Previous | Contents | Index | Next

Appendix A: PuTTY FAQ

This FAQ is published on the PuTTY web site, and also provided as an appendix in the manual.

A.1 Introduction

A.1.1 What is PuTTY?

PuTTY is a client program for the SSH, Telnet and Rlogin network protocols.

These protocols are all used to run a remote session on a computer, over a network. PuTTY implements the client end of that session: the end at which the session is displayed, rather than the end at which it runs.

In really simple terms: you run PuTTY on a Windows machine, and tell it to connect to (for example) a Unix machine. PuTTY opens a window. Then, anything you type into that window is sent straight to the Unix machine, and everything the Unix machine sends back is displayed in the window. So you can work on the Unix machine as if you were sitting at its console, while actually sitting somewhere else.

A.2 Features supported in PuTTY

In general, if you want to know if PuTTY supports a particular feature, you should look for it on the PuTTY web site. In particular:

- try the changes page, and see if you can find the feature on there. If a feature is listed there, it's been implemented. If it's listed as a change made *since* the latest version, it should be available in the development snapshots, in which case testing will be very welcome.
- try the Wishlist page, and see if you can find the feature there. If it's on there, and not in the 'Recently fixed' section, it probably *hasn't* been implemented.

A.2.1 Does PuTTY support SSH-2?

Yes. SSH-2 support has been available in PuTTY since version 0.50.

Public key authentication (both RSA and DSA) in SSH-2 is new in version 0.52.

A.2.2 Does PuTTY support reading OpenSSH or `ssh.com` SSH-2 private key files?

PuTTY doesn't support this natively (see the wishlist entry for reasons why not), but as of 0.53 PuTTYgen can convert both OpenSSH and `ssh.com` private key files into PuTTY's format.

A.2.3 Does PuTTY support SSH-1?

Yes. SSH-1 support has always been available in PuTTY.

A.2.4 Does PuTTY support local echo?

Yes. Version 0.52 has proper support for local echo.

In version 0.51 and before, local echo could not be separated from local line editing (where you type a line of text locally, and it is not sent to the server until you press Return, so you have the chance to edit it and correct

mistakes *before* the server sees it). New in version 0.52, local echo and local line editing are separate options, and by default PuTTY will try to determine automatically whether to enable them or not, based on which protocol you have selected and also based on hints from the server. If you have a problem with PuTTY's default choice, you can force each option to be enabled or disabled as you choose. The controls are in the Terminal panel, in the section marked 'Line discipline options'.

A.2.5 Does PuTTY support storing settings, so I don't have to change them every time?

Yes, all of PuTTY's settings can be saved in named session profiles. You can also change the default settings that are used for new sessions. See section 4.1.2 in the documentation for how to do this.

A.2.6 Does PuTTY support storing its settings in a disk file?

Not at present, although section 4.26 in the documentation gives a method of achieving the same effect.

A.2.7 Does PuTTY support full-screen mode, like a DOS box?

Yes; this is a new feature in version 0.52.

A.2.8 Does PuTTY have the ability to remember my password so I don't have to type it every time?

No, it doesn't.

Remembering your password is a bad plan for obvious security reasons: anyone who gains access to your machine while you're away from your desk can find out the remembered password, and use it, abuse it or change it.

In addition, it's not even *possible* for PuTTY to automatically send your password in a Telnet session, because Telnet doesn't give the client software any indication of which part of the login process is the password prompt. PuTTY would have to guess, by looking for words like 'password' in the session data; and if your login program is written in something other than English, this won't work.

In SSH, remembering your password would be possible in theory, but there doesn't seem to be much point since SSH supports public key authentication, which is more flexible and more secure. See chapter 8 in the documentation for a full discussion of public key authentication.

A.2.9 Is there an option to turn off the annoying host key prompts?

No, there isn't. And there won't be. Even if you write it yourself and send us the patch, we won't accept it.

Those annoying host key prompts are the *whole point* of SSH. Without them, all the cryptographic technology SSH uses to secure your session is doing nothing more than making an attacker's job slightly harder; instead of sitting between you and the server with a packet sniffer, the attacker must actually subvert a router and start modifying the packets going back and forth. But that's not all that much harder than just sniffing; and without host key checking, it will go completely undetected by client or server.

Host key checking is your guarantee that the encryption you put on your data at the client end is the *same* encryption taken off the data at the server end; it's your guarantee that it hasn't been removed and replaced somewhere on the way. Host key checking makes the attacker's job *astronomically* hard, compared to packet sniffing, and even compared to subverting a router. Instead of applying a little intelligence and keeping an eye on Bugtraq, the attacker must now perform a brute-force attack against at least one military-strength cipher. That insignificant host key prompt really does make *that* much difference.

If you're having a specific problem with host key checking - perhaps you want an automated batch job to make use of PSCP or Plink, and the interactive host key prompt is hanging the batch process - then the right way to fix it is to add the correct host key to the Registry in advance. That way, you retain the *important* feature of host key checking: the right key will be accepted and the wrong ones will not. Adding an option to turn host key checking off completely is the wrong solution and we will not do it.

If you have host keys available in the common `known_hosts` format, we have a script called `kh2reg.py` to convert them to a Windows .REG file, which can be installed ahead of time by double-clicking or using `REGEDIT`.

A.2.10 Will you write an SSH server for the PuTTY suite, to go with the client?

No. The only reason we might want to would be if we could easily re-use existing code and significantly cut down the effort. We don't believe this is the case; there just isn't enough common ground between an SSH client and server to make it worthwhile.

If someone else wants to use bits of PuTTY in the process of writing a Windows SSH server, they'd be perfectly welcome to of course, but I really can't see it being a lot less effort for us to do that than it would be for us to write a server from the ground up. We don't have time, and we don't have motivation. The code is available if anyone else wants to try it.

A.2.11 Can PSCP or PSFTP transfer files in ASCII mode?

Unfortunately not.

Until recently, this was a limitation of the file transfer protocols: the SCP and SFTP protocols had no notion of transferring a file in anything other than binary mode. (This is still true of SCP.)

The current draft protocol spec of SFTP proposes a means of implementing ASCII transfer. At some point PSCP/PSFTP may implement this proposal.

A.3 Ports to other operating systems

The eventual goal is for PuTTY to be a multi-platform program, able to run on at least Windows, Mac OS and Unix.

Porting will become easier once PuTTY has a generalised porting layer, drawing a clear line between platform-dependent and platform-independent code. The general intention was for this porting layer to evolve naturally as part of the process of doing the first port; a Unix port has now been released and the plan seems to be working so far.

A.3.1 What ports of PuTTY exist?

Currently, release versions of PuTTY tools only run on full Win32 systems and Unix. ‘Win32’ includes Windows 95, 98, and ME, and it includes Windows NT, Windows 2000 and Windows XP.

In the development code, a partial port to the Mac OS (see question A.3.6) is under way.

Currently PuTTY does *not* run on Windows CE (see question A.3.4), and it does not quite run on the Win32s environment under Windows 3.1 (see question A.3.5).

We do not have release-quality ports for any other systems at the present time. If anyone told you we had an EPOC port, or an iPaq port, or any other port of PuTTY, they were mistaken. We don't.

There are some third-party ports to various platforms, mentioned on the Links page of our website.

A.3.2 Is there a port to Unix?

As of 0.54, there are Unix ports of most of the traditional PuTTY tools, and also one entirely new application.

If you look at the source release, you should find a `unix` subdirectory containing `Makefile.gtk`, which should build you Unix ports of Plink, PuTTY itself, PuTTYgen, PSCP, PSFTP, and also `pterm` - an `xterm`-type program which supports the same terminal emulation as PuTTY. We do not yet have a Unix port of Pageant.

If you don't have Gtk, you should still be able to build the command-line tools.

Note that Unix PuTTY has mostly only been tested on Linux so far; portability problems such as BSD-style ptys or different header file requirements are expected.

A.3.3 What's the point of the Unix port? Unix has OpenSSH.

All sorts of little things. `pterm` is directly useful to anyone who prefers PuTTY's terminal emulation to `xterm`'s, which at least some people do. Unix Plink has apparently found a niche among people who find the complexity of OpenSSL makes OpenSSH hard to install (and who don't mind Plink not having as many features). Some users want to generate a large number of SSH keys on Unix and then copy them all into PuTTY, and the Unix PuTTYgen should allow them to automate that conversion process.

There were development advantages as well; porting PuTTY to Unix was a valuable path-finding effort for other future ports, and also allowed us to use the excellent Linux tool Valgrind to help with debugging, which has already improved PuTTY's stability on *all* platforms.

However, if you're a Unix user and you can see no reason to switch from OpenSSH to PuTTY/Plink, then you're probably right. We don't expect our Unix port to be the right thing for everybody.

A.3.4 Will there be a port to Windows CE or PocketPC?

We have done some work on such a port, but it only reached an early stage, and certainly not a useful one. It's no longer being actively worked on.

However, there's a third-party port at `http://www.pocketputty.net/`.

A.3.5 Is there a port to Windows 3.1?

PuTTY is a 32-bit application from the ground up, so it won't run on Windows 3.1 as a native 16-bit program; and it would be *very* hard to port it to do so, because of Windows 3.1's vile memory allocation mechanisms.

However, it is possible in theory to compile the existing PuTTY source in such a way that it will run under Win32s (an extension to Windows 3.1 to let you run 32-bit programs). In order to do this you'll need the right kind of C compiler - modern versions of Visual C at least have stopped being backwards compatible to Win32s. Also, the last time we tried this it didn't work very well.

If you're interested in running PuTTY under Windows 3.1, help and testing in this area would be very welcome!

A.3.6 Will there be a port to the Mac?

There are several answers to this question:

- The Unix/Gtk port is already fully working under Mac OS X as an X11 application.
- A native (Cocoa) Mac OS X port has been started. It's just about usable, but is of nowhere near release quality yet, and is likely to behave in unexpected ways. Currently it's unlikely to be completed unless someone steps in to help.
- A separate port to the classic Mac OS (pre-OSX) is also in progress; it too is not ready yet.

A.3.7 Will there be a port to EPOC?

I hope so, but given that ports aren't really progressing very fast even on systems the developers *do* already know how to program for, it might be a long time before any of us get round to learning a new system and doing the port for that.

However, some of the work has been done by other people, and a beta port of PuTTY for the Nokia 9200 Communicator series is available from `http://s2putty.sourceforge.net/`

A.4 Embedding PuTTY in other programs

A.4.1 Is the SSH or Telnet code available as a DLL?

No, it isn't. It would take a reasonable amount of rewriting for this to be possible, and since the PuTTY project itself doesn't believe in DLLs (they make installation more error-prone) none of us has taken the time to do it.

Most of the code cleanup work would be a good thing to happen in general, so if anyone feels like helping, we wouldn't say no.

A.4.2 Is the SSH or Telnet code available as a Visual Basic component?

No, it isn't. None of the PuTTY team uses Visual Basic, and none of us has any particular need to make SSH connections from a Visual Basic application. In addition, all the preliminary work to turn it into a DLL would be necessary first; and furthermore, we don't even know how to write VB components.

If someone offers to do some of this work for us, we might consider it, but unless that happens I can't see VB integration being anywhere other than the very bottom of our priority list.

A.4.3 How can I use PuTTY to make an SSH connection from within another program?

Probably your best bet is to use Plink, the command-line connection tool. If you can start Plink as a second Windows process, and arrange for your primary process to be able to send data to the Plink process, and receive data from it, through pipes, then you should be able to make SSH connections from your program.

This is what CVS for Windows does, for example.

A.5 Details of PuTTY's operation

A.5.1 What terminal type does PuTTY use?

For most purposes, PuTTY can be considered to be an `xterm` terminal.

PuTTY also supports some terminal control sequences not supported by the real `xterm`: notably the Linux console sequences that reconfigure the colour palette, and the title bar control sequences used by `DECterm` (which are different from the `xterm` ones; PuTTY supports both).

By default, PuTTY announces its terminal type to the server as `xterm`. If you have a problem with this, you can reconfigure it to say something else; `vt220` might help if you have trouble.

A.5.2 Where does PuTTY store its data?

On Windows, PuTTY stores most of its data (saved sessions, SSH host keys) in the Registry. The precise location is

```
HKEY_CURRENT_USER\Software\SimonTatham\PuTTY
```

and within that area, saved sessions are stored under `Sessions` while host keys are stored under `SshHostKeys`.

PuTTY also requires a random number seed file, to improve the unpredictability of randomly chosen data needed as part of the SSH cryptography. This is stored by default in a file called `PUTTY.RND` in your Windows home directory (`%HOMEDRIVE%\%HOMEPATH%`), or in the actual Windows directory (such as `C:\WINDOWS`) if the home directory doesn't exist, for example if you're using Win95. If you want to change the location of the random number seed file, you can put your chosen pathname in the Registry, at

```
HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\RandSeedFile
```

You can ask PuTTY to delete all this data; see question A.8.2.

On Unix, PuTTY stores all of this data in a directory `~/.putty`.

A.6 HOWTO questions

A.6.1 What login name / password should I use?

This is not a question you should be asking *us*.

PuTTY is a communications tool, for making connections to other computers. We maintain the tool; we *don't* administer any computers that you're likely to be able to use, in the same way that the people who make web browsers aren't responsible for most of the content you can view in them. We cannot help with questions of this sort.

If you know the name of the computer you want to connect to, but don't know what login name or password to use, you should talk to whoever administers that computer. If you don't know who that is, see the next question for some possible ways to find out.

A.6.2 What commands can I type into my PuTTY terminal window?

Again, this is not a question you should be asking *us*. You need to read the manuals, or ask the administrator, of *the computer you have connected to*.

PuTTY does not process the commands you type into it. It's only a communications tool. It makes a connection to another computer; it passes the commands you type to that other computer; and it passes the other computer's responses back to you. Therefore, the precise range of commands you can use will not depend on PuTTY, but on what kind of computer you have connected to and what software is running on it. The PuTTY team cannot help you with that.

(Think of PuTTY as being a bit like a telephone. If you phone somebody up and you don't know what language to speak to make them understand you, it isn't *the telephone company*'s job to find that out for you. We just provide the means for you to get in touch; making yourself understood is somebody else's problem.)

If you are unsure of where to start looking for the administrator of your server, a good place to start might be to remember how you found out the host name in the PuTTY configuration. If you were given that host name by e-mail, for example, you could try asking the person who sent you that e-mail. If your company's IT department provided you with ready-made PuTTY saved sessions, then that IT department can probably also tell you something about what commands you can type during those sessions. But the PuTTY maintainer team does not administer any server you are likely to be connecting to, and cannot help you with questions of this type.

A.6.3 How can I make PuTTY start up maximised?

Create a Windows shortcut to start PuTTY from, and set it as 'Run Maximized'.

A.6.4 How can I create a Windows shortcut to start a particular saved session directly?

To run a PuTTY session saved under the name '`mysession`', create a Windows shortcut that invokes PuTTY with a command line like

```
\path\name\to\putty.exe -load "mysession"
```

(Note: prior to 0.53, the syntax was `@session`. This is now deprecated and may be removed at some point.)

A.6.5 How can I start an SSH session straight from the command line?

Use the command line `putty -ssh host.name`. Alternatively, create a saved session that specifies the SSH protocol, and start the saved session as shown in question A.6.4.

A.6.6 How do I copy and paste between PuTTY and other Windows applications?

Copy and paste works similarly to the X Window System. You use the left mouse button to select text in the PuTTY window. The act of selection *automatically* copies the text to the clipboard: there is no need to press Ctrl-Ins or Ctrl-C or anything else. In fact, pressing Ctrl-C will send a Ctrl-C character to the other end of your connection (just like it does the rest of the time), which may have unpleasant effects. The *only* thing you need to do, to copy text to the clipboard, is to select it.

To paste the clipboard contents into a PuTTY window, by default you click the right mouse button. If you have a three-button mouse and are used to X applications, you can configure pasting to be done by the middle button instead, but this is not the default because most Windows users don't have a middle button at all.

You can also paste by pressing Shift-Ins.

A.6.7 How do I use all PuTTY's features (public keys, proxying, cipher selection, etc.) in PSCP, PSFTP and Plink?

Most major features (e.g., public keys, port forwarding) are available through command line options. See the documentation.

Not all features are accessible from the command line yet, although we'd like to fix this. In the meantime, you can use most of PuTTY's features if you create a PuTTY saved session, and then use the name of the saved session on the command line in place of a hostname. This works for PSCP, PSFTP and Plink (but don't expect port forwarding in the file transfer applications!).

A.6.8 How do I use PSCP.EXE? When I double-click it gives me a command prompt window which then closes instantly.

PSCP is a command-line application, not a GUI application. If you run it without arguments, it will simply print a help message and terminate.

To use PSCP properly, run it from a Command Prompt window. See chapter 5 in the documentation for more details.

A.6.9 How do I use PSCP to copy a file whose name has spaces in?

If PSCP is using the traditional SCP protocol, this is confusing. If you're specifying a file at the local end, you just use one set of quotes as you would normally do:

```
pscp "local filename with spaces" user@host:
pscp user@host:myfile "local filename with spaces"
```

But if the filename you're specifying is on the *remote* side, you have to use backslashes and two sets of quotes:

```
pscp user@host:"\"remote filename with spaces\"" local_filename
pscp local_filename user@host:"\"remote filename with spaces\""
```

Worse still, in a remote-to-local copy you have to specify the local file name explicitly, otherwise PSCP will complain that they don't match (unless you specified the `-unsafe` option). The following command will give an error message:

```
c:\>pscp user@host:"\"oo er\"" .
warning: remote host tried to write to a file called 'oo er'
         when we requested a file called '"oo er"'.
```

Instead, you need to specify the local file name in full:

```
c:\>pscp user@host:"\"oo er\"" "oo er"
```

If PSCP is using the newer SFTP protocol, none of this is a problem, and all filenames with spaces in are specified using a single pair of quotes in the obvious way:

```
pscp "local file" user@host:
pscp user@host:"remote file" .
```

A.7 Troubleshooting

A.7.1 Why do I see ‘Incorrect MAC received on packet’?

One possible cause of this that used to be common is a bug in old SSH-2 servers distributed by `ssh.com`. (This is not the only possible cause; see section 10.11 in the documentation.) Version 2.3.0 and below of their SSH-2 server constructs Message Authentication Codes in the wrong way, and expects the client to construct them in the same wrong way. PuTTY constructs the MACs correctly by default, and hence these old servers will fail to work with it.

If you are using PuTTY version 0.52 or better, this should work automatically: PuTTY should detect the buggy servers from their version number announcement, and automatically start to construct its MACs in the same incorrect manner as they do, so it will be able to work with them.

If you are using PuTTY version 0.51 or below, you can enable the workaround by going to the SSH panel and ticking the box labelled ‘Imitate SSH2 MAC bug’. It's possible that you might have to do this with 0.52 as well, if a buggy server exists that PuTTY doesn't know about.

In this context MAC stands for Message Authentication Code. It's a cryptographic term, and it has nothing at all to do with Ethernet MAC (Media Access Control) addresses.

A.7.2 Why do I see ‘Fatal: Protocol error: Expected control record’ in PSCP?

This happens because PSCP was expecting to see data from the server that was part of the PSCP protocol exchange, and instead it saw data that it couldn't make any sense of at all.

This almost always happens because the startup scripts in your account on the server machine are generating output. This is impossible for PSCP, or any other SCP client, to work around. You should never use startup files (`.bashrc`, `.cshrc` and so on) which generate output in non-interactive sessions.

This is not actually a PuTTY problem. If PSCP fails in this way, then all other SCP clients are likely to fail in exactly the same way. The problem is at the server end.

A.7.3 I clicked on a colour in the Colours panel, and the colour didn't change in my terminal.

That isn't how you're supposed to use the Colours panel.

During the course of a session, PuTTY potentially uses *all* the colours listed in the Colours panel. It's not a question of using only one of them and you choosing which one; PuTTY will use them *all*. The purpose of the Colours panel is to let you adjust the appearance of all the colours. So to change the colour of the cursor, for example, you would select ‘Cursor Colour’, press the ‘Modify’ button, and select a new colour from the dialog box that appeared. Similarly, if you want your session to appear in green, you should select ‘Default Foreground’ and press ‘Modify’. Clicking on ‘ANSI Green’ won't turn your session green; it will only allow you to adjust the *shade* of green used when PuTTY is instructed by the server to display green text.

A.7.4 Plink on Windows 95 says it can't find `WS2_32.DLL`.

Plink requires the extended Windows network library, WinSock version 2. This is installed as standard on Windows 98 and above, and on Windows NT, and even on later versions of Windows 95; but early Win95 installations don't have it.

In order to use Plink on these systems, you will need to download the WinSock 2 upgrade:

```
http://www.microsoft.com/windows95/downloads/contents/
  wuadmintools/s_wunetworkingtools/w95sockets2/
```

A.7.5 After trying to establish an SSH-2 connection, PuTTY says ‘Out of memory’ and dies.

If this happens just while the connection is starting up, this often indicates that for some reason the client and server have failed to establish a session encryption key. Somehow, they have performed calculations that should have given each of them the same key, but have ended up with different keys; so data encrypted by one and decrypted by the other looks like random garbage.

This causes an ‘out of memory’ error because the first encrypted data PuTTY expects to see is the length of an SSH message. Normally this will be something well under 100 bytes. If the decryption has failed, PuTTY will see a completely random length in the region of two *gigabytes*, and will try to allocate enough memory to store this non-existent message. This will immediately lead to it thinking it doesn't have enough memory, and panicking.

If this happens to you, it is quite likely to still be a PuTTY bug and you should report it (although it might be a bug in your SSH server instead); but it doesn't necessarily mean you've actually run out of memory.

A.7.6 When attempting a file transfer, either PSCP or PSFTP says 'Out of memory' and dies.

This is almost always caused by your login scripts on the server generating output. PSCP or PSFTP will receive that output when they were expecting to see the start of a file transfer protocol, and they will attempt to interpret the output as file-transfer protocol. This will usually lead to an 'out of memory' error for much the same reasons as given in question A.7.5.

This is a setup problem in your account on your server, *not* a PSCP/PSFTP bug. Your login scripts should *never* generate output during non-interactive sessions; secure file transfer is not the only form of remote access that will break if they do.

On Unix, a simple fix is to ensure that all the parts of your login script that might generate output are in `.profile` (if you use a Bourne shell derivative) or `.login` (if you use a C shell). Putting them in more general files such as `.bashrc` or `.cshrc` is liable to lead to problems.

A.7.7 PSFTP transfers files much slower than PSCP.

The throughput of PSFTP 0.54 should be much better than 0.53b and prior; we've added code to the SFTP backend to queue several blocks of data rather than waiting for an acknowledgement for each. (The SCP backend did not suffer from this performance issue because SCP is a much simpler protocol.)

A.7.8 When I run full-colour applications, I see areas of black space where colour ought to be, or vice versa.

You almost certainly need to change the 'Use background colour to erase screen' setting in the Terminal panel. If there is too much black space (the commoner situation), you should enable it, while if there is too much colour, you should disable it. (See section 4.3.4.)

In old versions of PuTTY, this was disabled by default, and would not take effect until you reset the terminal (see question A.7.9). Since 0.54, it is enabled by default, and changes take effect immediately.

A.7.9 When I change some terminal settings, nothing happens.

Some of the terminal options (notably Auto Wrap and background-colour screen erase) actually represent the *default* setting, rather than the currently active setting. The server can send sequences that modify these options in mid-session, but when the terminal is reset (by server action, or by you choosing 'Reset Terminal' from the System menu) the defaults are restored.

In versions 0.53b and prior, if you change one of these options in the middle of a session, you will find that the change does not immediately take effect. It will only take effect once you reset the terminal.

In version 0.54, the behaviour has changed - changes to these settings take effect immediately.

A.7.10 My PuTTY sessions unexpectedly close after they are idle for a while.

Some types of firewall, and almost any router doing Network Address Translation (NAT, also known as IP masquerading), will forget about a connection through them if the connection does nothing for too long. This will cause the connection to be rudely cut off when contact is resumed.

You can try to combat this by telling PuTTY to send *keepalives*: packets of data which have no effect on the actual session, but which reassure the router or firewall that the network connection is still active and worth remembering about.

Keepalives don't solve everything, unfortunately; although they cause greater robustness against this sort of router, they can also cause a *loss* of robustness against network dropouts. See section 4.13.1 in the documentation for more discussion of this.

A.7.11 PuTTY's network connections time out too quickly when network connectivity is temporarily lost.

This is a Windows problem, not a PuTTY problem. The timeout value can't be set on per application or per session basis. To increase the TCP timeout globally, you need to tinker with the Registry.

On Windows 95, 98 or ME, the registry key you need to create or change is

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\
  MSTCP\MaxDataRetries
```

(it must be of type DWORD in Win95, or String in Win98/ME). (See MS Knowledge Base article 158474 for more information.)

On Windows NT, 2000, or XP, the registry key to create or change is

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\
  Parameters\TcpMaxDataRetransmissions
```

and it must be of type DWORD. (See MS Knowledge Base articles 120642 and 314053 for more information.)

Set the key's value to something like 10. This will cause Windows to try harder to keep connections alive instead of abandoning them.

A.7.12 When I `cat` a binary file, I get 'PuTTYPuTTYPuTTY' on my command line.

Don't do that, then.

This is designed behaviour; when PuTTY receives the character Control-E from the remote server, it interprets it as a request to identify itself, and so it sends back the string '`PuTTY`' as if that string had been entered at the keyboard. Control-E should only be sent by programs that are prepared to deal with the response. Writing a binary file to your terminal is likely to output many Control-E characters, and cause this behaviour. Don't do it. It's a bad plan.

To mitigate the effects, you could configure the answerback string to be empty (see section 4.3.6); but writing binary files to your terminal is likely to cause various other unpleasant behaviour, so this is only a small remedy.

A.7.13 When I `cat` a binary file, my window title changes to a nonsense string.

Don't do that, then.

It is designed behaviour that PuTTY should have the ability to adjust the window title on instructions from the server. Normally the control sequence that does this should only be sent deliberately, by programs that know what they are doing and intend to put meaningful text in the window title. Writing a binary file to your terminal runs the risk of sending the same control sequence by accident, and cause unexpected changes in the window title. Don't do it.

A.7.14 My keyboard stops working once PuTTY displays the password prompt.

No, it doesn't. PuTTY just doesn't display the password you type, so that someone looking at your screen can't see what it is.

Unlike the Windows login prompts, PuTTY doesn't display the password as a row of asterisks either. This is so that someone looking at your screen can't even tell how *long* your password is, which might be valuable information.

A.7.15 One or more function keys don't do what I expected in a server-side application.

If you've already tried all the relevant options in the PuTTY Keyboard panel, you may need to mail the PuTTY maintainers and ask.

It is *not* usually helpful just to tell us which application, which server operating system, and which key isn't working; in order to replicate the problem we would need to have a copy of every operating system, and every application, that anyone has ever complained about.

PuTTY responds to function key presses by sending a sequence of control characters to the server. If a function key isn't doing what you expect, it's likely that the character sequence your application is expecting to receive is not the same as the one PuTTY is sending. Therefore what we really need to know is *what* sequence the application is expecting.

The simplest way to investigate this is to find some other terminal environment, in which that function key *does* work; and then investigate what sequence the function key is sending in that situation. One reasonably easy way to do this on a Unix system is to type the command `cat`, and then press the function key. This is likely to produce output of the form `^[[11~`. You can also do this in PuTTY, to find out what sequence the function key is producing in that. Then you can mail the PuTTY maintainers and tell us ‘I wanted the F1 key to send `^[[11~`, but instead it's sending `^[OP`, can this be done?’, or something similar.

You should still read the Feedback page on the PuTTY website (also provided as appendix B in the manual), and follow the guidelines contained in that.

A.7.16 Since my SSH server was upgraded to OpenSSH 3.1p1/3.4p1, I can no longer connect with PuTTY.

There is a known problem when OpenSSH has been built against an incorrect version of OpenSSL; the quick workaround is to configure PuTTY to use SSH protocol 2 and the Blowfish cipher.

For more details and OpenSSH patches, see bug 138 in the OpenSSH BTS.

This is not a PuTTY-specific problem; if you try to connect with another client you'll likely have similar problems. (Although PuTTY's default cipher differs from many other clients.)

OpenSSH 3.1p1: configurations known to be broken (and symptoms):

- SSH-2 with AES cipher (PuTTY says ‘Assertion failed! Expression: (len & 15) == 0’ in `sshaes.c`, or ‘Out of memory’, or crashes)
- SSH-2 with 3DES (PuTTY says ‘Incorrect MAC received on packet’)
- SSH-1 with Blowfish (PuTTY says ‘Incorrect CRC received on packet’)
- SSH-1 with 3DES

OpenSSH 3.4p1: as of 3.4p1, only the problem with SSH-1 and Blowfish remains. Rebuild your server, apply the patch linked to from bug 138 above, or use another cipher (e.g., 3DES) instead.

Other versions: we occasionally get reports of the same symptom and workarounds with older versions of OpenSSH, although it's not clear the underlying cause is the same.

A.7.17 Why do I see 'Couldn't load private key from ...'? Why can PuTTYgen load my key but not PuTTY?

It's likely that you've generated an SSH protocol 2 key with PuTTYgen, but you're trying to use it in an SSH-1 connection. SSH-1 and SSH-2 keys have different formats, and (at least in 0.52) PuTTY's reporting of a key in the wrong format isn't optimal.

To connect using SSH-2 to a server that supports both versions, you need to change the configuration from the default (see question A.2.1).

A.7.18 When I'm connected to a Red Hat Linux 8.0 system, some characters don't display properly.

A common complaint is that hyphens in man pages show up as a-acute.

With release 8.0, Red Hat appear to have made UTF-8 the default character set. There appears to be no way for terminal emulators such as PuTTY to know this (as far as we know, the appropriate escape sequence to switch into UTF-8 mode isn't sent).

A fix is to configure sessions to RH8 systems to use UTF-8 translation - see section 4.10.1 in the documentation. (Note that if you use 'Change Settings', changes may not take place immediately - see question A.7.9.)

If you really want to change the character set used by the server, the right place is `/etc/sysconfig/i18n`, but this shouldn't be necessary.

A.7.19 Since I upgraded to PuTTY 0.54, the scrollback has stopped working when I run `screen`.

PuTTY's terminal emulator has always had the policy that when the 'alternate screen' is in use, nothing is added to the scrollback. This is because the usual sorts of programs which use the alternate screen are things like text editors, which tend to scroll back and forth in the same document a lot; so (a) they would fill up the scrollback with a large amount of unhelpfully disordered text, and (b) they contain their *own* method for the user to scroll back to the bit they were interested in. We have generally found this policy to do the Right Thing in almost all situations.

Unfortunately, `screen` is one exception: it uses the alternate screen, but it's still usually helpful to have PuTTY's scrollback continue working. The simplest solution is to go to the Features control panel and tick 'Disable switching to alternate terminal screen'. (See section 4.6.4 for more details.) Alternatively, you can tell `screen` itself not to use the alternate screen: the `screen` FAQ suggests adding the line '`termcapinfo xterm ti@:te@`' to your `.screenrc` file.

The reason why this only started to be a problem in 0.54 is because `screen` typically uses an unusual control sequence to switch to the alternate screen, and previous versions of PuTTY did not support this sequence.

A.7.20 Since I upgraded Windows XP to Service Pack 2, I can't use addresses like `127.0.0.2`.

Some people who ask PuTTY to listen on localhost addresses other than `127.0.0.1` to forward services such as SMB and Windows Terminal Services have found that doing so no longer works since they upgraded to WinXP SP2.

This is apparently an issue with SP2 that is acknowledged by Microsoft in MS Knowledge Base article 884020. The article links to a fix you can download.

(*However*, we've been told that SP2 *also* fixes the bug that means you need to use non-`127.0.0.1` addresses to forward Terminal Services in the first place.)

A.7.21 PSFTP commands seem to be missing a directory separator (slash).

Some people have reported the following incorrect behaviour with PSFTP:

```
psftp> pwd
Remote directory is /dir1/dir2
psftp> get filename.ext
/dir1/dir2filename.ext: no such file or directory
```

This is not a bug in PSFTP. There is a known bug in some versions of portable OpenSSH (bug 697) that causes these symptoms; it appears to have been introduced around 3.7.x. It manifests only on certain platforms (AIX is what has been reported to us).

There is a patch for OpenSSH attached to that bug; it's also fixed in recent versions of portable OpenSSH (from around 3.8).

A.7.22 Do you want to hear about ‘Software caused connection abort’?

In the documentation for PuTTY 0.53 and 0.53b, we mentioned that we'd like to hear about any occurrences of this error. Since the release of PuTTY 0.54, however, we've been convinced that this error doesn't indicate that PuTTY's doing anything wrong, and we don't need to hear about further occurrences. See section 10.14 for our current documentation of this error.

A.7.23 My SSH-2 session locks up for a few seconds every so often.

Recent versions of PuTTY automatically initiate repeat key exchange once per hour, to improve session security. If your client or server machine is slow, you may experience this as a delay of anything up to thirty seconds or so.

These delays are inconvenient, but they are there for your protection. If they really cause you a problem, you can choose to turn off periodic rekeying using the ‘Kex’ configuration panel (see section 4.19), but be aware that you will be sacrificing security for this. (Falling back to SSH-1 would also remove the delays, but would lose a *lot* more security still. We do not recommend it.)

A.7.24 PuTTY fails to start up. Windows claims that ‘the application configuration is incorrect’.

This is caused by a bug in certain versions of Windows XP which is triggered by PuTTY 0.58. This was fixed in 0.59. The ‘xp-wont-run’ entry in PuTTY's wishlist has more details.

A.8 Security questions

A.8.1 Is it safe for me to download PuTTY and use it on a public PC?

It depends on whether you trust that PC. If you don't trust the public PC, don't use PuTTY on it, and don't use any other software you plan to type passwords into either. It might be watching your keystrokes, or it might

tamper with the PuTTY binary you download. There is *no* program safe enough that you can run it on an actively malicious PC and get away with typing passwords into it.

If you do trust the PC, then it's probably OK to use PuTTY on it (but if you don't trust the network, then the PuTTY download might be tampered with, so it would be better to carry PuTTY with you on a floppy).

A.8.2 What does PuTTY leave on a system? How can I clean up after it?

PuTTY will leave some Registry entries, and a random seed file, on the PC (see question A.5.2). If you are using PuTTY on a public PC, or somebody else's PC, you might want to clean these up when you leave. You can do that automatically, by running the command `putty -cleanup`. (Note that this only removes settings for the currently logged-in user on multi-user systems.)

If PuTTY was installed from the installer package, it will also appear in 'Add/Remove Programs'. Older versions of the uninstaller do not remove the above-mentioned registry entries and file.

A.8.3 How come PuTTY now supports DSA, when the website used to say how insecure it was?

DSA has a major weakness *if badly implemented*: it relies on a random number generator to far too great an extent. If the random number generator produces a number an attacker can predict, the DSA private key is exposed - meaning that the attacker can log in as you on all systems that accept that key.

The PuTTY policy changed because the developers were informed of ways to implement DSA which do not suffer nearly as badly from this weakness, and indeed which don't need to rely on random numbers at all. For this reason we now believe PuTTY's DSA implementation is probably OK. However, if you have the choice, we still recommend you use RSA instead.

A.8.4 Couldn't Pageant use `VirtualLock()` to stop private keys being written to disk?

Unfortunately not. The `VirtualLock()` function in the Windows API doesn't do a proper job: it may prevent small pieces of a process's memory from being paged to disk while the process is running, but it doesn't stop the process's memory as a whole from being swapped completely out to disk when the process is long-term inactive. And Pageant spends most of its time inactive.

A.9 Administrative questions

A.9.1 Would you like me to register you a nicer domain name?

No, thank you. Even if you can find one (most of them seem to have been registered already, by people who didn't ask whether we actually wanted it before they applied), we're happy with the PuTTY web site being exactly where it is. It's not hard to find (just type 'putty' into google.com and we're the first link returned), and we don't believe the administrative hassle of moving the site would be worth the benefit.

In addition, if we *did* want a custom domain name, we would want to run it ourselves, so we knew for certain that it would continue to point where we wanted it, and wouldn't suddenly change or do strange things. Having it registered for us by a third party who we don't even know is not the best way to achieve this.

A.9.2 Would you like free web hosting for the PuTTY web site?

We already have some, thanks.

A.9.3 Would you link to my web site from the PuTTY web site?

Only if the content of your web page is of definite direct interest to PuTTY users. If your content is unrelated, or only tangentially related, to PuTTY, then the link would simply be advertising for you.

One very nice effect of the Google ranking mechanism is that by and large, the most popular web sites get the highest rankings. This means that when an ordinary person does a search, the top item in the search is very likely to be a high-quality site or the site they actually wanted, rather than the site which paid the most money for its ranking.

The PuTTY web site is held in high esteem by Google, for precisely this reason: lots of people have linked to it simply because they like PuTTY, without us ever having to ask anyone to link to us. We feel that it would be an abuse of this esteem to use it to boost the ranking of random advertisers' web sites. If you want your web site to have a high Google ranking, we'd prefer that you achieve this the way we did - by being good enough at what you do that people will link to you simply because they like you.

In particular, we aren't interested in trading links for money (see above), and we *certainly* aren't interested in trading links for other links (since we have no advertising on our web site, our Google ranking is not even directly worth anything to us). If we don't want to link to you for free, then we probably won't want to link to you at all.

If you have software based on PuTTY, or specifically designed to interoperate with PuTTY, or in some other way of genuine interest to PuTTY users, then we will probably be happy to add a link to you on our Links page. And if you're running a mirror of the PuTTY web site, we're *definitely* interested.

A.9.4 Why don't you move PuTTY to SourceForge?

Partly, because we don't want to move the web site location (see question A.9.1).

Also, security reasons. PuTTY is a security product, and as such it is particularly important to guard the code and the web site against unauthorised modifications which might introduce subtle security flaws. Therefore, we prefer that the Subversion repository, web site and FTP site remain where they are, under the direct control of system administrators we know and trust personally, rather than being run by a large organisation full of people we've never met and which is known to have had breakins in the past.

No offence to SourceForge; I think they do a wonderful job. But they're not ideal for everyone, and in particular they're not ideal for us.

A.9.5 Why can't I subscribe to the putty-bugs mailing list?

Because you're not a member of the PuTTY core development team. The putty-bugs mailing list is not a general newsgroup-like discussion forum; it's a contact address for the core developers, and an *internal* mailing list for us to discuss things among ourselves. If we opened it up for everybody to subscribe to, it would turn into something more like a newsgroup and we would be completely overwhelmed by the volume of traffic. It's hard enough to keep up with the list as it is.

A.9.6 If putty-bugs isn't a general-subscription mailing list, what is?

There isn't one, that we know of.

If someone else wants to set up a mailing list or other forum for PuTTY users to help each other with common problems, that would be fine with us, though the PuTTY team would almost certainly not have the time to read it. It's probably better to use one of the established newsgroups for this purpose (see section B.1.2).

A.9.7 How can I donate to PuTTY development?

Please, *please* don't feel you have to. PuTTY is completely free software, and not shareware. We think it's very important that *everybody* who wants to use PuTTY should be able to, whether they have any money or not; so the last thing we would want is for a PuTTY user to feel guilty because they haven't paid us any money. If you want to keep your money, please do keep it. We wouldn't dream of asking for any.

Having said all that, if you still really *want* to give us money, we won't argue :-) The easiest way for us to accept donations is if you send money to <anakin@pobox.com> using PayPal (www.paypal.com). Alternatively, if you don't trust PayPal, you could donate through e-gold (www.e-gold.com): deposit your donation in account number 174769, then send us e-mail to let us know you've done so (otherwise we might not notice for months!).

Small donations (tens of dollars or tens of euros) will probably be spent on beer or curry, which helps motivate our volunteer team to continue doing this for the world. Larger donations will be spent on something that actually helps development, if we can find anything (perhaps new hardware, or a copy of Windows XP), but if we can't find anything then we'll just distribute the money among the developers. If you want to be sure your donation is going towards something worthwhile, ask us first. If you don't like these terms, feel perfectly free not to donate. We don't mind.

A.9.8 Can I have permission to put PuTTY on a cover disk / distribute it with other software / etc?

Yes. For most things, you need not bother asking us explicitly for permission; our licence already grants you permission.

See section B.7 for more details.

A.9.9 Can you sign an agreement indemnifying us against security problems in PuTTY?

No!

A vendor of physical security products (e.g. locks) might plausibly be willing to accept financial liability for a product that failed to perform as advertised and resulted in damage (e.g. valuables being stolen). The reason they can afford to do this is because they sell a *lot* of units, and only a small proportion of them will fail; so they can meet their financial liability out of the income from all the rest of their sales, and still have enough left over to make a profit. Financial liability is intrinsically linked to selling your product for money.

There are two reasons why PuTTY is not analogous to a physical lock in this context. One is that software products don't exhibit random variation: *if* PuTTY has a security hole (which does happen, although we do our utmost to prevent it and to respond quickly when it does), every copy of PuTTY will have the same hole, so it's likely to affect all the users at the same time. So even if our users were all paying us to use PuTTY, we wouldn't be able to *simultaneously* pay every affected user compensation in excess of the amount they had paid us in the first place. It just wouldn't work.

The second, much more important, reason is that PuTTY users *don't* pay us. The PuTTY team does not have an income; it's a volunteer effort composed of people spending their spare time to try to write useful software. We aren't even a company or any kind of legally recognised organisation. We're just a bunch of people who happen to do some stuff in our spare time.

Therefore, to ask us to assume financial liability is to ask us to assume a risk of having to pay it out of our own *personal* pockets: out of the same budget from which we buy food and clothes and pay our rent. That's more than we're willing to give. We're already giving a lot of our spare *time* to developing software for free; if we had to pay our own *money* to do it as well, we'd start to wonder why we were bothering.

Free software fundamentally does not work on the basis of financial guarantees. Your guarantee of the software functioning correctly is simply that you have the source code and can check it before you use it. If you want to be sure there aren't any security holes, do a security audit of the PuTTY code, or hire a security engineer if you don't have the necessary skills yourself: instead of trying to ensure you can get compensation in the event of a disaster, try to ensure there isn't a disaster in the first place.

If you *really* want financial security, see if you can find a security engineer who will take financial responsibility for the correctness of their review. (This might be less likely to suffer from the everything-failing-at-once problem mentioned above, because such an engineer would probably be reviewing a lot of *different* products which would tend to fail independently.) Failing that, see if you can persuade an insurance company to insure you against security incidents, and if the insurer demands it as a condition then get our code reviewed by a security engineer they're happy with.

A.9.10 Can you sign this form granting us permission to use/distribute PuTTY?

If your form contains any clause along the lines of 'the undersigned represents and warrants', we're not going to sign it. This is particularly true if it asks us to warrant that PuTTY is secure; see question A.9.9 for more discussion of this. But it doesn't really matter what we're supposed to be warranting: even if it's something we already believe is true, such as that we don't infringe any third-party copyright, we will not sign a document accepting any legal or financial liability. This is simply because the PuTTY development project has no income out of which to satisfy that liability, or pay legal costs, should it become necessary. We cannot afford to be sued. We are assuring you that *we have done our best*; if that isn't good enough for you, tough.

The existing PuTTY licence document already gives you permission to use or distribute PuTTY in pretty much any way which does not involve pretending you wrote it or suing us if it goes wrong. We think that really ought to be enough for anybody.

See also question A.9.12 for another reason why we don't want to do this sort of thing.

A.9.11 Can you write us a formal notice of permission to use PuTTY?

We could, in principle, but it isn't clear what use it would be. If you think there's a serious chance of one of the PuTTY copyright holders suing you (which we don't!), you would presumably want a signed notice from *all* of them; and we couldn't provide that even if we wanted to, because many of the copyright holders are people who contributed some code in the past and with whom we subsequently lost contact. Therefore the best we would be able to do *even in theory* would be to have the core development team sign the document, which wouldn't guarantee you that some other copyright holder might not sue.

See also question A.9.12 for another reason why we don't want to do this sort of thing.

A.9.12 Can you sign *anything* for us?

Not unless there's an incredibly good reason.

We are generally unwilling to set a precedent that involves us having to enter into individual agreements with PuTTY users. We estimate that we have literally *millions* of users, and we absolutely would not have time to go round signing specific agreements with every one of them. So if you want us to sign something specific for you, you might usefully stop to consider whether there's anything special that distinguishes you from 999,999 other users, and therefore any reason we should be willing to sign something for you without it setting such a precedent.

If your company policy requires you to have an individual agreement with the supplier of any software you use, then your company policy is simply not well suited to using popular free software, and we urge you to consider this as a flaw in your policy.

A.9.13 If you won't sign anything, can you give us some sort of assurance that you won't make PuTTY closed-source in future?

Yes and no.

If what you want is an assurance that some *current version* of PuTTY which you've already downloaded will remain free, then you already have that assurance: it's called the PuTTY Licence. It grants you permission to use, distribute and copy the software to which it applies; once we've granted that permission (which we have), we can't just revoke it.

On the other hand, if you want an assurance that *future* versions of PuTTY won't be closed-source, that's more difficult. We could in principle sign a document stating that we would never release a closed-source PuTTY, but that wouldn't assure you that we *would* keep releasing *open*-source PuTTYs: we would still have the option of ceasing to develop PuTTY at all, which would surely be even worse for you than making it closed-source! (And we almost certainly wouldn't *want* to sign a document guaranteeing that we would actually continue to do development work on PuTTY; we certainly wouldn't sign it for free. Documents like that are called contracts of employment, and are generally not signed except in return for a sizeable salary.)

If we *were* to stop developing PuTTY, or to decide to make all future releases closed-source, then you would still be free to copy the last open release in accordance with the current licence, and in particular you could start your own fork of the project from that release. If this happened, I confidently predict that *somebody* would do that, and that some kind of a free PuTTY would continue to be developed. There's already precedent for that sort of thing happening in free software. We can't guarantee that somebody *other than you* would do it, of course; you might have to do it yourself. But we can assure you that there would be nothing *preventing* anyone from continuing free development if we stopped.

(Finally, we can also confidently predict that if we made PuTTY closed-source and someone made an open-source fork, most people would switch to the latter. Therefore, it would be pretty stupid of us to try it.)

A.9.14 Can you provide us with export control information / FIPS certification for PuTTY?

Some people have asked us for an Export Control Classification Number (ECCN) for PuTTY. We don't know whether we have one, and as a team of free software developers based in the UK we don't have the time, money, or effort to deal with US bureaucracy to investigate any further. We believe that PuTTY falls under 5D002 on the US Commerce Control List, but that shouldn't be taken as definitive. If you need to know more you should seek professional legal advice. The same applies to any other country's legal requirements and restrictions.

Similarly, some people have asked us for FIPS certification of the PuTTY tools. Unless someone else is prepared to do the necessary work and pay any costs, we can't provide this.

A.10 Miscellaneous questions

A.10.1 Is PuTTY a port of OpenSSH, or based on OpenSSH?

No, it isn't. PuTTY is almost completely composed of code written from scratch for PuTTY. The only code we share with OpenSSH is the detector for SSH-1 CRC compensation attacks, written by CORE SDI S.A.

A.10.2 Where can I buy silly putty?

You're looking at the wrong web site; the only PuTTY we know about here is the name of a computer program.

If you want the kind of putty you can buy as an executive toy, the PuTTY team can personally recommend Thinking Putty, which you can buy from Crazy Aaron's Putty World, at www.puttyworld.com.

A.10.3 What does ‘PuTTY’ mean?

It's the name of a popular SSH and Telnet client. Any other meaning is in the eye of the beholder. It's been rumoured that ‘PuTTY’ is the antonym of ‘getty’, or that it's the stuff that makes your Windows useful, or that it's a kind of plutonium Teletype. We couldn't possibly comment on such allegations.

A.10.4 How do I pronounce ‘PuTTY’?

Exactly like the English word ‘putty’, which we pronounce /ˈpʌti/.

If you want to provide feedback on this manual or on the PuTTY tools themselves, see the Feedback page.

[PuTTY release 0.60]