TS DOCS
TS-800

Configuring SSH Client Software in UNIX and Windows Environments for Use with the SFTP Access Method in SAS® 9.2, SAS® 9.3, and SAS® 9.4

Table of Contents

Overview

During execution of the SFTP access method in a SAS® FILENAME statement, the method dynamically launches an sftp or a psftp executable file. Launching this executable initiates an SSH client process that creates a secure connection to an OpenSSH server. All conversations across this connection are encrypted—from the user authentication to the data transfers. However, before you can use the SFTP access method in a SAS FILENAME statement, you must set up the SSH software that enables SAS to access a file on an OpenSSH server.

Only Windows and UNIX operating systems can access an OpenSSH server on another UNIX system. To access an OpenSSH server, UNIX systems require OpenSSH software, and Windows systems require PuTTY software. Currently, only the OpenSSH client and server that supports protocol level SSH-2 has been tested in UNIX environments. In Windows environments, only the PuTTY client has been tested. Other third-party applications that support the SSH-2 and SFTP protocols currently are untested and, therefore, SAS does not support these applications. Because of the secure nature of the OpenSSH SSHD server, public-key authentication is required. An SSH agent is also required, which avoids frequent key validations. The SSH agent is available for authentication service only while the Windows session or UNIX shell is established. When a session or a shell that has a running SSH agent is terminated, access to that agent by the SSH client is also terminated. You must add the SSH agent again when the session is re-established.

To understand the configuration options that are required for the OpenSSH and PuTTY clients and the OpenSSH SSHD server, it is recommended that you have a copy of the book *SSH: The Secure Shell—The Definitive Guide,* by Daniel J. Barrett, Richard E. Silverman, and Robert G. Byrnes. This book is an invaluable resource when you are configuring the SSH applications, and it describes in detail topics that include public key authentication, SSH agents, and SSHD host keys.

Configuring OpenSSH Software in UNIX Environments

Under UNIX, you use OpenSSH software to access your UNIX OpenSSH server. However, first you must configure OpenSSH, as follows, in order to access the SFTP access method under UNIX:

1. Install the OpenSSH software that includes the sftp executable file. **You must install** this software in a directory that is accessible via the UNIX $PATH search path or via the PATH environment variable.

   When you submit the SFTP access method in a FILENAME statement, the SFTP access method searches the directories in your PATH environment variable for the sftp executable file in order for it to be executed. **Note:** The first sftp executable file that the software finds in the PATH environment variable is executed.

2. Create RSA or DSA key pairs for public-key authentication. The public-key authentication method for user validation can use a variety of different encryption algorithms. SAS tests and supports both the RSA and the DSA encryption algorithms. Other encryption algorithms should work, but SAS has not tested them. To create the key pairs, follow these steps:

   a. Create an .ssh directory in your home directory. **Example:** For the user ID abcuid, you would create the directory **`/abcuid/.ssh`**.

   b. Issue the **ssh-keygen** command. This command prompts you for a secret passphrase that protects your private key. When you set up a passphrase, it is recommended that you assign 10-15 characters. Blank spaces are allowed in the passphrase.

   c. To create an RSA key pair, enter the following commands from a UNIX prompt. In this sequence, *key-filename* is the name (for example, abckey) for the key files that you create:

```
cd ~your-home-directory
mkdir .ssh
ssh-keygen –f key-filename –t rsa
```

The private- and public-generated keys are saved in two user-named files (for example, abckey and abckey.pub) in your local directory. If for some reason the key files are not created in the $HOME/.ssh directory, be sure to copy both key files to that location.

Once created, the private key file is readable only by your account. In addition, its contents are further protected by encryption that uses the secret passphrase that you create during the key-generation process. Ensure that only your account can write to the associated .ssh directory and files by submitting the following commands:

```
chmod 755 ~/.ssh   /* You can also use 700 here. */
chmod 644 $HOME/.ssh/authorized_keys
chmod 751 $HOME/
```

3. Install your public key on a UNIX OpenSSH server machine. To do that, create or edit the OpenSSH SSHD configuration directory's **`~/.ssh/authorized_keys`** file on the OpenSSH server machine so that it contains the contents of the public key (.pub) file that you generated previously on the client machine. A typical authorized_keys file contains a list of public keys, one key per line.

   Note: Most likely, a system administrator with update permission will have to perform this step.

   You must copy and append the public-key file (.pub) to the server's authorized_keys file. You can do that by submitting the following command at a UNIX prompt :

   ```
   ssh–copy-id –i key-name user-id@host-name
   ```

   This command enables you to log on to the server machine to install the public key, and concatenation of the key is automatic. Make sure that the server's authorized_keys file (**`~/.ssh/authorized_keys)`** ends with a newline character. If a newline character is not present, the last two keys will be concatenated and corrupted.

   If this method does not work correctly, you can also use the following method:

   a. Use File Transfer Protocol (FTP) or Secure File Transfer Protocol (SFTP) to transfer the public key to a file (for example, a file named a.pub) on the server's machine.

   b. Then append the public key to the end of the authorized_keys file by submitting the following command from a UNIX prompt:

      ```
      cat a.pub >> authorized_keys
      ```

4. Use an SSH agent to load your private key on the client machine , as follows:

   a. Execute the ssh-agent executable file on the OpenSSH client by submitting the following commands from a UNIX prompt:

      ```
      ssh-agent $SHELL /* Executes an OpenSSH agent.            */
      ssh-add abckey   /* Adds the private key, called abckey,  */
                       /* to the agent's list.                  */
      ```

      The executable file is provided by OpenSSH software that supports the SSH-2 and the SFTP protocols. The OpenSSH agent stores your private key in memory and provides authentication services to OpenSSH clients. An agent is required for the SFTP access method in a FILENAME statement. The private key is the key that you generated previously with the **ssh-keygen** command. In this example, the key is called abckey.

   b. Enter the private passphrase for abckey, and the identity is added to **`/abcuid/.ssh/abckey`**.

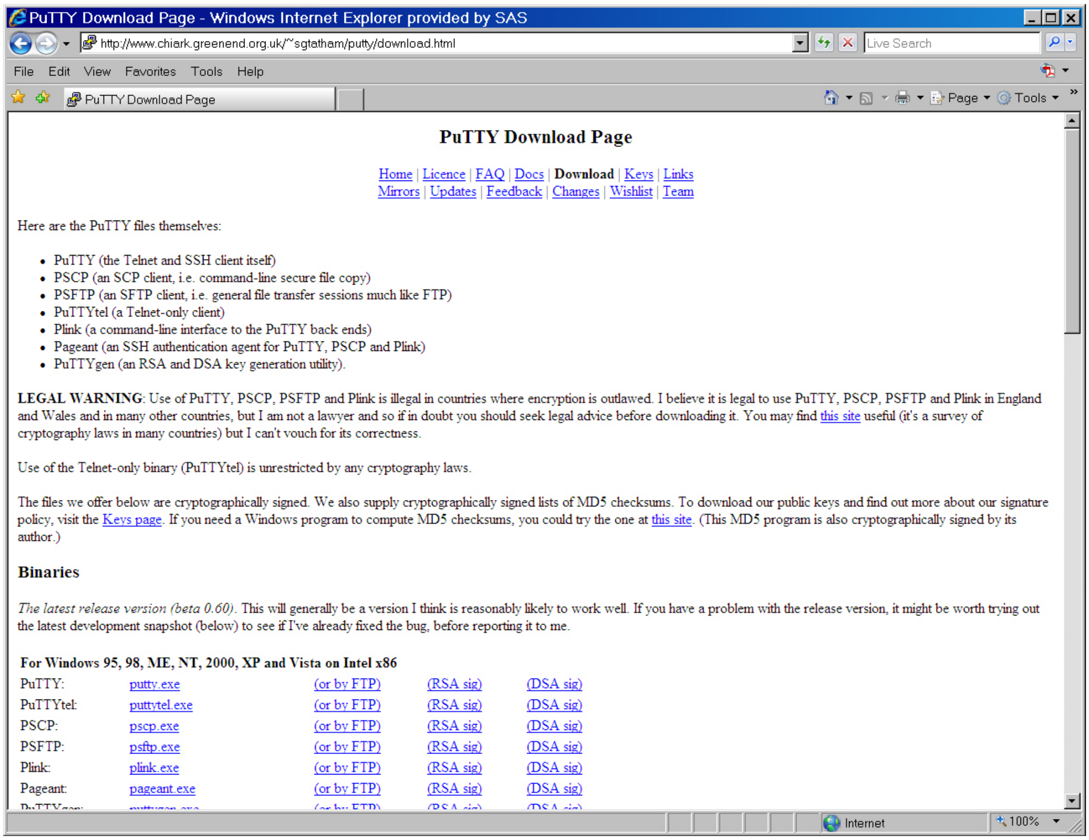
In this step, SHELL is an environment variable that contains the name of your login shell. The agent runs and invokes the shell as a child process. When you submit this **ssh-agent** command, another shell prompt appears that has access to the OpenSSH agent. The **ssh-add** command loads the private key into the OpenSSH agent. Be aware that If you preload an agent at the beginning of an OpenSSH session, you are not prompted again for the passphrase from within this shell when you connect to an OpenSSH server.

Caution: When you use an agent, lock your computer if it you leave so that no one can access the remote accounts that are accessible via your personal keys.

Configuring PuTTY Software in Windows Environments

Under Windows, you use PuTTY software to access your UNIX OpenSSH machine. However, first you must configure the PuTTY software, as follows, to be able to access the SFTP access method under UNIX:

1. Navigate to the PuTTY Download Page, as shown here:


PuTTY Download Page - Windows Internet Explorer provided by SAS
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
Live Search
File Edit View Favorites Tools Help
PuTTY Download Page
Page Tools
Internet
100%

PuTTY Download Page

Home | Licence | FAQ | Docs | **Download** | Keys | Links
Mirrors | Updates | Feedback | Changes | Wishlist | Team

Here are the PuTTY files themselves:

- PuTTY (the Telnet and SSH client itself)
- PSCP (an SCP client, i.e. command-line secure file copy)
- PSFTP (an SFTP client, i.e. general file transfer sessions much like FTP)
- PuTTYtel (a Telnet-only client)
- Plink (a command-line interface to the PuTTY back ends)
- Pageant (an SSH authentication agent for PuTTY, PSCP and Plink)
- PuTTYgen (an RSA and DSA key generation utility).

LEGAL WARNING: Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. I believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England and Wales and in many other countries, but I am not a lawyer and so if in doubt you should seek legal advice before downloading it. You may find this site useful (it's a survey of cryptography laws in many countries) but I can't vouch for its correctness.

Use of the Telnet-only binary (PuTTYtel) is unrestricted by any cryptography laws.

The files we offer below are cryptographically signed. We also supply cryptographically signed lists of MD5 checksums. To download our public keys and find out more about our signature policy, visit the Keys page. If you need a Windows program to compute MD5 checksums, you could try the one at this site. (This MD5 program is also cryptographically signed by its author.)

Binaries

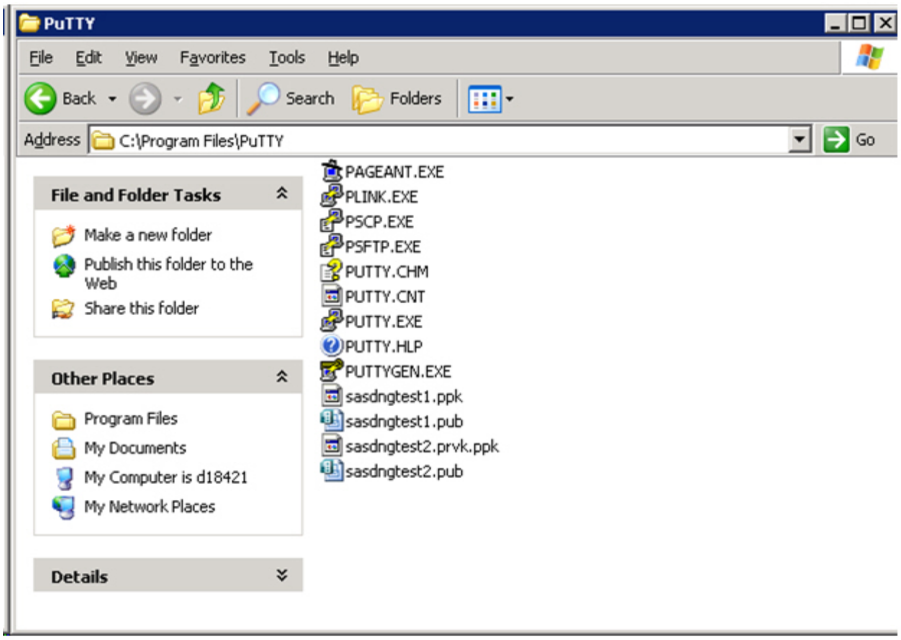
The latest release version (beta 0.60). This will generally be a version I think is reasonably likely to work well. If you have a problem with the release version, it might be worth trying out the latest development snapshot (below) to see if I've already fixed the bug, before reporting it to me.

For Windows 95, 98, ME, NT, 2000, XP and Vista on Intel x86				
PuTTY:	putty.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYtel:	puttytel.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSCP:	pscp.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSFTP:	psftp.exe	(or by FTP)	(RSA sig)	(DSA sig)
Plink:	plink.exe	(or by FTP)	(RSA sig)	(DSA sig)
Pageant:	pageant.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYgen:	puttygen.exe	(or by FTP)	(RSA sig)	(DSA sig)

2. Download the following binary files from the PuTTY Download Page:

- PuTTY
- PSFTP
- Pageant
- PuTTYgen

The files will download to a directory named **`PuTTY`** in your **`C:\Program Files\`** directory, as shown in the following display:



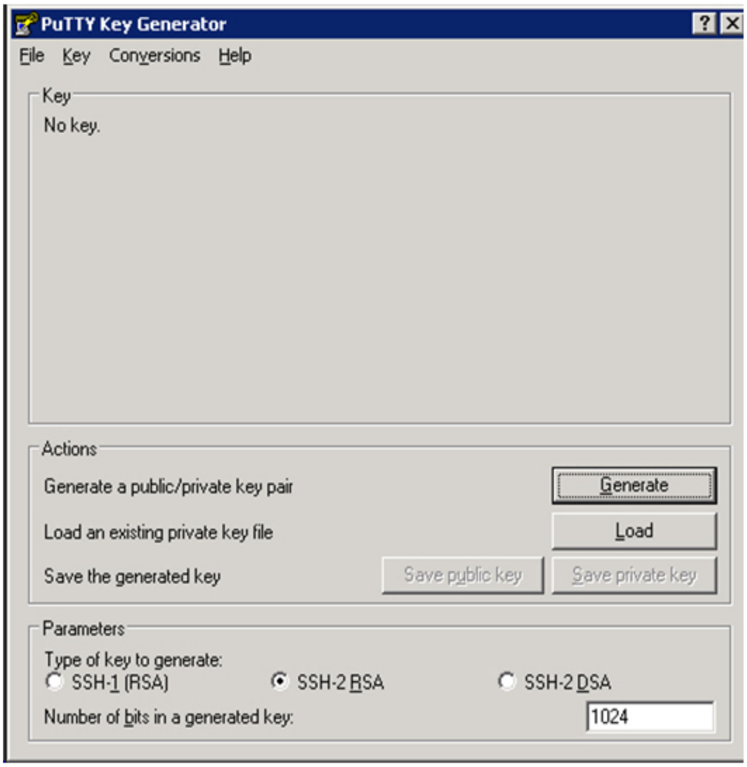
Address C:\Program Files\PuTTY

PAGEANT.EXE
PLINK.EXE
PSCP.EXE
PSFTP.EXE
PUTTY.CHM
PUTTY.CNT
PUTTY.EXE
PUTTY.HLP
PUTTYGEN.EXE
sasdngtest1.ppk
sasdngtest1.pub
sasdngtest2.prvk.ppk
sasdngtest2.pub

Note: The PuTTY software that includes the psftp executable file must be installed in a directory that is accessible via the Windows PATH environment variable. The SFTP access method in the FILENAME statement searches for the path to the psftp executable file in the concatenated list of paths that are contained in that environment variable. The first pfstp executable file that the method finds in the list is executed.

3. Double-click the puttygen.exe file to open the PuTTY Key Generator dialog box.

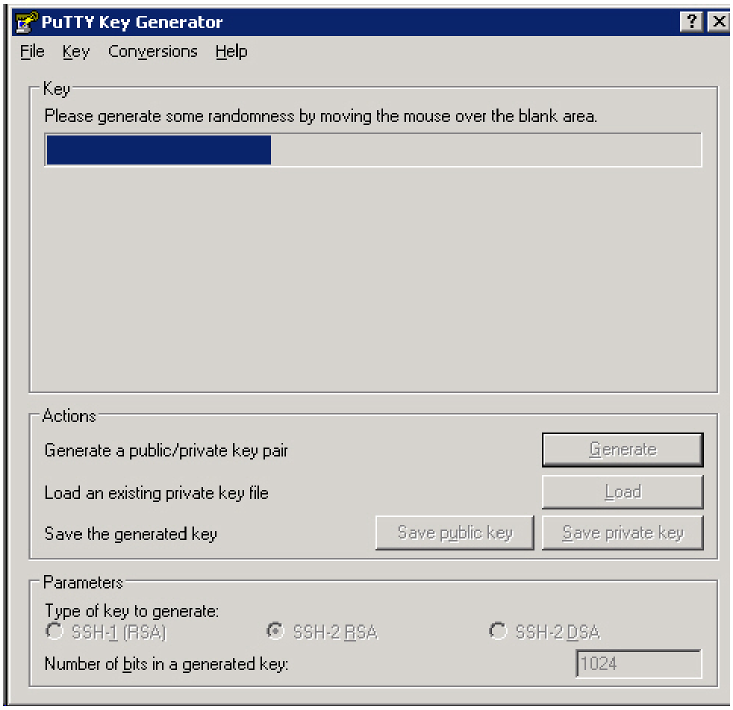
4. Set the **`Type of key to generate`** parameter to either **`SSH-2RSA`** or **`SSH-2DSA`**, and set the value for **`Number of bits in a generated key`** to **`1024`**.


PuTTY Key Generator
File Key Conversions Help
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key

Key
No key.

Parameters
Type of key to generate:
○ SSH-1 (RSA) ● SSH-2 RSA ○ SSH-2 DSA
Number of bits in a generated key: 1024

5. Click the **Generate** button to generate the public and private keys. As shown in the following display, the dialog box prompts you to move your mouse over the blank area in the **Key** box to generate the randomness that is required for public and private keys.



Please generate some randomness by moving the mouse over the blank area.

Generate a public/private key pair

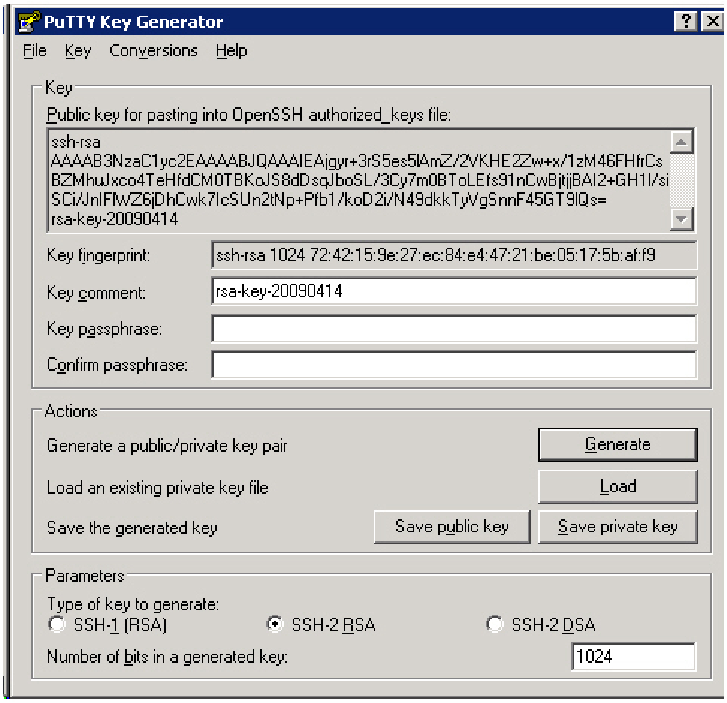
Load an existing private key file

Save the generated key

Type of key to generate:
○ SSH-1 (RSA) ● SSH-2 RSA ○ SSH-2 DSA

Number of bits in a generated key: 1024

6. Enter the key passphrase and the confirmation passphrase in their respective fields.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIEAjgyr+3rS5es5lAmZ/2VKHE2Zw+x/1zM46FHfrCs
BZMhuJxco4TeHfdCM0TBKoJS8dDsqJboSL/3Cy7m0BToLEfs91nCwBjtjjBAI2+GH1l/si
SCi/JnlFlWZ6jDhCwk7IcSUn2tNp+Pfb1/koD2i/N49dkkTyVgSnnF45GT9lQs=
rsa-key-20090414
```

Key fingerprint: ssh-rsa 1024 72:42:15:9e:27:ec:84:e4:47:21:be:05:17:5b:af:f9

Key comment: rsa-key-20090414

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

Load an existing private key file

Save the generated key

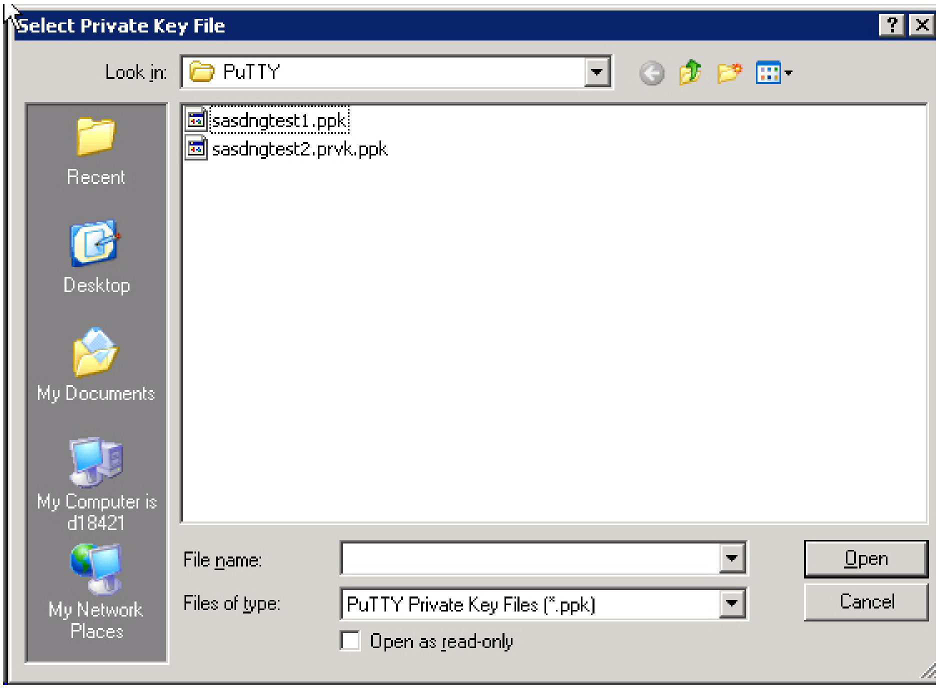
Type of key to generate:
○ SSH-1 (RSA) ● SSH-2 RSA ○ SSH-2 DSA

Number of bits in a generated key: 1024

7. Save both keys to a directory by clicking the **Save public key** and the **Save private key** buttons. In this example, the keys are saved to the **C:/.ssh** directory.

8. Go back to your **`C:\Program Files\PuTTY`** directory and double-click the pageant.exe file to run it. When you execute this file, a terminal icon appears on your desktop's task bar.

9. Right-click the icon and select **`Add Key`** from the menu that appears. This selection displays the Select Private Key File dialog box, as shown in this example:



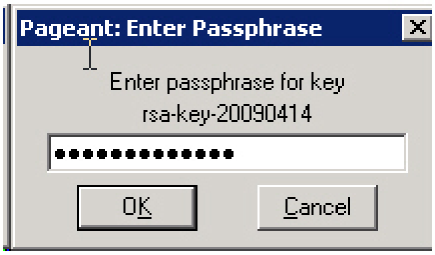
sasdngtest1.ppk
sasdngtest2.prvk.ppk

Files of type: PuTTY Private Key Files (*.ppk)

☐ Open as read-only

10. Select one of the private keys and click **`Open`**.

11. In the Pageant: Enter Passphrase dialog box that appears, enter the passphrase for the private key.



Enter passphrase for key
rsa-key-20090414

●●●●●●●●●●●●●●

Navigate back to the directory that contains your public key. Transfer the public key file to the **`~/.ssh`** directory on the UNIX machine to which you are trying to connect. Use the **psftp** command on a Windows host to transfer the key file to the UNIX host. **Note:** When you begin the transfer, a prompt appears that requests password authentication for the UNIX host. Enter your password to authenticate the prompt, as follows:

```
U:\>psftp doors
Login as: your-user-name
Using keyboard-interactive authentication.
Password: your-password
Remote working directory is /users/your-user-name
psftp>
```

Then you can begin the transfer, as shown in the following example:

```
psftp> cd .ssh       /* The remote directory is now     */
                     /* /folder1/users/user-id/.ssh. */

psftp> lcd C:/.ssh   /* The new local directory is C:\.ssh */

psftp> put puttyzenkey.pub  /* The local file puttyzenkey.pub is moved   */
                            /* to the remote location:                   */
                            /* /folder1/users/user-id                    */
                            /* /.ssh/puttyzenkey.pub.                    */
```

Note: You can also use other methods (such as FTP) to transfer files from Windows environments to UNIX environments.

12. After you transfer the public key to the UNIX host, convert the key file to OpenSSH format, and then append the converted public key to **`~/.ssh/authorized_keys`** on the UNIX machine by submitting the following command from a UNIX prompt:

    ```
    ssh-keygen –i –f {PuTTY-public-key-filename} >>
                      ~/.ssh/authorized_keys
    ```

    This single command enables you to both convert and append the public key at one time.

    If this method does not work correctly, you can also convert and append the public key by submitting the following commands from a UNIX prompt:

    ```
    ssh-keygen –i –f {PuTTY-public-key-filename} >>
                     {OpenSSH-public-key-filename}
    cat openssh-public-key-filename >>
        ~/.ssh/authorized_keys
    ```

At this point, you can use the SFTP access method. For examples using this access method, see “FILENAME Statement, SFTP Access Method: Examples” in *SAS 9.2 Language Reference: Dictionary*, *SAS 9.3 Statements: Reference*, or *SAS 9.4 Statements: Reference.*

Note: When you execute a FILENAME statement on a Windows client machine, the statement must contain the USERID= option.

To execute the SFTP access method, you must preload the private key with the *pageant executable file*, which is provided by the PuTTY software. The pageant executable file must be executed on the Windows client machine. When the pageant executable file is preloaded with a private key, the PuTTY software does not prompt you for authentication. To preload the private key, follow these steps:

1. Execute the pageant executable file on the PuTTY client.
2. Click the pageant icon, then select the private key that you want to preload.
3. Enter the private passphrase for the key.

The pageant executable file then stores your private key in memory and provides authentication services to SSH clients. As mentioned earlier in this section, make sure that you install the psftp executable file in a directory that is accessible via the Windows PATH environment.

The communication between the SSH client and the OpenSSH server is automatic, as needed. The agent lasts until the pageant process is terminated or until you exit the Windows environment.

Caution: When you use an agent, you should lock your computer so that no one can access the remote accounts that are accessible via your personal keys.

Troubleshooting Problems with the SFTP Access Method in the FILENAME Statement

Problem 1

You are unable to authenticate using public-key authentication.

Solution

If you have difficulty authenticating using public-key authentication, you should manually validate whether the SFTP client can access an OpenSSH SSHD server without involving the SAS System, as follows.

For UNIX Hosts

1. Start the agent and load the private key into memory by submitting the following commands from a UNIX prompt:

```
ssh-agent $SHELL
ssh-add ~user-id/.ssh/acRSAkey
Enter passphrase for /users/user-id/.ssh/acRSAkey: your-passphrase
Identity added: /users/user-id/.ssh/acRSAkey(users/user-id
                /.ssh/acRSAkey)
```

2. Next, initiate the SFTP transfer by submitting the following SSH command:

```
sftp user-id@host-name
```

After you submit this command, you might receive the following message:

```
Connecting to host...The authenticity of host 'doors (10.12.9.82)' can't be
established. RSA key fingerprint is
79:9a:3c:1e:ff:88:9f:e8:9c:b4:90:de:a9:b5:e2:df.

Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'doors,10.12.9.82' (RSA) to the list of known
hosts.
```

You should enter **`Yes`** as the response to the prompt **`Are you sure you want to continue connecting?.`** When you enter **`Yes`**, the server's host-machine key is added to your client machine's directory **`~/.ssh/known_hosts`**. In addition, you are given authorization to connect automatically to this host in the future.

For Windows Hosts

1. Execute the psftp.exe file, which is located in **`C:\Program Files\PuTTY`** (the same directory where the pageant.exe file resides). A command-prompt window appears with the following message:

```
psftp: no hostname specified; use “open host.name:” to connect
psftp>
```

2. At the prompt, enter the following commands:

```
Psftp> open userid@machine_name
Using username “userid”
Remote working directory is /users/<userid>/
psftp>
```

The message that appears (`Using username “userid” / Remote working directory is /users/<userid>/`) indicates that you can now manually authenticate.

Problem 2

You receive the following error in the public-key authentication section of an SSHD log when you submit the command **ssh *user-id@host-name* –vvv**:

```
debug2: we did not send a packet, disable method
```

Solution

If you receive this error, you need to ensure that the proper permissions are set on the server host, as follows:

```
chmod 775 ~/.ssh    /* You can also use 700. */
chmod 644 $HOME/.ssh/authorized_keys
chmod 751 $HOME/
```

Appendix: Other Third-Party Applications That Support the SSH-2 and the SFTP Protocols

A batch script, such as the one in the following example, can be useful when you work with other third-party applications that support the SSH-2 and the SFTP protocols. The following example uses a UNIX machine (client1) and a UNIX server (remotehost1). In the example, public-key authentication is set up in the client machine's UNIX shell environment, which is used to execute the runSFTP.bat file.

In the following example, the runSFTP.bat batch file resides on the client1 UNIX host. When you execute the batch file, the runSFTP.bat issues the **sftp** command and securely connects to the remote host (remotehost1) via public-key authentication. The **sftp –b** command reads the SFTP command file (SFTPcdmfile1) and issues the **sftp** commands that are read on the SSH connection to the remote host. In this example, data files are retrieved from the remote host. (See the file SFTPcmdfile1.) At that point, you can execute SAS programs and perform file I/O on the retrieved files.

Note: The **sftp** command files simply transfer the data files that the SAS job is going to process. If SAS is interacting with these files on a secure basis (that is, via SFTP), you might want to delete the files after processing is complete.

Example Batch Script

In the following example, public-key authentication is set up in the client machine's UNIX shell environment, which is used to execute the runSFTP.bat file. If the SSH key is the same on all remote hosts, you need to copy only one key to each host.

Example

1. Submit the following commands to preload the SSH agent:

   ```
   ssh-agent $SHELL
   ssh-add rsakey
   ```

2. Execute the runSFTP.bat file, which contains the following **sftp** command:

   ```
   sftp -b ~/SFTPcmdfile1 userid@remotehost1
   ```

   In this command, the **–b** option executes an SFTP batch file named SFTPcmdfile1.bat. This file contains the following commands:

   ```
   ls
   get state1.dat
   get account1.dat
   quit
   ```

   When the file executes, the SFTP **get** commands transfer data files to the client. This batch file resides on the SSH client host that retrieves files with SFTP commands.

At this point, the script now executes the following command, which runs a SAS job to process the data:

```
runsas process.sas
```

After the data is processed, the batch file ends with the script submitting the following commands to remove the data files:

```
rm state*.dat
rm account*.dat
```

Resources

Barrett, Daniel J., Richard E. Silverman, and Robert G. Byrnes. "SSH:TGD References form the Book" in *SSH: The Secure Shell—The Definitive Guide, Second Edition*. 2005. Sebastopol, CA: O'Reilly Media, Inc. Available at www.snailbook.com/index.html**.**

OpenBSD. (2009). OpenSSH: Keeping Your Communiqués Secret. Available at www.openssh.org/**.**

Tatham, Simon. *PuTTY User Manual.* 2001-2004. Available at the.earth.li/~sgtatham/putty/0.55/htmldoc/.