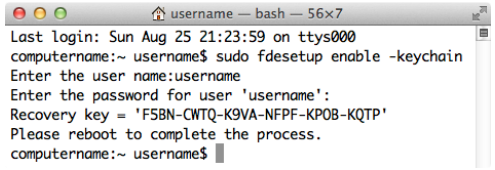
fdesetup enable -defer



Since this can be something that's better shown than explained, lets take a look at how the defer process works.

fdesetup enable -keychain

sudo fdesetup enable -keychain

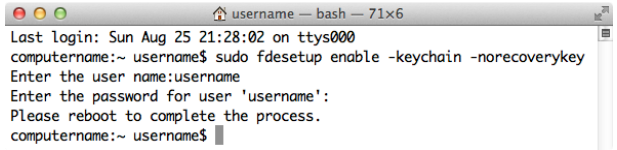


Another capability of FileVault 2 in Mavericks is the ability to use the alphanumeric personal recovery key, an institutional recovery key using FileVaultMaster.keychain, or both kinds of recovery key at the same time.

As seen in the earlier examples, fdesetup will provide the alphanumeric personal recovery key by default. To use the institutional recovery key, the -keychain flag needs to be used as shown on the screen. The alphanumeric personal recovery key is displayed, but the encryption will also use the FileVaultMaster.keychain institutional recovery key. In case recovery is needed, either recovery key will work to unlock or decrypt the encrypted drive.

fdesetup enable -keychain

sudo fdesetup enable -keychain -norecoverykey



If you want to specify that only the FileVaultMaster keychain be used, both the -keychain and -norecoverykey flags need to be used when enabling encryption

fdesetup enable -certificate

sudo fdesetup enable -certificate cert.cer

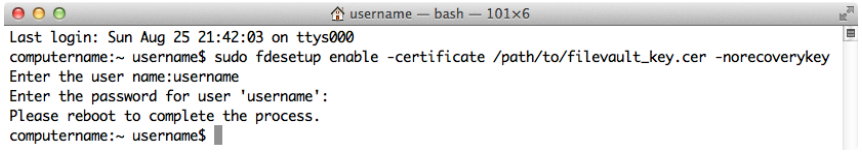
fdesetup is also capable of creating a FileVaultMaster keychain and automatically storing it in /Library/Keychains. To do this, an existing FileVault 2 public key needs to be available as a DER encoded certificate file. Once that's available, the command shown on the screen will enable FileVault 2, automatically create the institutional recovery key with the supplied certificate file and store it as /Library/Keychains/FileVaultMaster.keychain

fdesetup enable -certificate

Let's take a look at how you would create a DER encoded certificate file from an existing public key. In this case, we're assuming that there is not a pre-existing recovery key so we'll be creating one with the create FileVaultMaster keychain tool.

fdesetup enable -certificate

sudo fdesetup enable -certificate cert.cer -norecoverykey



To specify that only the FileVaultMaster keychain be used as the recovery key, you would add the norecoverykey flag to the command.

fdesetup enable -inputplist with public key data

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Username</key>
<string>username</string>
<key>Password</key>
<string>pass123</string>
<key>AdditionalUsers</key>
<array>
    <dict>
        <key>Username</key>
        <string>tom</string>
        <key>Password</key>
        <string>password</string>
    </dict>
    <dict>
        <key>Username</key>
        <string>harry</string>
        <key>Password</key>
        <string>password</string>
    </dict>
</array>
<key>Certificate</key>
<data>
MIIDLjCCAhagAwIBAgIENsqyWTALBgkqhkiG9w0BAQUwSTEfMB0GA1UEAwwWRmlsZVZhdWx0IFJlY292ZXJ5IEtleTEmMCQGA1UE
DQwddHJvdXRvbnQtdm0zLnRhb21lY2h3b3Jrcy5uZXQwHhcNMTMwODI0MjEzNTI1WhcNMTQwODI0MjEzNTI1WjBJMR8wHQYDVQQD
DBZGaWklVmF1bHQgUmVjb3ZlcnkgS2V5MSYwJAYDVQQNDB10cm91dG9udC12bTMudGFvbWVjaHdvcmtzLm5ldDCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAMJp04TYyH8HC3H4WWV0FPV48I1cdRjhBGi9OYJX8cQrgWb72FTDCNbU+fD1ce1++
KALheMNCmhOIwPshQtbUuU9ngpfRYI/L6KIgIaUW4bpUksHXZ0CIwSvfqD6+
np06wniS48KNPmnyHqxNeYyfU20wAITPznP7w0VjU7pMbvvJra03gNDxxLc44Rj5YdHxn7U7gIUNc08RxntNgC6hPSMuZVIgz8AU
FcOYuBYML9fXrHytNoaZ0JXS2djGRdwRT0V3OtK2qDFh8+vVr9vTBt3xmk6hPREXfB9odRDH6ZP9mn6R2vLLzF4ANwklha0L4+
mwbPtdfVOSs3wdkoUFF8CAwEAAaMgMB4wCwYDVR0PBAQDAgK0MA8GA1UdEwEB/wQFMAMBAQAwDQYJKoZIhvcNAQEFBQADggEBAJ+
5ZaqzQmnVDrIQBkxiBXql/
aRbQpetu5jLsJtHDxRd1d2YiEFJHwy4jnUefrvTUf764BoCQ5FtOjYL5lYn7QQ3MjvsFKzBh2wVlqJJAhjjjwbzMoVdRxfY8Aftc
ler9Tw/WvxiECvdgxBA37iA+EGVTrRUGuCytY+GhawIKQMR5Ac2XBpfgPatbV9xAtCo4+M9I4jqhSxOGMwLjRHs4yELyH71w/
Au3VlTYezf+dqofOaZ8c4wBOK3T4VeoWMJjjwDe9Q3/j3sf/CjxvNhm1yYRUqUieZx7Vdf2+
oKKqMczyAVz0Y8UX26lMnpMQn4Fv9b5m0vo5U2/ZgpiQ8cHxM=
</data>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

It is also possible to include the public key data in a plist file, which allows the use of a plist to set up the institutional recovery key. The plist needs to follow the format shown on the screen.

fdesetup enable -inputplist with public key data

base64 /path/to/filename.cer > /path/to/filename.txt

Using the public key's DER encoded certificate file, the public key data for the plist can be obtained using the base64 tool by using the command shown on the screen to export the data to a text file.

fdesetup enable -inputplist with public key data

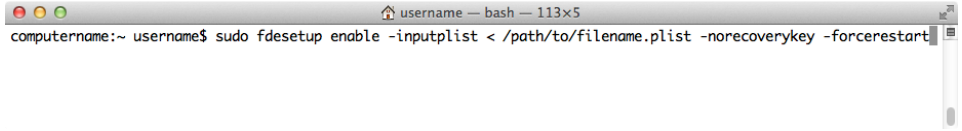
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Username</key>
<string>username</string>
<key>Password</key>
<string>pass123</string>
<key>AdditionalUsers</key>
<array>
    <dict>
        <key>Username</key>
        <string>tom</string>
        <key>Password</key>
        <string>password</string>
    </dict>
    <dict>
        <key>Username</key>
        <string>harry</string>
        <key>Password</key>
        <string>password</string>
    </dict>
</array>
<key>Certificate</key>
<data>
MIIDLjCCAhagAwIBAgIENsqyWTALBgkqhkiG9w0BAQUwSTEfMB0GA1UEAwwWRmlsZVZhdWx0IFJlY292ZXJ5IEtleTEmMCQGA1UE
DQwddHJvdXRvbnQtdm0zLnRhb21lY2h3b3Jrcy5uZXQwHhcNMTMwODI0MjEzNTI1WhcNMTQwODI0MjEzNTI1WjBJMR8wHQYDVQQD
DBZGaWklVmF1bHQgUmVjb3ZlcnkgS2V5MSYwJAYDVQQNDB10cm91dG9udC12bTMudGFvbWVjaHdvcmtzLm5ldDCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAMJp04TYyH8HC3H4WWV0FPV48I1cdRjhBGi9OYJX8cQrgWb72FTDCNbU+fD1ce1++
KALheMNCmhOIwPshQtbUuU9ngpfRYI/L6KIgIaUW4bpUksHXZ0CIwSvfqD6+
np06wniS48KNPmnyHqxNeYyfU20wAITPznP7w0VjU7pMbvvJra03gNDxxLc44Rj5YdHxn7U7gIUNc08RxntNgC6hPSMuZVIgz8AU
FcOYuBYML9fXrHytNoaZ0JXS2djGRdwRT0V3OtK2qDFh8+vVr9vTBt3xmk6hPREXfB9odRDH6ZP9mn6R2vLLzF4ANwklha0L4+
mwbPtdfVOSs3wdkoUFF8CAwEAAaMgMB4wCwYDVR0PBAQDAgK0MA8GA1UdEwEB/wQFMAMBAQAwDQYJKoZIhvcNAQEFBQADggEBAJ+
5ZaqzQmnVDrIQBkxiBXql/
aRbQpetu5jLsJtHDxRd1d2YiEFJHwy4jnUefrvTUf764BoCQ5FtOjYL5lYn7QQ3MjvsFKzBh2wVlqJJAhjjjwbzMoVdRxfY8Aftc
ler9Tw/WvxiECvdgxBA37iA+EGVTrRUGuCytY+GhawIKQMR5Ac2XBpfgPatbV9xAtCo4+M9I4jqhSxOGMwLjRHs4yELyH71w/
Au3VlTYezf+dqofOaZ8c4wBOK3T4VeoWMJjjwDe9Q3/j3sf/CjxvNhm1yYRUqUieZx7Vdf2+
oKKqMczyAVz0Y8UX26lMnpMQn4Fv9b5m0vo5U2/ZgpiQ8cHxM=
</data>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

You would then add the exported data from the text file into the Certificate key of the plist.

fdesetup enable -forcerestart

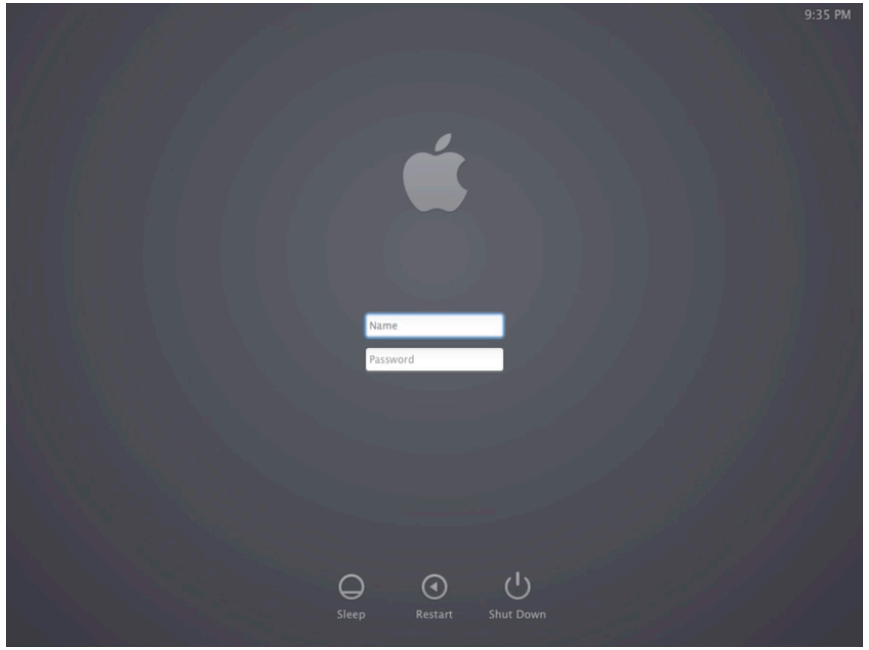
sudo fdesetup enable -inputplist < /path/to/filename.plist -norecoverykey -forcerestart



Along with the various options for enabling, it's also possible to force a restart of the Mac once FileVault 2 has been successfully configured. This can help automate the process of enabling FileVault 2 on a Mac if no input from a logged-in user is needed.

For example, an organization may want to pre-configure its Macs to automatically encrypt with FileVault 2 at first boot with a local admin account enabled. It also wants to use only the institutional recovery key. If a plist with the desired account information and public key data to create the institutional recovery key is available, the command shown on the screen could be run to enable FileVault 2 and force a restart at the first boot.

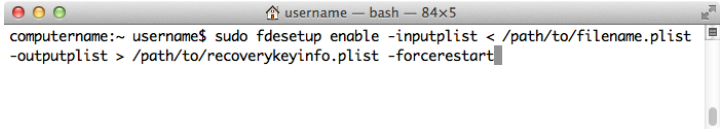
fdesetup enable -forcerestart



Since this combines three different enable options, let's take a look at how it works when you run that command to automatically encrypt. In this case, I'm going to be enabling three accounts via a plist file and setting the institutional key as the sole recovery key.

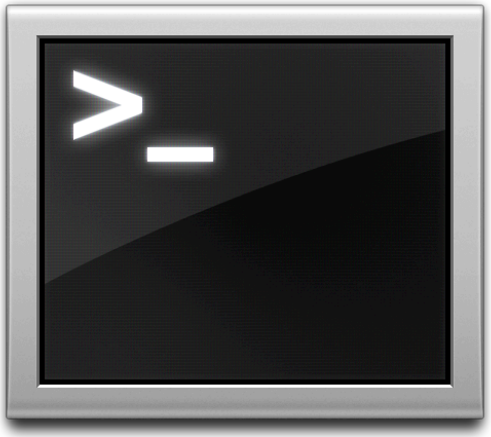
fdesetup enable –forcerestart

sudo fdesetup enable -inputplist < /path/to/filename.plist -outputplist > /path/to/recoverykeyinfo.plist –forcerestart



If you want to use the alphanumeric personal recovery key with –forcerestart, the personal recovery key can be exported into a plist file. Taking the example shown on the screen, the institution's automated setup would run the following command to automatically encrypt with FileVault 2 at first boot using both types of recovery key and a local admin account enabled.

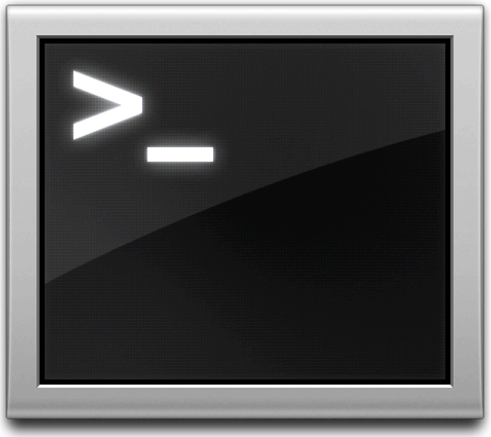
fdesetup disable



› Disables FileVault 2 encryption

In contrast to all of the various options available for enabling FileVault 2 using fdesetup, the command to turn off FileVault 2 encryption is fdesetup disable. There are no additional flags associated with this command.

fdesetup add

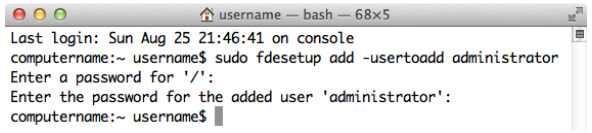


- Enables additional accounts after FileVault 2 encryption is complete
  - Can enable multiple user accounts
  - Can import user information

Once the Mac has been fully encrypted with FileVault 2, you can add additional users using fdesetup. To do so, you will need to provide the password of a previously enabled account as well as the password of the account you want to add.

fdesetup add -usertoadd

sudo fdesetup add -usertoadd username



The command shown on the screen will enable a specified user on this encrypted Mac. The initial password prompt can be the password of any account on the Mac that's already been enabled for use with FileVault 2.

fdesetup add -inputplist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Username</key>
<string>rtrouton</string>
<key>Password</key>
<string>password</string>
<key>AdditionalUsers</key>
<array>
    <dict>
        <key>Username</key>
        <string>fcheeryble</string>
        <key>Password</key>
        <string>password</string>
    </dict>
    <dict>
        <key>Username</key>
        <string>nnickleby</string>
        <key>Password</key>
        <string>password</string>
    </dict>
</array>
</dict>
</plist>
```

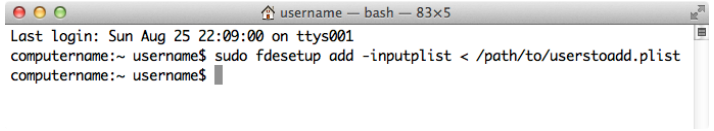
Note: All account passwords need to be supplied in cleartext.

For those who want to automate the process, fdesetup also supports importing a plist file via standard input (stdin). The plist needs to follow the format shown up on the screen.

When adding additional users using a plist file, the top level Username key is ignored, and the Password key value should either be an existing FileVault user's password or the personal recovery key. Additional users can be added as needed by adding additional user information under the AdditionalUsers plist key.

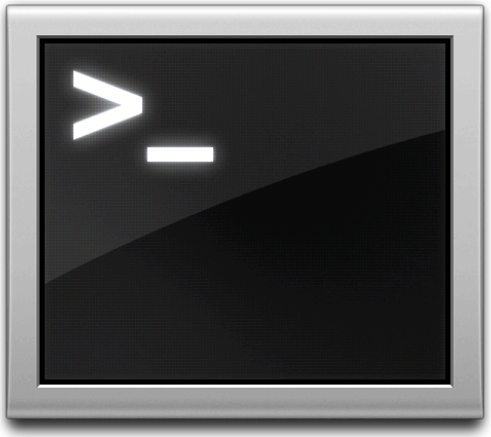
fdesetup add -inputplist

sudo fdesetup add -inputplist /path/to/plistname.plist



Once the plist has been set up, you can run the command shown on the screen to add additional users by referencing the account information in the plist file.

fdesetup list

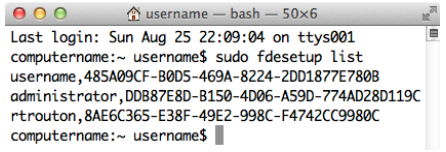


- Displays enabled accounts
  - List includes the accounts' usernames and UUIDs

To list all accounts enabled for FileVault 2, fdesetup includes the list verb.

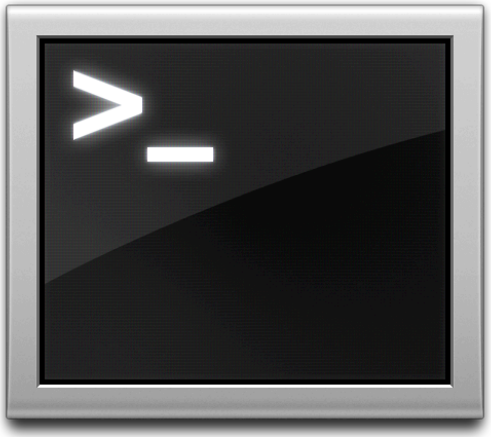
fdesetup list

sudo fdesetup list



To get a list of all FileVault 2 enabled accounts on your Mac, you would run the command shown on the screen. All enabled accounts will be listed with both the accounts' username and UUID.

fdesetup remove

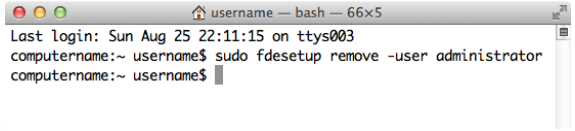


- Removes accounts from the list of FileVault 2 enabled accounts
  - Can disable using account username
  - Can disable using account UUID

To remove accounts from the list of FileVault 2 enabled accounts, fdesetup includes the remove verb. You can remove users by using either the username or the account's UUID.

fdesetup remove -user

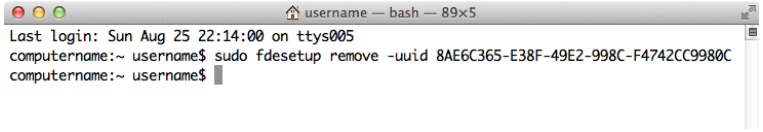
sudo fdesetup remove -user username



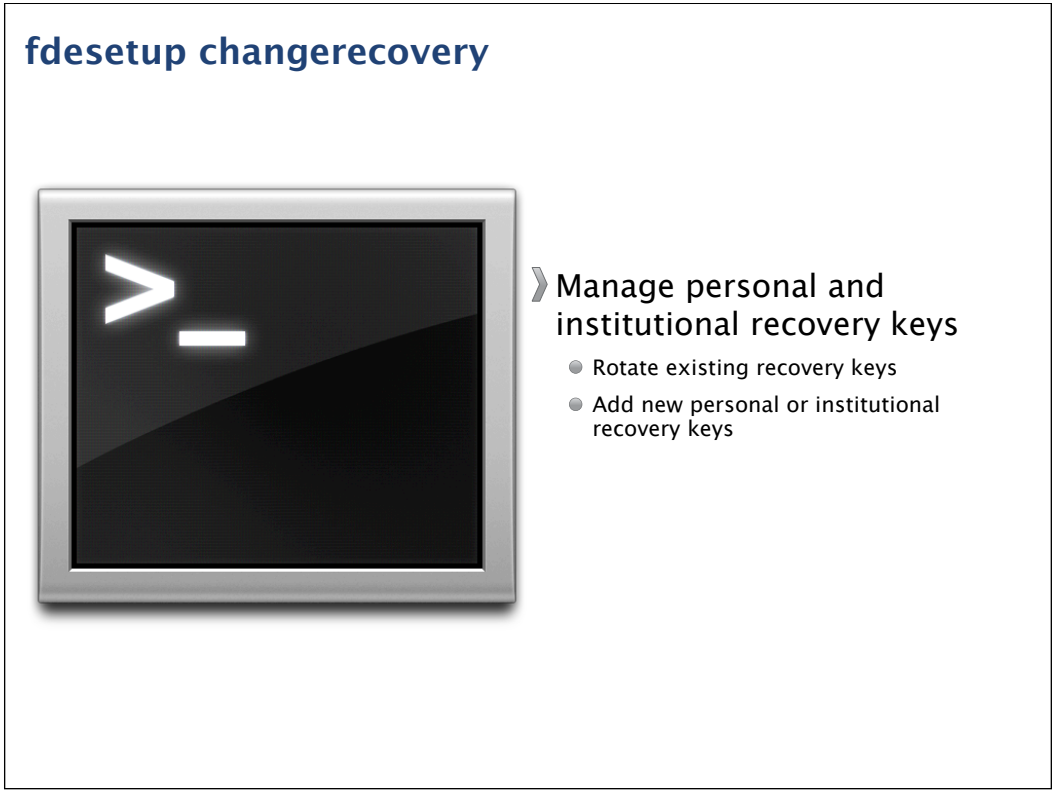
To remove the account by username, you would run the command as shown on the screen and provide the account's username.

fdesetup remove -uuid

sudo fdesetup remove -uuid uuid_here



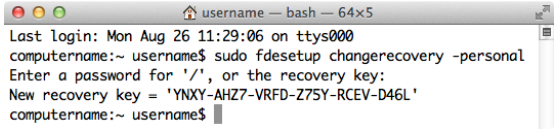
To remove the account using the UUID, you would run the command as shown on the screen and provide the account's UUID.



fdesetup in Mavericks adds the new ability to change, add and remove both personal and institutional recovery keys. This gives Mac admins much greater ability to manage recovery keys, including the capability to quickly update or remove compromised personal or institutional recovery keys in the event of a data breach or other problem.

fdesetup changerecovery -personal

sudo fdesetup changerecovery -personal



To change to a new personal key, you would run the command shown on the screen. You'll be prompted for the password of an existing FileVault 2-enabled user or an existing personal recovery key. Once entered, a new personal recovery key will be generated and displayed. The former personal recovery key will no longer work.

fdesetup changerecovery -personal -inputplist

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Password</key>
<string>password</string>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

For those who want to automate the process, fdesetup supports importing a properly formatted plist via standard input. The plist needs to follow the format shown on the screen. You would store either the password of an existing FileVault 2-enabled user or the existing personal recovery key in the Password key in the plist.

fdesetup changerecovery -personal -inputplist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>KeychainPath</key>
<string>/path/to/filename.keychain</string>
<key>KeychainPassword</key>
<string>password</string>
</dict>
</plist>
```

Note: All keychain passwords need to be supplied in cleartext.

If applicable, you can also import a plist which references a keychain containing the public and private key for an institutional recovery key. For the KeychainPath key, you will need to provide the file path to the keychain as the plist value and for the KeychainPassword key, you will need to provide the password that unlocks that keychain.

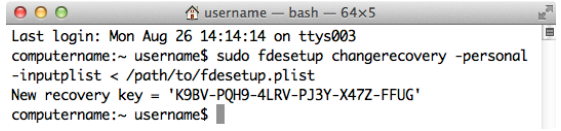
Using KeychainPath and KeychainPassword in plists

One thing to know about using the KeychainPath and KeychainPassword keys in a plist file is the fdesetup man page notes that KeychainPath only works with certain commands. Regrettably, it doesn't mention which commands those are, so I did some testing to find out. As of OS X 10.9.4, it appears that you can leverage KeychainPath and KeychainPassword with the following two commands:

fdesetup changerecovery
fdesetup removerecovery

fdesetup changerecovery -personal -inputplist

sudo fdesetup changerecovery -personal -inputplist < /path/to/filename.plist

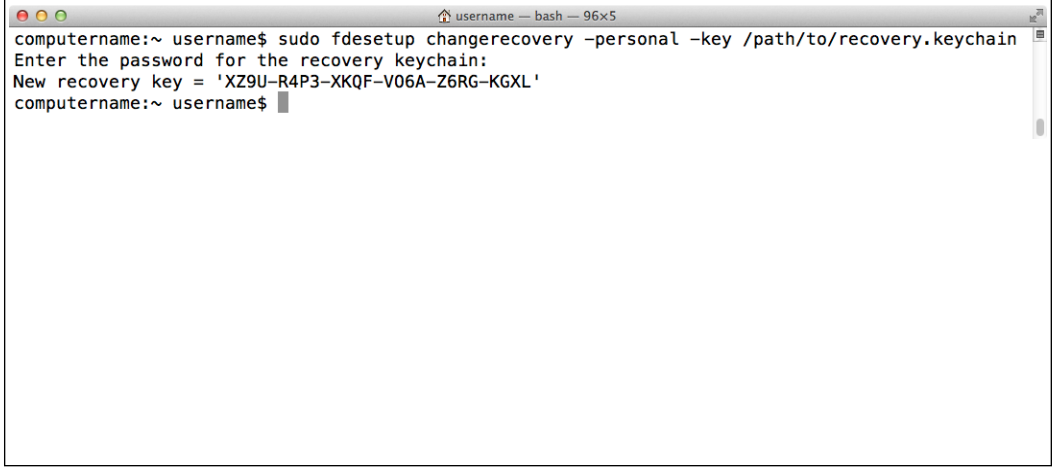


To change to a new personal key, you would run the command shown on the screen to reference the information in the plist file and change to a new personal recovery key. The former personal recovery key will no longer work.

In the event that the Mac in question does not have a personal recovery key, running the described commands will add a personal recovery key instead of changing an existing one.

fdesetup changerecovery -personal -key

sudo fdesetup changerecovery -personal -key /path/to/ recovery.keychain

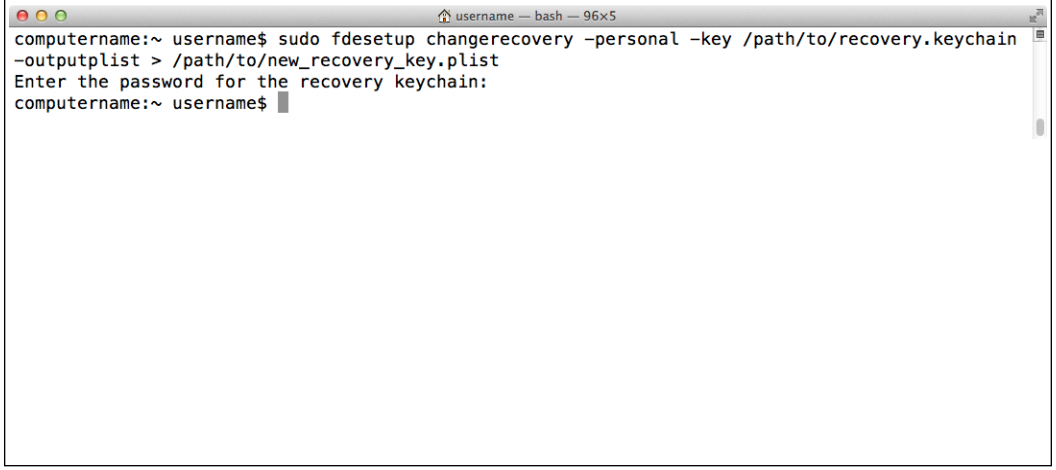


The changerecovery function also allows the use of an institutional keychain to authorize the generation of a new personal recovery key. The institutional keychain would need to have both the private and public keys inside.

To do this, you would run the command as shown on the screen. You'll be prompted for the password to unlock the institutional recovery keychain. Once that password is provided, a new personal recovery key will be generated and displayed.

fdesetup changerecovery -personal -key -outputplist

sudo fdesetup changerecovery -personal -key /path/to/ recovery.keychain -outputplist > /path/to/ new_recovery_key.plist



fdesetup can also export the recovery key to a plist file by using the outputplist flag. To generate a new personal recovery key and have it exported to a plist, you would run the command shown on the screen.

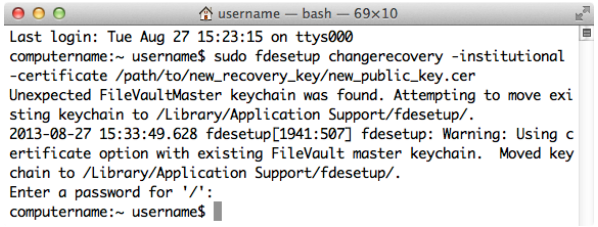
fdesetup changerecovery -personal -key -outputplist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Change</key>
    <true/>
    <key>EnabledDate</key>
    <string>2013-12-20 13:51:58 -0500</string>
    <key>HardwareUUID</key>
    <string>00000000-0000-1000-8000-000C2991B2C4</string>
    <key>HasMasterKeychain</key>
    <true/>
    <key>RecoveryKey</key>
    <string>MLZA-NZTC-MVLM-O82Q-Y8TW-F8FX</string>
    <key>SerialNumber</key>
    <string>VM401BlpPKGn</string>
</dict>
</plist>
```

The exported plist should contain information similar to what's shown on the screen and store the new personal recovery key in the RecoveryKey value.

fdesetup changerecovery -institutional

sudo fdesetup changerecovery -institutional -certificate /path/to/filename.cer



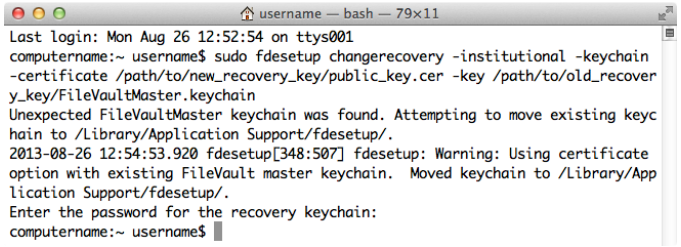
To change to a new institutional recovery key, you will need to have the new public key available. If you have a new institutional public key available as a DER encoded certificate file, you can run the command shown on the screen to replace the current institutional key.

If an institutional keychain is being used on this Mac, you will see a message that an existing FileVault Master keychain was found and moved. The reason for this is that, as part of this process, the current institutional key's FileVaultMaster keychain is replaced with a new FileVaultMaster keychain that includes the new institutional recovery key's public key.

While the former institutional key's FileVaultMaster keychain was moved and not deleted from the Mac, the former institutional recovery key will no longer work.

fdesetup changerecovery -institutional

sudo fdesetup changerecovery -institutional -keychain
-certificate /path/to/filename.cer
-key /path/to/filename.keychain



You can also use the current institutional recovery key to authenticate the change to the new institutional key. If you have a keychain file available containing both the public and private keys for the current institutional key, you can run the command shown on the screen to enable the change to the new institutional key.

As part of this process, you'll be prompted for the keychain password for the institutional key that's being replaced. Once entered, the current institutional key will be replaced with the new one.

fdesetup changerecovery -institutional -inputplist

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Password</key>
<string>password</string>
<key>Certificate</key>
<data>
MIIDLjCCAhagAwIBAgIENsqyWTALBgkqhkiG9w0BAQUwSTEfMB0GA1UEAwwWRmlsZVZhdWx0
IFJlY29ZZXJ5IEtleTEmMCQGA1UEDQwddHJvdXRvbnQtdm0zLnRhb21lY2h3b3Jrcy5uZXQw
HhcNMTMwODI0MjEzNTI1WhcNMTQwODI0MjEzNTI1WjBJMR8wHQYDVQQDDBZGaWxlVmF1bHQg
UmVjb3ZlcnkgS2V5MSYwJAYDVQQNDB10cm91dG9udC12bTMudGFvbWVjaHdvcmtzLm5ldDCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMJpO4TYyH8HC3H4WWV0FPV48I1cdRjh
BGi9OYJX8cQrgWb72FTDCNbU+fD1ce1++KALheMNCmhOIwPshQtbUuU9ngpfRYI/
L6KIgIaUW4bpUksHXZ0CIwSvfqD6+
np06wniS48KNPmnyHqxNeYyfU20wAITPznP7w0VjU7pMbvvJra03gNDxxLc44Rj5YdHxn7U7
gIUNc08RxntNgC6hPSMuZVIgz8AUFcOYuBYML9fXrHytNoaZ0JXS2djGRdwRT0V3OtK2qDFh
8+vVr9vTBt3xmk6hPREXfB9odRDH6ZP9mn6R2vLLzF4ANwklha0L4+
mwbPtdfVOSs3wdkoUFF8CAwEAAaMgMB4wCwYDVR0PBAQDAgK0MA8GA1UdEwEB/
wQFMAMBAQAwDQYJKoZIhvcNAQEFBQADggEBAJ+5ZaqzQmnVDrIQBkxiBXql/
aRbQpetu5jLsJtHDxRd1d2YiEFJHwy4jnUefrvTUf7648oCQ5FtOjYL5lYn7QQ3MjvsFKzBh
2wVlqJJAhjjjwbzMoVdRxfY8Aftcler9Tw/WvxiECvdgxBA37iA+EGVTrRUGuCytY+
GhawIKQMR5Ac2XBpfgPatbV9xAtCo4+M9I4jqhSxOGMwLjRHs4yELyH71w/Au3VlTYezf+
dqofOaZ8c4wBOK3T4VeoWMJjjwDe9Q3/j3sf/CjxvNhm1yYRUqUieZx7Vdf2+
oKKqMczyAVz0Y8UX26lMnpMQn4Fv9b5mOvo5U2/ZgpiQ8cHxM=
</data>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

For those who want to automate the process, fdesetup also supports importing a plist via standard input. You would store either the password of an existing FileVault 2-enabled user or an existing personal recovery key in the Password key and have the public key data stored in the Certificate key of the plist.

fdesetup changerecovery -institutional -inputplist

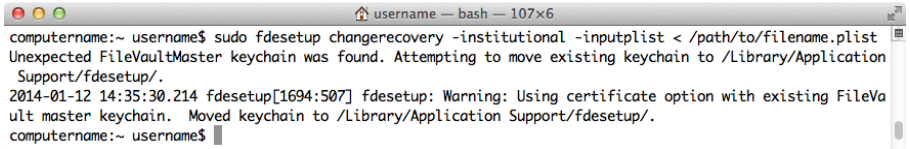
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>KeychainPath</key>
<string>/path/to/filename.keychain</string>
<key>KeychainPassword</key>
<string>password</string>
<key>Certificate</key>
<data>
MIIDCDCCAfCgAwIBAgIEXKugDjALBgkqhkiG9w0BAQUwNjEfMB0GA1UEAwwWRmlsZVZhdWx0IFJlY292ZXJ5IEtleTETMBEGA1UEDQw
KZWJvbi1zaGlueTAeFw0xNDA3MDQyMzM1MjRaFw0xNTA3MDQyMzM1MjRaMDYxHzAdBgNVBAMMFkZpbGVWYXVsdCBSZWNvdmVyeSBLZX
kxEzARBgNVBA0MCmVib24tc2hpbnkwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDRNNxuc8ilkxDcDX+
wrm4ut1M8QErWjehdrVpCGA6ipCcyeYL4s8ZBgi/
ObI43q0vwgTvTVHrzzU46RIKGQcC5zYhvbvuFYIYjAgnVfaobW8xWEfRwSGihzpljlNcyvV1jQre4N6i/iIPBY8UR1svME+
1PEDUURgB/XXyhgWlIzu8V5RrUAkdLh3HiOd+XctDs3sDigE69M+I0oThcgdN8tx3ZbGFUv+4vvMZH3bRc2Lc7goC44M5i+
dpdR1tc0xpcwxq4XYkkA/dcA+AauS95FoStVXgXBOCnfK5RneF5+
AqOibUHNhIZz5dIYvjFDPG3vjjpsIZZdokVeCt0hSLlAgMBAAGjIDAeMAsGA1UdDwQEAwICtDAPBgNVHRMBAf8EBTADAQEAMA0GCSqG
SIb3DQEBBQUAA4IBAQCeeDPE7u++bYlrbPBWcySHRN8L/0Kqp5F554krJO6nFOFcwpgNBgZIyLzKx7gqYrJA3RHeT/yVysJ40/
4ZOZJjjr6Sze6vIpPdahd5xL3QKr1A2ILBBKJtGBfgsevPfi9Gs4G/
PHb0LcEeT9mLm8gAlChK5V2HA00R3b31S5Zcw5rUNNxzSqSy4ZZzlnpY4WC7pzEDZ1KIUnIQZm1r3CkWdqzREzeW90o8VhnY3ypitNP
h+BmzbUks1zHccxJSCFp1cSZQw47nceqWtZkRcsLiSpwlPfXPY7wnrbJriPN9Yz0MFax5fXVWHefDXPy/lBCt9milfz30EGkOk0q/
AxD3
</data>
</dict>
</plist>
```

Note: All keychain passwords need to be supplied in cleartext.

You can also use a plist which references a keychain containing the public and private key of the current institutional key. For the KeychainPath key, you will need to provide the file path to the keychain as the plist value, for the KeychainPassword key, you will need to provide the password that unlocks that keychain, and store the public key data in the Certificate key.

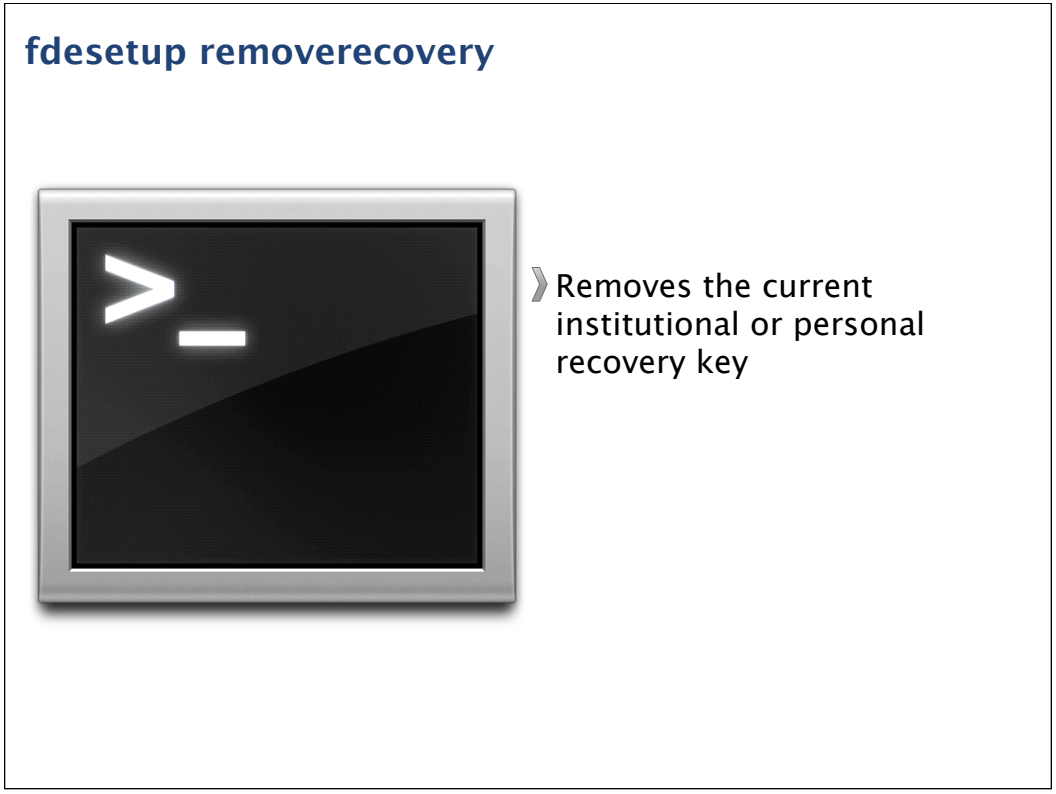
fdesetup changerecovery -institutional -inputplist

sudo fdesetup changerecovery -institutional -inputplist < /path/to/filename.plist



To change to a new institutional key, you would run the command shown on the screen to reference the information in the plist file and change to a new institutional recovery key. Once changed, the former institutional recovery key will no longer work.

In the event that the Mac in question does not have a institutional key, running the previously described commands will add a institutional key instead of changing an existing one.

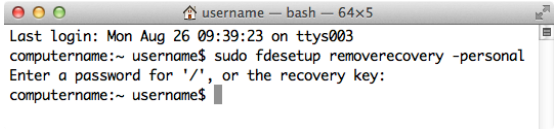


To remove recovery keys, fdesetup includes the removerecovery verb. You can remove the personal recovery key or the institutional recovery key. Once removed, the recovery keys will no longer work.

One thing to note is that you can also use this command to remove all of the current recovery keys associated with your encrypted Mac. Once the recovery keys have been removed from your Mac, only FileVault 2-enabled accounts will be able to unlock or decrypt it.

fdesetup removerecovery -personal

sudo fdesetup removerecovery -personal



To remove an existing personal key, run the command shown on the screen. You'll be prompted for the password of an existing FileVault 2-enabled user or the existing personal recovery key. Once entered, the personal recovery key will be removed from the system.

fdesetup removerecovery -personal -inputplist

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Password</key>
<string>password</string>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

For those who want to automate the process, fdesetup also supports importing a plist via standard input. The plist needs to follow the format shown on the screen, with the password of an existing FileVault 2-enabled user or the existing personal recovery key stored in the Password key in the plist.

fdesetup removerecovery -personal -inputplist

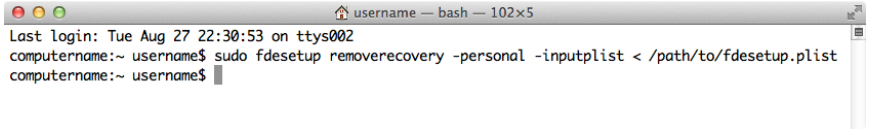
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>KeychainPath</key>
<string>/path/to/filename.keychain</string>
<key>KeychainPassword</key>
<string>password</string>
</dict>
</plist>
```

Note: All keychain passwords need to be supplied in cleartext.

If applicable, you can also use a plist which references a keychain containing the public and private key for an institutional recovery key. For the KeychainPath key, you will need to provide the file path to the keychain as the plist value and for the KeychainPassword key, you will need to provide the password that unlocks that keychain.

fdesetup removerecovery -personal -inputplist

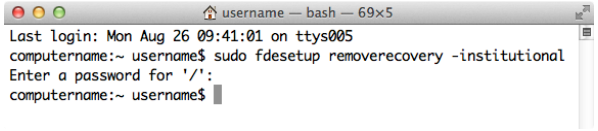
sudo fdesetup removerecovery -personal -inputplist < /path/to/filename.plist



To remove the existing personal key, you would run the command shown on the screen to reference the information in the plist file and remove the current personal recovery key.

fdesetup removerecovery -institutional

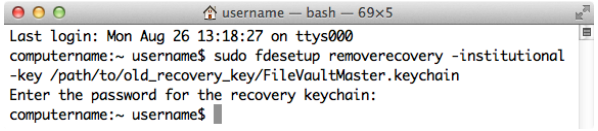
sudo fdesetup removerecovery -institutional



To remove an existing institutional key, run the command shown on the screen. You'll be prompted for the password of an existing FileVault 2-enabled user. You can also use an existing personal recovery key if applicable.

fdesetup removerecovery -institutional -keychain

sudo fdesetup removerecovery -institutional -keychain -key /path/to/filename.keychain



You can also use the current institutional recovery key to authenticate the removal of itself. If you have a keychain file available containing both the public and private keys for the current institutional key, you can run the command shown on the screen to remove the current institutional key. You'll be prompted for the keychain's password. Once entered, the current institutional key will be removed and will no longer work.

fdesetup removerecovery -institutional -inputplist

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Password</key>
<string>password</string>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

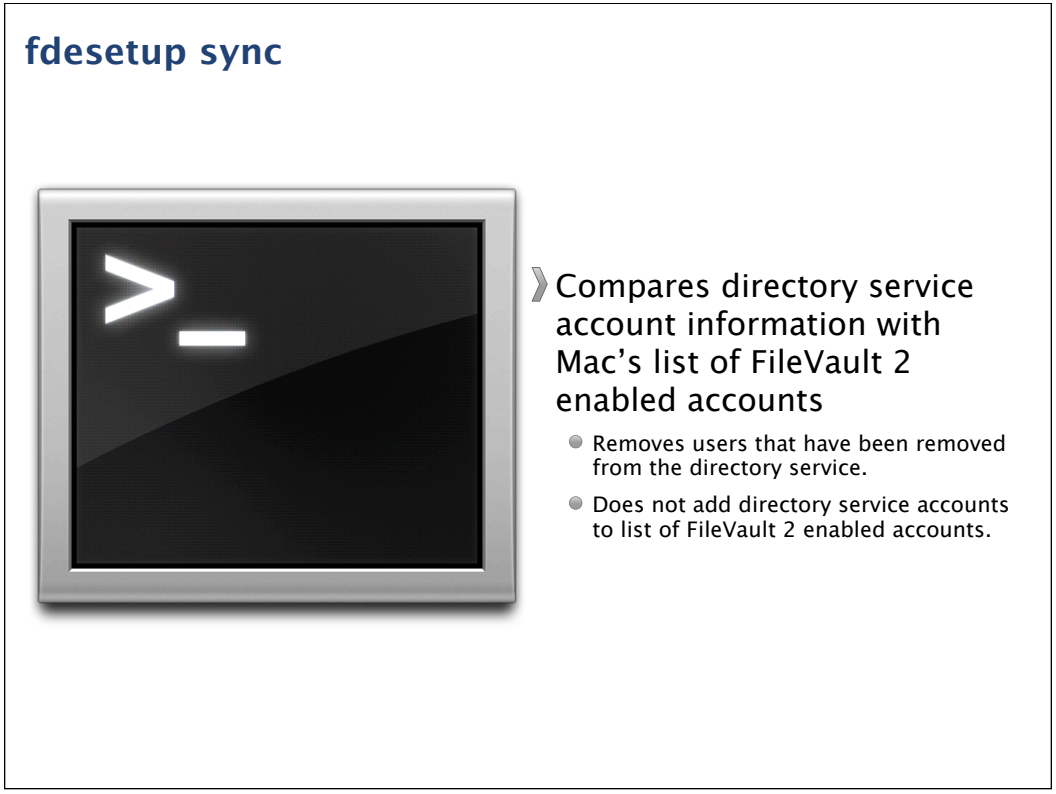
For those who want to automate the process, fdesetup also supports using a plist to remove the institutional key. You would store either the password of an existing FileVault 2-enabled user or an existing personal recovery key in the Password key in the plist.

fdesetup removerecovery -institutional -inputplist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>KeychainPath</key>
<string>/path/to/filename.keychain</string>
<key>KeychainPassword</key>
<string>password</string>
</dict>
</plist>
```

Note: All keychain passwords need to be supplied in cleartext.

You can also use a plist which references a keychain containing the public and private key of the institutional recovery key that you want to remove. For the KeychainPath key, you will need to provide the file path to the keychain as the plist value and for the KeychainPassword key, you will need to provide the password that unlocks that keychain.



fdesetup also has the sync verb, which allows FileVault 2 to check with the Mac's directory service and see which accounts have been changed. Its main use currently is to automate the disabling of FileVault 2-enabled accounts by checking the directory service to see which accounts have been removed. If an account has been removed from the directory service, running fdesetup sync on an encrypted Mac will automatically remove the account from the list of FileVault 2 enabled accounts. The sync only affects the account's FileVault 2 status and will not remove the account or account home folder from the Mac.

One important thing to know is that sync does not allow accounts to be automatically added, only removed.

fdesetup authrestart

fdesetup authrestart will allow a one-time restart of a FileVault 2-encrypted Mac which goes to the regular login window instead of the FileVault 2 pre-boot login screen.

As this is something that's best shown, here's what happens when "fdesetup authrestart" is executed on a Mac running Mavericks that's encrypted with FileVault 2 . When you run the command, it asks for a password or recovery key. The password must be an account that has been enabled for FileVault 2. After that, it puts an unlock key in system memory and reboots. On reboot, the reboot process automatically clears the unlock key from memory.

fdesetup authrestart -inputplist

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Password</key>
<string>password</string>
</dict>
</plist>
```

Note: All account passwords need to be supplied in cleartext.

For those who want to automate the process, fdesetup also supports using a plist to authenticate the authrestart process. You would store either the password of an existing FileVault 2-enabled user or an existing personal recovery key in the Password key in the plist.

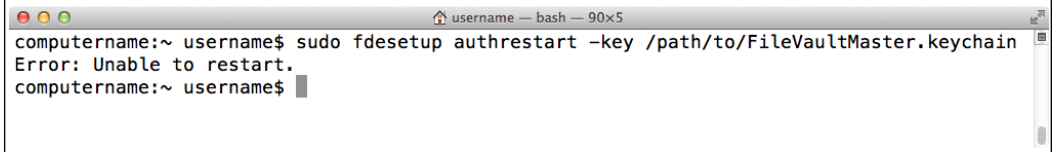
fdesetup authrestart -inputplist

sudo fdesetup authrestart -inputplist < /path/to/filename.plist

To run authrestart this way, you would run the command shown on the screen to reference the information in the plist file. The authrestart process will authenticate and then reboot the machine.

fdesetup authrestart -key

sudo fdesetup authrestart -key /path/to/filename.keychain



Apple Bug Report ID: 17423687

http://openradar.appspot.com/radar?id=6385064946434048

According to the fdesetup man page, you can also authenticate using a keychain and you would run the command shown on the screen to do this. However, as of 10.9.4, this particular function does not appear to be working properly and I've filed a bug report with Apple about it. If you want to also file a bug report on this, please submit it at bugreport.apple.com and reference the ID number on the screen when submitting your report.

I've also got the details of my bug report posted at Open Radar, available from the link below the bug ID.

There are a number of FileVault 2 management solutions that use fdesetup to manage FileVault 2, available from JAMF Software, Dell and open source projects.

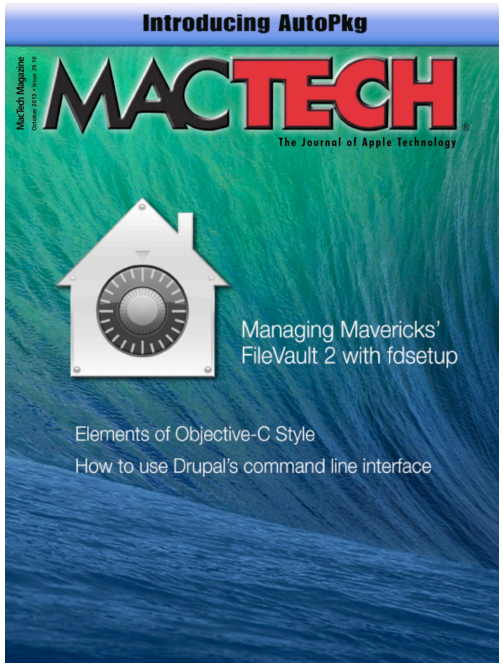
	Name	Supported OSs	Recovery Key Support	Vendor
	Cauliflower Vest	10.7.x –10.9.x	Individual	Open Source
	The Casper Suite	10.8.x –10.9.x	Individual and Institutional	JAMF Software
	Dell Data Protection | Encryption	10.7.x –10.9.x	Institutional	Dell
	Crypt	10.8.x –10.9.x	Individual	Open Source
	FileVault Setup.app	10.8.x –10.9.x	Individual and Institutional	Open Source

The grid shows a listing of the management solutions that I've worked with that utilize fdesetup. All have their strengths, so I recommend evaluating them carefully to find the one that meets your needs. This is also not a comprehensive list; there's even more out there.

fdesetup = FileVault 2 multi-tool

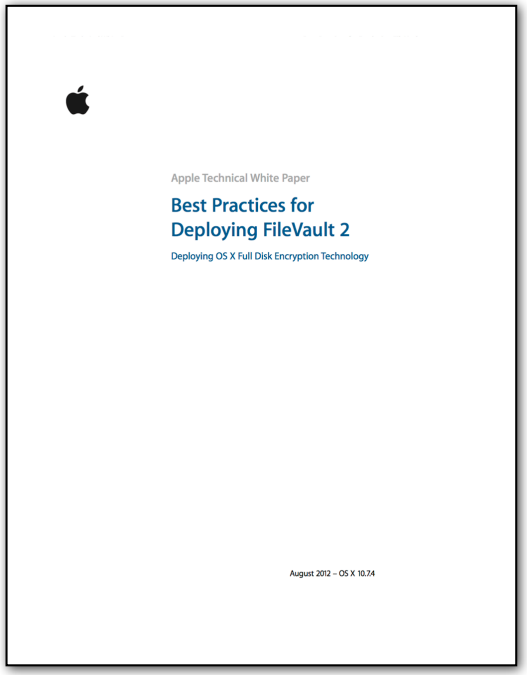
fdesetup is a Swiss Army knife for managing FileVault 2. It can enable FileVault 2, add and remove users, report on FileVault 2's status, rotate recovery keys and more. If you're managing FileVault 2 encryption in your own environment, I recommend using this tool. Properly used, it will save you time and give encryption options available with no other software.

Additional information



If you want more information about managing 10.9's FileVault 2 with fdesetup, I recommend checking out the October 2013 issue of MacTech. It's available via the MacTech iPad app and print copies should be available for ordering.

Additional information



For more general information on FileVault 2, Apple has released a white paper that describes best practices for deploying FileVault 2 on OS X Lion. Because it is focused on Lion, it does not cover fdesetup but it does include a lot of interesting technical detail on how FileVault 2 works.

FileVault 2 talk videos

- Penn State MacAdmins 2013 - http://youtu.be/fsxtNHj_IY8
- MacSysAdmin 2013 – http://docs.macsysadmin.se/2013/2013doc.html
- JAMF National User Conference 2013 – http://www.youtube.com/watch?v=ySYs_z_DYiU

This talk has covered just fdesetup but more information about FileVault 2 is available from other talks that I've given. The video links for these talks are available here.

Links

- Apple Best Practices for Deploying FileVault 2 - http://training.apple.com/osx
- Managing Mavericks' FileVault 2 with fdesetup: http://derflounder.wordpress.com/2013/10/22/managing-mavericks-filevault-2-with-fdesetup/
- Using a FileVault 2 institutional recovery key in Mavericks to generate an individual recovery key: http://derflounder.wordpress.com/2013/12/20/using-a-filevault-2-institutional-recovery-key-in-mavericks-to-generate-an-individual-recovery-key/
- Referencing a FileVault 2 institutional recovery key as part of an fdesetup plist file in Mavericks: http://derflounder.wordpress.com/2014/07/05/referencing-a-filevault-2-institutional-recovery-key-as-part-of-an-fdesetup-plist-file-in-mavericks/
- Enabling users for FileVault 2 with a non-enabled admin user does not work in Mavericks: http://derflounder.wordpress.com/2013/10/24/enabling-users-for-filevault-2-with-a-non-enabled-admin-user-does-not-work-in-mavericks/
- Disabling FileVault 2 with fdesetup on Mountain Lion and Mavericks: http://derflounder.wordpress.com/2014/03/22/disabling-filevault-2-with-fdesetup-on-mountain-lion-and-mavericks/
- Macs that support authenticated restart with FileVault: http://support.apple.com/kb/HT6116

Here's some useful links for FileVault 2 and fdesetup, including links to some topics not discussed as part of today's talk.

PDF available from the following link:

http://tinyurl.com/PSUMac2014PDF

Keynote slides available from the following link:

http://tinyurl.com/PSUMac2014key

Feedback link: http://j.mp/psumac46

As mentioned earlier, here are the download links for this talk. It's available in PDF format with the speakers notes and you can also download the Keynote slides with all of the speakers notes and demos.