Previous | Contents | Index | Next

- Chapter 4: Configuring PuTTY
  - 4.1 The Session panel
    - 4.1.1 The host name section
    - 4.1.2 Loading and storing saved sessions
    - 4.1.3 'Close Window on Exit'
  - 4.2 The Logging panel
    - 4.2.1 'Log file name'
    - 4.2.2 'What to do if the log file already exists'
    - 4.2.3 'Flush log file frequently'
    - 4.2.4 Options specific to SSH packet logging
  - 4.3 The Terminal panel
    - 4.3.1 'Auto wrap mode initially on'
    - 4.3.2 'DEC Origin Mode initially on'
    - 4.3.3 'Implicit CR in every LF'
    - 4.3.4 'Use background colour to erase screen'
    - 4.3.5 'Enable blinking text'
    - 4.3.6 'Answerback to ^E'
    - 4.3.7 'Local echo'
    - 4.3.8 'Local line editing'
    - 4.3.9 Remote-controlled printing
  - 4.4 The Keyboard panel
    - 4.4.1 Changing the action of the Backspace key
    - 4.4.2 Changing the action of the Home and End keys
    - 4.4.3 Changing the action of the function keys and keypad
    - 4.4.4 Controlling Application Cursor Keys mode
    - 4.4.5 Controlling Application Keypad mode
    - 4.4.6 Using NetHack keypad mode
    - 4.4.7 Enabling a DEC-like Compose key
    - 4.4.8 'Control-Alt is different from AltGr'
  - 4.5 The Bell panel
    - 4.5.1 'Set the style of bell'
    - 4.5.2 'Taskbar/caption indication on bell'
    - 4.5.3 'Control the bell overload behaviour'
  - 4.6 The Features panel
    - 4.6.1 Disabling application keypad and cursor keys
    - 4.6.2 Disabling `xterm`-style mouse reporting
    - 4.6.3 Disabling remote terminal resizing
    - 4.6.4 Disabling switching to the alternate screen
    - 4.6.5 Disabling remote window title changing
    - 4.6.6 Response to remote window title querying
    - 4.6.7 Disabling destructive backspace
    - 4.6.8 Disabling remote character set configuration
    - 4.6.9 Disabling Arabic text shaping
    - 4.6.10 Disabling bidirectional text display
  - 4.7 The Window panel
    - 4.7.1 Setting the size of the PuTTY window
    - 4.7.2 What to do when the window is resized
    - 4.7.3 Controlling scrollback
    - 4.7.4 'Push erased text into scrollback'
  - 4.8 The Appearance panel
    - 4.8.1 Controlling the appearance of the cursor

        - 4.8.2 Controlling the font used in the terminal window
        - 4.8.3 ‘Hide mouse pointer when typing in window’
        - 4.8.4 Controlling the window border
    - 4.9 The Behaviour panel
        - 4.9.1 Controlling the window title
        - 4.9.2 ‘Warn before closing window’
        - 4.9.3 ‘Window closes on ALT-F4’
        - 4.9.4 ‘System menu appears on ALT-Space’
        - 4.9.5 ‘System menu appears on Alt alone’
        - 4.9.6 ‘Ensure window is always on top’
        - 4.9.7 ‘Full screen on Alt-Enter’
    - 4.10 The Translation panel
        - 4.10.1 Controlling character set translation
        - 4.10.2 ‘Treat CJK ambiguous characters as wide’
        - 4.10.3 ‘Caps Lock acts as Cyrillic switch’
        - 4.10.4 Controlling display of line-drawing characters
        - 4.10.5 Controlling copy and paste of line drawing characters
    - 4.11 The Selection panel
        - 4.11.1 Pasting in Rich Text Format
        - 4.11.2 Changing the actions of the mouse buttons
        - 4.11.3 ‘Shift overrides application's use of mouse’
        - 4.11.4 Default selection mode
        - 4.11.5 Configuring word-by-word selection
    - 4.12 The Colours panel
        - 4.12.1 ‘Allow terminal to specify ANSI colours’
        - 4.12.2 ‘Allow terminal to use xterm 256-colour mode’
        - 4.12.3 ‘Bolded text is a different colour’
        - 4.12.4 ‘Attempt to use logical palettes’
        - 4.12.5 ‘Use system colours’
        - 4.12.6 Adjusting the colours in the terminal window
    - 4.13 The Connection panel
        - 4.13.1 Using keepalives to prevent disconnection
        - 4.13.2 ‘Disable Nagle's algorithm’
        - 4.13.3 ‘Enable TCP keepalives’
        - 4.13.4 ‘Internet protocol’
    - 4.14 The Data panel
        - 4.14.1 ‘Auto-login username’
        - 4.14.2 ‘Terminal-type string’
        - 4.14.3 ‘Terminal speeds’
        - 4.14.4 Setting environment variables on the server
    - 4.15 The Proxy panel
        - 4.15.1 Setting the proxy type
        - 4.15.2 Excluding parts of the network from proxying
        - 4.15.3 Name resolution when using a proxy
        - 4.15.4 Username and password
        - 4.15.5 Specifying the Telnet or Local proxy command
    - 4.16 The Telnet panel
        - 4.16.1 ‘Handling of OLD_ENVIRON ambiguity’
        - 4.16.2 Passive and active Telnet negotiation modes
        - 4.16.3 ‘Keyboard sends Telnet special commands’
        - 4.16.4 ‘Return key sends Telnet New Line instead of ^M’
    - 4.17 The Rlogin panel
        - 4.17.1 ‘Local username’
    - 4.18 The SSH panel

        - 4.18.1 Executing a specific command on the server
        - 4.18.2 ‘Don't start a shell or command at all’
        - 4.18.3 ‘Enable compression’
        - 4.18.4 ‘Preferred SSH protocol version’
        - 4.18.5 Encryption algorithm selection
    - 4.19 The Kex panel
        - 4.19.1 Key exchange algorithm selection
        - 4.19.2 Repeat key exchange
    - 4.20 The Auth panel
        - 4.20.1 ‘Bypass authentication entirely’
        - 4.20.2 ‘Attempt authentication using Pageant’
        - 4.20.3 ‘Attempt TIS or CryptoCard authentication’
        - 4.20.4 ‘Attempt keyboard-interactive authentication’
        - 4.20.5 ‘Allow agent forwarding’
        - 4.20.6 ‘Allow attempted changes of username in SSH-2’
        - 4.20.7 ‘Private key file for authentication’
    - 4.21 The TTY panel
        - 4.21.1 ‘Don't allocate a pseudo-terminal’
        - 4.21.2 Sending terminal modes
    - 4.22 The X11 panel
        - 4.22.1 Remote X11 authentication
    - 4.23 The Tunnels panel
        - 4.23.1 Controlling the visibility of forwarded ports
        - 4.23.2 Selecting Internet protocol version for forwarded ports
    - 4.24 The Bugs panel
        - 4.24.1 ‘Chokes on SSH-1 ignore messages’
        - 4.24.2 ‘Refuses all SSH-1 password camouflage’
        - 4.24.3 ‘Chokes on SSH-1 RSA authentication’
        - 4.24.4 ‘Miscomputes SSH-2 HMAC keys’
        - 4.24.5 ‘Miscomputes SSH-2 encryption keys’
        - 4.24.6 ‘Requires padding on SSH-2 RSA signatures’
        - 4.24.7 ‘Misuses the session ID in SSH-2 PK auth’
        - 4.24.8 ‘Handles SSH-2 key re-exchange badly’
    - 4.25 The Serial panel
        - 4.25.1 Selecting a serial line to connect to
        - 4.25.2 Selecting the speed of your serial line
        - 4.25.3 Selecting the number of data bits
        - 4.25.4 Selecting the number of stop bits
        - 4.25.5 Selecting the serial parity checking scheme
        - 4.25.6 Selecting the serial flow control scheme
    - 4.26 Storing configuration in a file

Chapter 4: Configuring PuTTY

This chapter describes all the configuration options in PuTTY.

PuTTY is configured using the control panel that comes up before you start a session. Some options can also be changed in the middle of a session, by selecting ‘Change Settings’ from the window menu.

4.1 The Session panel

The Session configuration panel contains the basic options you need to specify in order to open a session at all, and also allows you to save your settings to be reloaded later.

4.1.1 The host name section

The top box on the Session panel, labelled 'Specify your connection by host name', contains the details that need to be filled in before PuTTY can open a session at all.

- The 'Host Name' box is where you type the name, or the IP address, of the server you want to connect to.
- The 'Connection type' radio buttons let you choose what type of connection you want to make: a raw connection, a Telnet connection, an Rlogin connection, an SSH connection, or a connection to a local serial line. (See section 1.2 for a summary of the differences between SSH, Telnet and rlogin; see section 3.6 for an explanation of 'raw' connections; see section 3.7 for information about using a serial line.)
- The 'Port' box lets you specify which port number on the server to connect to. If you select Telnet, Rlogin, or SSH, this box will be filled in automatically to the usual value, and you will only need to change it if you have an unusual server. If you select Raw mode, you will almost certainly need to fill in the 'Port' box yourself.

If you select 'Serial' from the 'Connection type' radio buttons, the 'Host Name' and 'Port' boxes are replaced by 'Serial line' and 'Speed'; see section 4.25 for more details of these.

4.1.2 Loading and storing saved sessions

The next part of the Session configuration panel allows you to save your preferred PuTTY options so they will appear automatically the next time you start PuTTY. It also allows you to create *saved sessions*, which contain a full set of configuration options plus a host name and protocol. A saved session contains all the information PuTTY needs to start exactly the session you want.

- To save your default settings: first set up the settings the way you want them saved. Then come back to the Session panel. Select the 'Default Settings' entry in the saved sessions list, with a single click. Then press the 'Save' button.

  Note that PuTTY does not allow you to save a host name into the Default Settings entry. This ensures that when PuTTY is started up, the host name box is always empty, so a user can always just type in a host name and connect.

If there is a specific host you want to store the details of how to connect to, you should create a saved session, which will be separate from the Default Settings.

- To save a session: first go through the rest of the configuration box setting up all the options you want. Then come back to the Session panel. Enter a name for the saved session in the 'Saved Sessions' input box. (The server name is often a good choice for a saved session name.) Then press the 'Save' button. Your saved session name should now appear in the list box.

  You can also save settings in mid-session, from the 'Change Settings' dialog. Settings changed since the start of the session will be saved with their current values; as well as settings changed through the dialog, this includes changes in window size, window title changes sent by the server, and so on.

- To reload a saved session: single-click to select the session name in the list box, and then press the 'Load' button. Your saved settings should all appear in the configuration panel.
- To modify a saved session: first load it as described above. Then make the changes you want. Come back to the Session panel, and press the 'Save' button. The new settings will be saved over the top of the old ones.

To save the new settings under a different name, you can enter the new name in the ‘Saved Sessions’ box, or single-click to select a session name in the list box to overwrite that session. To save ‘Default Settings’, you must single-click the name before saving.

- To start a saved session immediately: double-click on the session name in the list box.
- To delete a saved session: single-click to select the session name in the list box, and then press the ‘Delete’ button.

Each saved session is independent of the Default Settings configuration. If you change your preferences and update Default Settings, you must also update every saved session separately.

Saved sessions are stored in the Registry, at the location

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

If you need to store them in a file, you could try the method described in section 4.26.

4.1.3 ‘Close Window on Exit’

Finally in the Session panel, there is an option labelled ‘Close Window on Exit’. This controls whether the PuTTY terminal window disappears as soon as the session inside it terminates. If you are likely to want to copy and paste text out of the session after it has terminated, or restart the session, you should arrange for this option to be off.

‘Close Window On Exit’ has three settings. ‘Always’ means always close the window on exit; ‘Never’ means never close on exit (always leave the window open, but inactive). The third setting, and the default one, is ‘Only on clean exit’. In this mode, a session which terminates normally will cause its window to close, but one which is aborted unexpectedly by network trouble or a confusing message from the server will leave the window up.

4.2 The Logging panel

The Logging configuration panel allows you to save log files of your PuTTY sessions, for debugging, analysis or future reference.

The main option is a radio-button set that specifies whether PuTTY will log anything at all. The options are:

- ‘None’. This is the default option; in this mode PuTTY will not create a log file at all.
- ‘Printable output’. In this mode, a log file will be created and written to, but only printable text will be saved into it. The various terminal control codes that are typically sent down an interactive session alongside the printable text will be omitted. This might be a useful mode if you want to read a log file in a text editor and hope to be able to make sense of it.
- ‘All session output’. In this mode, *everything* sent by the server into your terminal session is logged. If you view the log file in a text editor, therefore, you may well find it full of strange control characters. This is a particularly useful mode if you are experiencing problems with PuTTY's terminal handling: you can record everything that went to the terminal, so that someone else can replay the session later in slow motion and watch to see what went wrong.
- ‘SSH packets’. In this mode (which is only used by SSH connections), the SSH message packets sent over the encrypted connection are written to the log file (as well as Event Log entries). You might need this to debug a network-level problem, or more likely to send to the PuTTY authors as part of a bug report. *BE WARNED* that if you log in using a password, the password can appear in the log file; see section 4.2.4 for options that may help to remove sensitive material from the log file before you send it to anyone else.
- ‘SSH packets and raw data’. In this mode, as well as the decrypted packets (as in the previous mode), the *raw* (encrypted, compressed, etc) packets are *also* logged. This could be useful to diagnose corruption in transit. (The same caveats as the previous mode apply, of course.)

Note that the non-SSH logging options ('Printable output' and 'All session output') only work with PuTTY proper; in programs without terminal emulation (such as Plink), they will have no effect, even if enabled via saved settings.

4.2.1 'Log file name'

In this edit box you enter the name of the file you want to log the session to. The 'Browse' button will let you look around your file system to find the right place to put the file; or if you already know exactly where you want it to go, you can just type a pathname into the edit box.

There are a few special features in this box. If you use the `&` character in the file name box, PuTTY will insert details of the current session in the name of the file it actually opens. The precise replacements it will do are:

- `&Y` will be replaced by the current year, as four digits.
- `&M` will be replaced by the current month, as two digits.
- `&D` will be replaced by the current day of the month, as two digits.
- `&T` will be replaced by the current time, as six digits (HHMMSS) with no punctuation.
- `&H` will be replaced by the host name you are connecting to.

For example, if you enter the host name `c:\puttylogs\log-&h-&y&m&d-&t.dat`, you will end up with files looking like

```
log-server1.example.com-20010528-110859.dat
log-unixbox.somewhere.org-20010611-221001.dat
```

4.2.2 'What to do if the log file already exists'

This control allows you to specify what PuTTY should do if it tries to start writing to a log file and it finds the file already exists. You might want to automatically destroy the existing log file and start a new one with the same name. Alternatively, you might want to open the existing log file and add data to the *end* of it. Finally (the default option), you might not want to have any automatic behaviour, but to ask the user every time the problem comes up.

4.2.3 'Flush log file frequently'

This option allows you to control how frequently logged data is flushed to disc. By default, PuTTY will flush data as soon as it is displayed, so that if you view the log file while a session is still open, it will be up to date; and if the client system crashes, there's a greater chance that the data will be preserved.

However, this can incur a performance penalty. If PuTTY is running slowly with logging enabled, you could try unchecking this option. Be warned that the log file may not always be up to date as a result (although it will of course be flushed when it is closed, for instance at the end of a session).

4.2.4 Options specific to SSH packet logging

These options only apply if SSH packet data is being logged.

The following options allow particularly sensitive portions of unencrypted packets to be automatically left out of the log file. They are only intended to deter casual nosiness; an attacker could glean a lot of useful information from even these obfuscated logs (e.g., length of password).

4.2.4.1 'Omit known password fields'

When checked, decrypted password fields are removed from the log of transmitted packets. (This includes any user responses to challenge-response authentication methods such as 'keyboard-interactive'.) This does not include X11 authentication data if using X11 forwarding.

Note that this will only omit data that PuTTY *knows* to be a password. However, if you start another login session within your PuTTY session, for instance, any password used will appear in the clear in the packet log. The next option may be of use to protect against this.

This option is enabled by default.

4.2.4.2 'Omit session data'

When checked, all decrypted 'session data' is omitted; this is defined as data in terminal sessions and in forwarded channels (TCP, X11, and authentication agent). This will usually substantially reduce the size of the resulting log file.

This option is disabled by default.

4.3 The Terminal panel

The Terminal configuration panel allows you to control the behaviour of PuTTY's terminal emulation.

4.3.1 'Auto wrap mode initially on'

Auto wrap mode controls what happens when text printed in a PuTTY window reaches the right-hand edge of the window.

With auto wrap mode on, if a long line of text reaches the right-hand edge, it will wrap over on to the next line so you can still see all the text. With auto wrap mode off, the cursor will stay at the right-hand edge of the screen, and all the characters in the line will be printed on top of each other.

If you are running a full-screen application and you occasionally find the screen scrolling up when it looks as if it shouldn't, you could try turning this option off.

Auto wrap mode can be turned on and off by control sequences sent by the server. This configuration option controls the *default* state, which will be restored when you reset the terminal (see section 3.1.3.6). However, if you modify this option in mid-session using 'Change Settings', it will take effect immediately.

4.3.2 'DEC Origin Mode initially on'

DEC Origin Mode is a minor option which controls how PuTTY interprets cursor-position control sequences sent by the server.

The server can send a control sequence that restricts the scrolling region of the display. For example, in an editor, the server might reserve a line at the top of the screen and a line at the bottom, and might send a control sequence that causes scrolling operations to affect only the remaining lines.

With DEC Origin Mode on, cursor coordinates are counted from the top of the scrolling region. With it turned off, cursor coordinates are counted from the top of the whole screen regardless of the scrolling region.

It is unlikely you would need to change this option, but if you find a full-screen application is displaying pieces of text in what looks like the wrong part of the screen, you could try turning DEC Origin Mode on to see whether that helps.

DEC Origin Mode can be turned on and off by control sequences sent by the server. This configuration option controls the *default* state, which will be restored when you reset the terminal (see section 3.1.3.6). However, if you modify this option in mid-session using 'Change Settings', it will take effect immediately.

4.3.3 'Implicit CR in every LF'

Most servers send two control characters, CR and LF, to start a new line of the screen. The CR character makes the cursor return to the left-hand side of the screen. The LF character makes the cursor move one line down (and might make the screen scroll).

Some servers only send LF, and expect the terminal to move the cursor over to the left automatically. If you come across a server that does this, you will see a stepped effect on the screen, like this:

```
First line of text
                  Second line
                             Third line
```

If this happens to you, try enabling the 'Implicit CR in every LF' option, and things might go back to normal:

```
First line of text
Second line
Third line
```

4.3.4 'Use background colour to erase screen'

Not all terminals agree on what colour to turn the screen when the server sends a 'clear screen' sequence. Some terminals believe the screen should always be cleared to the *default* background colour. Others believe the screen should be cleared to whatever the server has selected as a background colour.

There exist applications that expect both kinds of behaviour. Therefore, PuTTY can be configured to do either.

With this option disabled, screen clearing is always done in the default background colour. With this option enabled, it is done in the *current* background colour.

Background-colour erase can be turned on and off by control sequences sent by the server. This configuration option controls the *default* state, which will be restored when you reset the terminal (see section 3.1.3.6). However, if you modify this option in mid-session using 'Change Settings', it will take effect immediately.

4.3.5 'Enable blinking text'

The server can ask PuTTY to display text that blinks on and off. This is very distracting, so PuTTY allows you to turn blinking text off completely.

When blinking text is disabled and the server attempts to make some text blink, PuTTY will instead display the text with a bolded background colour.

Blinking text can be turned on and off by control sequences sent by the server. This configuration option controls the *default* state, which will be restored when you reset the terminal (see section 3.1.3.6). However, if you modify this option in mid-session using 'Change Settings', it will take effect immediately.

4.3.6 'Answerback to ^E'

This option controls what PuTTY will send back to the server if the server sends it the ^E enquiry character. Normally it just sends the string 'PuTTY'.

If you accidentally write the contents of a binary file to your terminal, you will probably find that it contains more than one ^E character, and as a result your next command line will probably read 'PuTTYPuTTYPuTTY...' as if you had typed the answerback string multiple times at the keyboard. If you set the answerback string to be empty, this problem should go away, but doing so might cause other problems.

Note that this is *not* the feature of PuTTY which the server will typically use to determine your terminal type. That feature is the 'Terminal-type string' in the Connection panel; see section 4.14.2 for details.

You can include control characters in the answerback string using `^C` notation. (Use `^~` to get a literal ^.)

4.3.7 'Local echo'

With local echo disabled, characters you type into the PuTTY window are not echoed in the window *by PuTTY*. They are simply sent to the server. (The *server* might choose to echo them back to you; this can't be controlled from the PuTTY control panel.)

Some types of session need local echo, and many do not. In its default mode, PuTTY will automatically attempt to deduce whether or not local echo is appropriate for the session you are working in. If you find it has made the wrong decision, you can use this configuration option to override its choice: you can force local echo to be turned on, or force it to be turned off, instead of relying on the automatic detection.

4.3.8 'Local line editing'

Normally, every character you type into the PuTTY window is sent immediately to the server the moment you type it.

If you enable local line editing, this changes. PuTTY will let you edit a whole line at a time locally, and the line will only be sent to the server when you press Return. If you make a mistake, you can use the Backspace key to correct it before you press Return, and the server will never see the mistake.

Since it is hard to edit a line locally without being able to see it, local line editing is mostly used in conjunction with local echo (section 4.3.7). This makes it ideal for use in raw mode or when connecting to MUDs or talkers. (Although some more advanced MUDs do occasionally turn local line editing on and turn local echo off, in order to accept a password from the user.)

Some types of session need local line editing, and many do not. In its default mode, PuTTY will automatically attempt to deduce whether or not local line editing is appropriate for the session you are working in. If you find it has made the wrong decision, you can use this configuration option to override its choice: you can force local line editing to be turned on, or force it to be turned off, instead of relying on the automatic detection.

4.3.9 Remote-controlled printing

A lot of VT100-compatible terminals support printing under control of the remote server. PuTTY supports this feature as well, but it is turned off by default.

To enable remote-controlled printing, choose a printer from the 'Printer to send ANSI printer output to' drop-down list box. This should allow you to select from all the printers you have installed drivers for on your computer. Alternatively, you can type the network name of a networked printer (for example, `\\printserver\printer1`) even if you haven't already installed a driver for it on your own machine.

When the remote server attempts to print some data, PuTTY will send that data to the printer *raw* - without translating it, attempting to format it, or doing anything else to it. It is up to you to ensure your remote server knows what type of printer it is talking to.

Since PuTTY sends data to the printer raw, it cannot offer options such as portrait versus landscape, print quality, or paper tray selection. All these things would be done by your PC printer driver (which PuTTY bypasses); if you need them done, you will have to find a way to configure your remote server to do them.

To disable remote printing again, choose 'None (printing disabled)' from the printer selection list. This is the default state.

4.4 The Keyboard panel

The Keyboard configuration panel allows you to control the behaviour of the keyboard in PuTTY. The correct state for many of these settings depends on what the server to which PuTTY is connecting expects. With a Unix server, this is likely to depend on the `termcap` or `terminfo` entry it uses, which in turn is likely to be controlled by the 'Terminal-type string' setting in the Connection panel; see section 4.14.2 for details. If none of the settings here seems to help, you may find question A.7.15 to be useful.

4.4.1 Changing the action of the Backspace key

Some terminals believe that the Backspace key should send the same thing to the server as Control-H (ASCII code 8). Other terminals believe that the Backspace key should send ASCII code 127 (usually known as Control-?) so that it can be distinguished from Control-H. This option allows you to choose which code PuTTY generates when you press Backspace.

If you are connecting over SSH, PuTTY by default tells the server the value of this option (see section 4.21.2), so you may find that the Backspace key does the right thing either way. Similarly, if you are connecting to a Unix system, you will probably find that the Unix `stty` command lets you configure which the server expects to see, so again you might not need to change which one PuTTY generates. On other systems, the server's expectation might be fixed and you might have no choice but to configure PuTTY.

If you do have the choice, we recommend configuring PuTTY to generate Control-? and configuring the server to expect it, because that allows applications such as `emacs` to use Control-H for help.

(Typing Shift-Backspace will cause PuTTY to send whichever code isn't configured here as the default.)

4.4.2 Changing the action of the Home and End keys

The Unix terminal emulator `rxvt` disagrees with the rest of the world about what character sequences should be sent to the server by the Home and End keys.

`xterm`, and other terminals, send `ESC [1~` for the Home key, and `ESC [4~` for the End key. `rxvt` sends `ESC [H` for the Home key and `ESC [Ow` for the End key.

If you find an application on which the Home and End keys aren't working, you could try switching this option to see if it helps.

4.4.3 Changing the action of the function keys and keypad

This option affects the function keys (F1 to F12) and the top row of the numeric keypad.

- In the default mode, labelled `ESC [n~`, the function keys generate sequences like `ESC [11~`, `ESC [12~` and so on. This matches the general behaviour of Digital's terminals.
- In Linux mode, F6 to F12 behave just like the default mode, but F1 to F5 generate `ESC [[A` through to `ESC [[E`. This mimics the Linux virtual console.

- In Xterm R6 mode, F5 to F12 behave like the default mode, but F1 to F4 generate `ESC OP` through to `ESC OS`, which are the sequences produced by the top row of the *keypad* on Digital's terminals.
- In VT400 mode, all the function keys behave like the default mode, but the actual top row of the numeric keypad generates `ESC OP` through to `ESC OS`.
- In VT100+ mode, the function keys generate `ESC OP` through to `ESC O[`
- In SCO mode, the function keys F1 to F12 generate `ESC [M` through to `ESC [X`. Together with shift, they generate `ESC [Y` through to `ESC [j`. With control they generate `ESC [k` through to `ESC [v`, and with shift and control together they generate `ESC [w` through to `ESC [{`.

If you don't know what any of this means, you probably don't need to fiddle with it.

4.4.4 Controlling Application Cursor Keys mode

Application Cursor Keys mode is a way for the server to change the control sequences sent by the arrow keys. In normal mode, the arrow keys send `ESC [A` through to `ESC [D`. In application mode, they send `ESC OA` through to `ESC OD`.

Application Cursor Keys mode can be turned on and off by the server, depending on the application. PuTTY allows you to configure the initial state.

You can also disable application cursor keys mode completely, using the 'Features' configuration panel; see section 4.6.1.

4.4.5 Controlling Application Keypad mode

Application Keypad mode is a way for the server to change the behaviour of the numeric keypad.

In normal mode, the keypad behaves like a normal Windows keypad: with NumLock on, the number keys generate numbers, and with NumLock off they act like the arrow keys and Home, End etc.

In application mode, all the keypad keys send special control sequences, *including* Num Lock. Num Lock stops behaving like Num Lock and becomes another function key.

Depending on which version of Windows you run, you may find the Num Lock light still flashes on and off every time you press Num Lock, even when application mode is active and Num Lock is acting like a function key. This is unavoidable.

Application keypad mode can be turned on and off by the server, depending on the application. PuTTY allows you to configure the initial state.

You can also disable application keypad mode completely, using the 'Features' configuration panel; see section 4.6.1.

4.4.6 Using NetHack keypad mode

PuTTY has a special mode for playing NetHack. You can enable it by selecting 'NetHack' in the 'Initial state of numeric keypad' control.

In this mode, the numeric keypad keys 1-9 generate the NetHack movement commands (`hjklyubn`). The 5 key generates the `.` command (do nothing).

In addition, pressing Shift or Ctrl with the keypad keys generate the Shift- or Ctrl-keys you would expect (e.g. keypad-7 generates 'y', so Shift-keypad-7 generates 'Y' and Ctrl-keypad-7 generates Ctrl-Y); these commands tell NetHack to keep moving you in the same direction until you encounter something interesting.

For some reason, this feature only works properly when Num Lock is on. We don't know why.

4.4.7 Enabling a DEC-like Compose key

DEC terminals have a Compose key, which provides an easy-to-remember way of typing accented characters. You press Compose and then type two more characters. The two characters are 'combined' to produce an accented character. The choices of character are designed to be easy to remember; for example, composing 'e' and '`' produces the 'è' character.

If your keyboard has a Windows Application key, it acts as a Compose key in PuTTY. Alternatively, if you enable the 'AltGr acts as Compose key' option, the AltGr key will become a Compose key.

4.4.8 'Control-Alt is different from AltGr'

Some old keyboards do not have an AltGr key, which can make it difficult to type some characters. PuTTY can be configured to treat the key combination Ctrl + Left Alt the same way as the AltGr key.

By default, this checkbox is checked, and the key combination Ctrl + Left Alt does something completely different. PuTTY's usual handling of the left Alt key is to prefix the Escape (Control-`[`) character to whatever character sequence the rest of the keypress would generate. For example, Alt-A generates Escape followed by `a`. So Alt-Ctrl-A would generate Escape, followed by Control-A.

If you uncheck this box, Ctrl-Alt will become a synonym for AltGr, so you can use it to type extra graphic characters if your keyboard has any.

(However, Ctrl-Alt will never act as a Compose key, regardless of the setting of 'AltGr acts as Compose key' described in section 4.4.7.)

4.5 The Bell panel

The Bell panel controls the terminal bell feature: the server's ability to cause PuTTY to beep at you.

In the default configuration, when the server sends the character with ASCII code 7 (Control-G), PuTTY will play the Windows Default Beep sound. This is not always what you want the terminal bell feature to do; the Bell panel allows you to configure alternative actions.

4.5.1 'Set the style of bell'

This control allows you to select various different actions to occur on a terminal bell:

- Selecting 'None' disables the bell completely. In this mode, the server can send as many Control-G characters as it likes and nothing at all will happen.
- 'Make default system alert sound' is the default setting. It causes the Windows 'Default Beep' sound to be played. To change what this sound is, or to test it if nothing seems to be happening, use the Sound configurer in the Windows Control Panel.
- 'Visual bell' is a silent alternative to a beeping computer. In this mode, when the server sends a Control-G, the whole PuTTY window will flash white for a fraction of a second.
- 'Beep using the PC speaker' is self-explanatory.
- 'Play a custom sound file' allows you to specify a particular sound file to be used by PuTTY alone, or even by a particular individual PuTTY session. This allows you to distinguish your PuTTY beeps from any other beeps on the system. If you select this option, you will also need to enter the name of your sound file in the edit control 'Custom sound file to play as a bell'.

4.5.2 ‘Taskbar/caption indication on bell’

This feature controls what happens to the PuTTY window's entry in the Windows Taskbar if a bell occurs while the window does not have the input focus.

In the default state (‘Disabled’) nothing unusual happens.

If you select ‘Steady’, then when a bell occurs and the window is not in focus, the window's Taskbar entry and its title bar will change colour to let you know that PuTTY session is asking for your attention. The change of colour will persist until you select the window, so you can leave several PuTTY windows minimised in your terminal, go away from your keyboard, and be sure not to have missed any important beeps when you get back.

‘Flashing’ is even more eye-catching: the Taskbar entry will continuously flash on and off until you select the window.

4.5.3 ‘Control the bell overload behaviour’

A common user error in a terminal session is to accidentally run the Unix command `cat` (or equivalent) on an inappropriate file type, such as an executable, image file, or ZIP file. This produces a huge stream of non-text characters sent to the terminal, which typically includes a lot of bell characters. As a result of this the terminal often doesn't stop beeping for ten minutes, and everybody else in the office gets annoyed.

To try to avoid this behaviour, or any other cause of excessive beeping, PuTTY includes a bell overload management feature. In the default configuration, receiving more than five bell characters in a two-second period will cause the overload feature to activate. Once the overload feature is active, further bells will have no effect at all, so the rest of your binary file will be sent to the screen in silence. After a period of five seconds during which no further bells are received, the overload feature will turn itself off again and bells will be re-enabled.

If you want this feature completely disabled, you can turn it off using the checkbox ‘Bell is temporarily disabled when over-used’.

Alternatively, if you like the bell overload feature but don't agree with the settings, you can configure the details: how many bells constitute an overload, how short a time period they have to arrive in to do so, and how much silent time is required before the overload feature will deactivate itself.

Bell overload mode is always deactivated by any keypress in the terminal. This means it can respond to large unexpected streams of data, but does not interfere with ordinary command-line activities that generate beeps (such as filename completion).

4.6 The Features panel

PuTTY's terminal emulation is very highly featured, and can do a lot of things under remote server control. Some of these features can cause problems due to buggy or strangely configured server applications.

The Features configuration panel allows you to disable some of PuTTY's more advanced terminal features, in case they cause trouble.

4.6.1 Disabling application keypad and cursor keys

Application keypad mode (see section 4.4.5) and application cursor keys mode (see section 4.4.4) alter the behaviour of the keypad and cursor keys. Some applications enable these modes but then do not deal correctly

with the modified keys. You can force these modes to be permanently disabled no matter what the server tries to do.

4.6.2 Disabling `xterm`-style mouse reporting

PuTTY allows the server to send control codes that let it take over the mouse and use it for purposes other than copy and paste. Applications which use this feature include the text-mode web browser `links`, the Usenet newsreader `trn` version 4, and the file manager `mc` (Midnight Commander).

If you find this feature inconvenient, you can disable it using the 'Disable xterm-style mouse reporting' control. With this box ticked, the mouse will *always* do copy and paste in the normal way.

Note that even if the application takes over the mouse, you can still manage PuTTY's copy and paste by holding down the Shift key while you select and paste, unless you have deliberately turned this feature off (see section 4.11.3).

4.6.3 Disabling remote terminal resizing

PuTTY has the ability to change the terminal's size and position in response to commands from the server. If you find PuTTY is doing this unexpectedly or inconveniently, you can tell PuTTY not to respond to those server commands.

4.6.4 Disabling switching to the alternate screen

Many terminals, including PuTTY, support an 'alternate screen'. This is the same size as the ordinary terminal screen, but separate. Typically a screen-based program such as a text editor might switch the terminal to the alternate screen before starting up. Then at the end of the run, it switches back to the primary screen, and you see the screen contents just as they were before starting the editor.

Some people prefer this not to happen. If you want your editor to run in the same screen as the rest of your terminal activity, you can disable the alternate screen feature completely.

4.6.5 Disabling remote window title changing

PuTTY has the ability to change the window title in response to commands from the server. If you find PuTTY is doing this unexpectedly or inconveniently, you can tell PuTTY not to respond to those server commands.

4.6.6 Response to remote window title querying

PuTTY can optionally provide the xterm service of allowing server applications to find out the local window title. This feature is disabled by default, but you can turn it on if you really want it.

NOTE that this feature is a *potential security hazard*. If a malicious application can write data to your terminal (for example, if you merely `cat` a file owned by someone else on the server machine), it can change your window title (unless you have disabled this as mentioned in section 4.6.5) and then use this service to have the new window title sent back to the server as if typed at the keyboard. This allows an attacker to fake keypresses and potentially cause your server-side applications to do things you didn't want. Therefore this feature is disabled by default, and we recommend you do not set it to 'Window title' unless you *really* know what you are doing.

There are three settings for this option:

'None'

PuTTY makes no response whatsoever to the relevant escape sequence. This may upset server-side software that is expecting some sort of response.

‘Empty string’
: PuTTY makes a well-formed response, but leaves it blank. Thus, server-side software that expects a response is kept happy, but an attacker cannot influence the response string. This is probably the setting you want if you have no better ideas.

‘Window title’
: PuTTY responds with the actual window title. This is dangerous for the reasons described above.

4.6.7 Disabling destructive backspace

Normally, when PuTTY receives character 127 (^?) from the server, it will perform a ‘destructive backspace’: move the cursor one space left and delete the character under it. This can apparently cause problems in some applications, so PuTTY provides the ability to configure character 127 to perform a normal backspace (without deleting a character) instead.

4.6.8 Disabling remote character set configuration

PuTTY has the ability to change its character set configuration in response to commands from the server. Some programs send these commands unexpectedly or inconveniently. In particular, (an IRC client) seems to have a habit of reconfiguring the character set to something other than the user intended.

If you find that accented characters are not showing up the way you expect them to, particularly if you're running BitchX, you could try disabling the remote character set configuration commands.

4.6.9 Disabling Arabic text shaping

PuTTY supports shaping of Arabic text, which means that if your server sends text written in the basic Unicode Arabic alphabet then it will convert it to the correct display forms before printing it on the screen.

If you are using full-screen software which was not expecting this to happen (especially if you are not an Arabic speaker and you unexpectedly find yourself dealing with Arabic text files in applications which are not Arabic-aware), you might find that the display becomes corrupted. By ticking this box, you can disable Arabic text shaping so that PuTTY displays precisely the characters it is told to display.

You may also find you need to disable bidirectional text display; see section 4.6.10.

4.6.10 Disabling bidirectional text display

PuTTY supports bidirectional text display, which means that if your server sends text written in a language which is usually displayed from right to left (such as Arabic or Hebrew) then PuTTY will automatically flip it round so that it is displayed in the right direction on the screen.

If you are using full-screen software which was not expecting this to happen (especially if you are not an Arabic speaker and you unexpectedly find yourself dealing with Arabic text files in applications which are not Arabic-aware), you might find that the display becomes corrupted. By ticking this box, you can disable bidirectional text display, so that PuTTY displays text from left to right in all situations.

You may also find you need to disable Arabic text shaping; see section 4.6.9.

4.7 The Window panel

The Window configuration panel allows you to control aspects of the PuTTY window.

4.7.1 Setting the size of the PuTTY window

The ‘Columns’ and ‘Rows’ boxes let you set the PuTTY window to a precise size. Of course you can also drag the window to a new size while a session is running.

4.7.2 What to do when the window is resized

These options allow you to control what happens when the user tries to resize the PuTTY window using its window furniture.

There are four options here:

- ‘Change the number of rows and columns’: the font size will not change. (This is the default.)
- ‘Change the size of the font’: the number of rows and columns in the terminal will stay the same, and the font size will change.
- ‘Change font size when maximised’: when the window is resized, the number of rows and columns will change, *except* when the window is maximised (or restored), when the font size will change.
- ‘Forbid resizing completely’: the terminal will refuse to be resized at all.

4.7.3 Controlling scrollback

These options let you configure the way PuTTY keeps text after it scrolls off the top of the screen (see section 3.1.2).

The ‘Lines of scrollback’ box lets you configure how many lines of text PuTTY keeps. The ‘Display scrollbar’ options allow you to hide the scrollbar (although you can still view the scrollback using the keyboard as described in section 3.1.2). You can separately configure whether the scrollbar is shown in full-screen mode and in normal modes.

If you are viewing part of the scrollback when the server sends more text to PuTTY, the screen will revert to showing the current terminal contents. You can disable this behaviour by turning off ‘Reset scrollback on display activity’. You can also make the screen revert when you press a key, by turning on ‘Reset scrollback on keypress’.

4.7.4 ‘Push erased text into scrollback’

When this option is enabled, the contents of the terminal screen will be pushed into the scrollback when a server-side application clears the screen, so that your scrollback will contain a better record of what was on your screen in the past.

If the application switches to the alternate screen (see section 4.6.4 for more about this), then the contents of the primary screen will be visible in the scrollback until the application switches back again.

This option is enabled by default.

4.8 The Appearance panel

The Appearance configuration panel allows you to control aspects of the appearance of PuTTY's window.

4.8.1 Controlling the appearance of the cursor

The ‘Cursor appearance’ option lets you configure the cursor to be a block, an underline, or a vertical line. A block cursor becomes an empty box when the window loses focus; an underline or a vertical line becomes dotted.

The ‘Cursor blinks’ option makes the cursor blink on and off. This works in any of the cursor modes.

4.8.2 Controlling the font used in the terminal window

This option allows you to choose what font, in what size, the PuTTY terminal window uses to display the text in the session. You will be offered a choice from all the fixed-width fonts installed on the system. (VT100-style terminal handling can only deal with fixed-width fonts.)

4.8.3 ‘Hide mouse pointer when typing in window’

If you enable this option, the mouse pointer will disappear if the PuTTY window is selected and you press a key. This way, it will not obscure any of the text in the window while you work in your session. As soon as you move the mouse, the pointer will reappear.

This option is disabled by default, so the mouse pointer remains visible at all times.

4.8.4 Controlling the window border

PuTTY allows you to configure the appearance of the window border to some extent.

The checkbox marked ‘Sunken-edge border’ changes the appearance of the window border to something more like a DOS box: the inside edge of the border is highlighted as if it sank down to meet the surface inside the window. This makes the border a little bit thicker as well. It's hard to describe well. Try it and see if you like it.

You can also configure a completely blank gap between the text in the window and the border, using the ‘Gap between text and window edge’ control. By default this is set at one pixel. You can reduce it to zero, or increase it further.

4.9 The Behaviour panel

The Behaviour configuration panel allows you to control aspects of the behaviour of PuTTY's window.

4.9.1 Controlling the window title

The ‘Window title’ edit box allows you to set the title of the PuTTY window. By default the window title will contain the host name followed by ‘PuTTY’, for example `server1.example.com - PuTTY`. If you want a different window title, this is where to set it.

PuTTY allows the server to send `xterm` control sequences which modify the title of the window in mid-session (unless this is disabled - see section 4.6.5); the title string set here is therefore only the *initial* window title.

As well as the *window* title, there is also an `xterm` sequence to modify the title of the window's *icon*. This makes sense in a windowing system where the window becomes an icon when minimised, such as Windows 3.1 or most X Window System setups; but in the Windows 95-like user interface it isn't as applicable.

By default, PuTTY only uses the server-supplied *window* title, and ignores the icon title entirely. If for some reason you want to see both titles, check the box marked ‘Separate window and icon titles’. If you do this, PuTTY's window title and Taskbar caption will change into the server-supplied icon title if you minimise the

PuTTY window, and change back to the server-supplied window title if you restore it. (If the server has not bothered to supply a window or icon title, none of this will happen.)

4.9.2 'Warn before closing window'

If you press the Close button in a PuTTY window that contains a running session, PuTTY will put up a warning window asking if you really meant to close the window. A window whose session has already terminated can always be closed without a warning.

If you want to be able to close a window quickly, you can disable the 'Warn before closing window' option.

4.9.3 'Window closes on ALT-F4'

By default, pressing ALT-F4 causes the window to close (or a warning box to appear; see section 4.9.2). If you disable the 'Window closes on ALT-F4' option, then pressing ALT-F4 will simply send a key sequence to the server.

4.9.4 'System menu appears on ALT-Space'

If this option is enabled, then pressing ALT-Space will bring up the PuTTY window's menu, like clicking on the top left corner. If it is disabled, then pressing ALT-Space will just send `ESC SPACE` to the server.

Some accessibility programs for Windows may need this option enabling to be able to control PuTTY's window successfully. For instance, Dragon NaturallySpeaking requires it both to open the system menu via voice, and to close, minimise, maximise and restore the window.

4.9.5 'System menu appears on Alt alone'

If this option is enabled, then pressing and releasing ALT will bring up the PuTTY window's menu, like clicking on the top left corner. If it is disabled, then pressing and releasing ALT will have no effect.

4.9.6 'Ensure window is always on top'

If this option is enabled, the PuTTY window will stay on top of all other windows.

4.9.7 'Full screen on Alt-Enter'

If this option is enabled, then pressing Alt-Enter will cause the PuTTY window to become full-screen. Pressing Alt-Enter again will restore the previous window size.

The full-screen feature is also available from the System menu, even when it is configured not to be available on the Alt-Enter key. See section 3.1.3.7.

4.10 The Translation panel

The Translation configuration panel allows you to control the translation between the character set understood by the server and the character set understood by PuTTY.

4.10.1 Controlling character set translation

During an interactive session, PuTTY receives a stream of 8-bit bytes from the server, and in order to display them on the screen it needs to know what character set to interpret them in.

There are a lot of character sets to choose from. The 'Received data assumed to be in which character set' option lets you select one. By default PuTTY will attempt to choose a character set that is right for your locale as reported by Windows; if it gets it wrong, you can select a different one using this control.

A few notable character sets are:

- The ISO-8859 series are all standard character sets that include various accented characters appropriate for different sets of languages.
- The Win125x series are defined by Microsoft, for similar purposes. In particular Win1252 is almost equivalent to ISO-8859-1, but contains a few extra characters such as matched quotes and the Euro symbol.
- If you want the old IBM PC character set with block graphics and line-drawing characters, you can select 'CP437'.
- PuTTY also supports Unicode mode, in which the data coming from the server is interpreted as being in the UTF-8 encoding of Unicode. If you select 'UTF-8' as a character set you can use this mode. Not all server-side applications will support it.

If you need support for a numeric code page which is not listed in the drop-down list, such as code page 866, then you can try entering its name manually (`CP866` for example) in the list box. If the underlying version of Windows has the appropriate translation table installed, PuTTY will use it.

4.10.2 'Treat CJK ambiguous characters as wide'

There are some Unicode characters whose width is not well-defined. In most contexts, such characters should be treated as single-width for the purposes of wrapping and so on; however, in some CJK contexts, they are better treated as double-width for historical reasons, and some server-side applications may expect them to be displayed as such. Setting this option will cause PuTTY to take the double-width interpretation.

If you use legacy CJK applications, and you find your lines are wrapping in the wrong places, or you are having other display problems, you might want to play with this setting.

This option only has any effect in UTF-8 mode (see section 4.10.1).

4.10.3 'Caps Lock acts as Cyrillic switch'

This feature allows you to switch between a US/UK keyboard layout and a Cyrillic keyboard layout by using the Caps Lock key, if you need to type (for example) Russian and English side by side in the same document.

Currently this feature is not expected to work properly if your native keyboard layout is not US or UK.

4.10.4 Controlling display of line-drawing characters

VT100-series terminals allow the server to send control sequences that shift temporarily into a separate character set for drawing simple lines and boxes. However, there are a variety of ways in which PuTTY can attempt to find appropriate characters, and the right one to use depends on the locally configured font. In general you should probably try lots of options until you find one that your particular font supports.

- 'Use Unicode line drawing code points' tries to use the box characters that are present in Unicode. For good Unicode-supporting fonts this is probably the most reliable and functional option.
- 'Poor man's line drawing' assumes that the font *cannot* generate the line and box characters at all, so it will use the `+`, `-` and `|` characters to draw approximations to boxes. You should use this option if none of

the other options works.

- ‘Font has XWindows encoding’ is for use with fonts that have a special encoding, where the lowest 32 character positions (below the ASCII printable range) contain the line-drawing characters. This is unlikely to be the case with any standard Windows font; it will probably only apply to custom-built fonts or fonts that have been automatically converted from the X Window System.
- ‘Use font in both ANSI and OEM modes’ tries to use the same font in two different character sets, to obtain a wider range of characters. This doesn't always work; some fonts claim to be a different size depending on which character set you try to use.
- ‘Use font in OEM mode only’ is more reliable than that, but can miss out other characters from the main character set.

4.10.5 Controlling copy and paste of line drawing characters

By default, when you copy and paste a piece of the PuTTY screen that contains VT100 line and box drawing characters, PuTTY will paste them in the form they appear on the screen: either Unicode line drawing code points, or the ‘poor man's’ line-drawing characters `+`, `-` and `|`. The checkbox ‘Copy and paste VT100 line drawing chars as lqqqk’ disables this feature, so line-drawing characters will be pasted as the ASCII characters that were printed to produce them. This will typically mean they come out mostly as `q` and `x`, with a scattering of `jklmntuvw` at the corners. This might be useful if you were trying to recreate the same box layout in another program, for example.

Note that this option only applies to line-drawing characters which *were* printed by using the VT100 mechanism. Line-drawing characters that were received as Unicode code points will paste as Unicode always.

4.11 The Selection panel

The Selection panel allows you to control the way copy and paste work in the PuTTY window.

4.11.1 Pasting in Rich Text Format

If you enable ‘Paste to clipboard in RTF as well as plain text’, PuTTY will write formatting information to the clipboard as well as the actual text you copy. The effect of this is that if you paste into (say) a word processor, the text will appear in the word processor in the same font, colour, and style (e.g. bold, underline) PuTTY was using to display it.

This option can easily be inconvenient, so by default it is disabled.

4.11.2 Changing the actions of the mouse buttons

PuTTY's copy and paste mechanism is by default modelled on the Unix `xterm` application. The X Window System uses a three-button mouse, and the convention is that the left button selects, the right button extends an existing selection, and the middle button pastes.

Windows often only has two mouse buttons, so in PuTTY's default configuration (‘Compromise’), the *right* button pastes, and the *middle* button (if you have one) extends a selection.

If you have a three-button mouse and you are already used to the `xterm` arrangement, you can select it using the ‘Action of mouse buttons’ control.

Alternatively, with the ‘Windows’ option selected, the middle button extends, and the right button brings up a context menu (on which one of the options is ‘Paste’). (This context menu is always available by holding down Ctrl and right-clicking, regardless of the setting of this option.)

4.11.3 ‘Shift overrides application's use of mouse’

PuTTY allows the server to send control codes that let it take over the mouse and use it for purposes other than copy and paste. Applications which use this feature include the text-mode web browser `links`, the Usenet newsreader `trn` version 4, and the file manager `mc` (Midnight Commander).

When running one of these applications, pressing the mouse buttons no longer performs copy and paste. If you do need to copy and paste, you can still do so if you hold down Shift while you do your mouse clicks.

However, it is possible in theory for applications to even detect and make use of Shift + mouse clicks. We don't know of any applications that do this, but in case someone ever writes one, unchecking the ‘Shift overrides application's use of mouse’ checkbox will cause Shift + mouse clicks to go to the server as well (so that mouse-driven copy and paste will be completely disabled).

If you want to prevent the application from taking over the mouse at all, you can do this using the Features control panel; see section 4.6.2.

4.11.4 Default selection mode

As described in section 3.1.1, PuTTY has two modes of selecting text to be copied to the clipboard. In the default mode (‘Normal’), dragging the mouse from point A to point B selects to the end of the line containing A, all the lines in between, and from the very beginning of the line containing B. In the other mode (‘Rectangular block’), dragging the mouse between two points defines a rectangle, and everything within that rectangle is copied.

Normally, you have to hold down Alt while dragging the mouse to select a rectangular block. Using the ‘Default selection mode’ control, you can set rectangular selection as the default, and then you have to hold down Alt to get the *normal* behaviour.

4.11.5 Configuring word-by-word selection

PuTTY will select a word at a time in the terminal window if you double-click to begin the drag. This panel allows you to control precisely what is considered to be a word.

Each character is given a *class*, which is a small number (typically 0, 1 or 2). PuTTY considers a single word to be any number of adjacent characters in the same class. So by modifying the assignment of characters to classes, you can modify the word-by-word selection behaviour.

In the default configuration, the character classes are:

- Class 0 contains white space and control characters.
- Class 1 contains most punctuation.
- Class 2 contains letters, numbers and a few pieces of punctuation (the double quote, minus sign, period, forward slash and underscore).

So, for example, if you assign the `@` symbol into character class 2, you will be able to select an e-mail address with just a double click.

In order to adjust these assignments, you start by selecting a group of characters in the list box. Then enter a class number in the edit box below, and press the ‘Set’ button.

This mechanism currently only covers ASCII characters, because it isn't feasible to expand the list to cover the whole of Unicode.

Character class definitions can be modified by control sequences sent by the server. This configuration option controls the *default* state, which will be restored when you reset the terminal (see section 3.1.3.6). However, if you modify this option in mid-session using 'Change Settings', it will take effect immediately.

4.12 The Colours panel

The Colours panel allows you to control PuTTY's use of colour.

4.12.1 'Allow terminal to specify ANSI colours'

This option is enabled by default. If it is disabled, PuTTY will ignore any control sequences sent by the server to request coloured text.

If you have a particularly garish application, you might want to turn this option off and make PuTTY only use the default foreground and background colours.

4.12.2 'Allow terminal to use xterm 256-colour mode'

This option is enabled by default. If it is disabled, PuTTY will ignore any control sequences sent by the server which use the extended 256-colour mode supported by recent versions of `xterm`.

If you have an application which is supposed to use 256-colour mode and it isn't working, you may find you need to tell your server that your terminal supports 256 colours. On Unix, you do this by ensuring that the setting of `TERM` describes a 256-colour-capable terminal. You can check this using a command such as `infocmp`:

```
$ infocmp | grep colors
        colors#256, cols#80, it#8, lines#24, pairs#256,
```

If you do not see '`colors#256`' in the output, you may need to change your terminal setting. On modern Linux machines, you could try '`xterm-256color`'.

4.12.3 'Bolded text is a different colour'

When the server sends a control sequence indicating that some text should be displayed in bold, PuTTY can handle this two ways. It can either change the font for a bold version, or use the same font in a brighter colour. This control lets you choose which.

By default the box is checked, so non-bold text is displayed in light grey and bold text is displayed in bright white (and similarly in other colours). If you uncheck the box, bold and non-bold text will be displayed in the same colour, and instead the font will change to indicate the difference.

4.12.4 'Attempt to use logical palettes'

Logical palettes are a mechanism by which a Windows application running on an 8-bit colour display can select precisely the colours it wants instead of going with the Windows standard defaults.

If you are not getting the colours you ask for on an 8-bit display, you can try enabling this option. However, be warned that it's never worked very well.

4.12.5 'Use system colours'

Enabling this option will cause PuTTY to ignore the configured colours for 'Default Background/Foreground' and 'Cursor Colour/Text' (see section 4.12.6), instead going with the system-wide defaults.

Note that non-bold and bold text will be the same colour if this option is enabled. You might want to change to indicating bold text by font changes (see section 4.12.3).

4.12.6 Adjusting the colours in the terminal window

The main colour control allows you to specify exactly what colours things should be displayed in. To modify one of the PuTTY colours, use the list box to select which colour you want to modify. The RGB values for that colour will appear on the right-hand side of the list box. Now, if you press the 'Modify' button, you will be presented with a colour selector, in which you can choose a new colour to go in place of the old one. (You may also edit the RGB values directly in the edit boxes, if you wish; each value is an integer from 0 to 255.)

PuTTY allows you to set the cursor colour, the default foreground and background, and the precise shades of all the ANSI configurable colours (black, red, green, yellow, blue, magenta, cyan, and white). You can also modify the precise shades used for the bold versions of these colours; these are used to display bold text if you have selected 'Bolded text is a different colour', and can also be used if the server asks specifically to use them. (Note that 'Default Bold Background' is *not* the background colour used for bold text; it is only used if the server specifically asks for a bold background.)

4.13 The Connection panel

The Connection panel allows you to configure options that apply to more than one type of connection.

4.13.1 Using keepalives to prevent disconnection

If you find your sessions are closing unexpectedly (most often with 'Connection reset by peer') after they have been idle for a while, you might want to try using this option.

Some network routers and firewalls need to keep track of all connections through them. Usually, these firewalls will assume a connection is dead if no data is transferred in either direction after a certain time interval. This can cause PuTTY sessions to be unexpectedly closed by the firewall if no traffic is seen in the session for some time.

The keepalive option ('Seconds between keepalives') allows you to configure PuTTY to send data through the session at regular intervals, in a way that does not disrupt the actual terminal session. If you find your firewall is cutting idle connections off, you can try entering a non-zero value in this field. The value is measured in seconds; so, for example, if your firewall cuts connections off after ten minutes then you might want to enter 300 seconds (5 minutes) in the box.

Note that keepalives are not always helpful. They help if you have a firewall which drops your connection after an idle period; but if the network between you and the server suffers from breaks in connectivity then keepalives can actually make things worse. If a session is idle, and connectivity is temporarily lost between the endpoints, but the connectivity is restored before either side tries to send anything, then there will be no problem - neither endpoint will notice that anything was wrong. However, if one side does send something during the break, it will repeatedly try to re-send, and eventually give up and abandon the connection. Then when connectivity is restored, the other side will find that the first side doesn't believe there is an open connection any more. Keepalives can make this sort of problem worse, because they increase the probability that PuTTY will attempt to send data during a break in connectivity. (Other types of periodic network activity can cause this behaviour; in particular, SSH-2 re-keys can have this effect. See section 4.19.2.)

Therefore, you might find that keepalives help connection loss, or you might find they make it worse, depending on what *kind* of network problems you have between you and the server.

Keepalives are only supported in Telnet and SSH; the Rlogin and Raw protocols offer no way of implementing them. (For an alternative, see section 4.13.3.)

Note that if you are using SSH-1 and the server has a bug that makes it unable to deal with SSH-1 ignore messages (see section 4.24.1), enabling keepalives will have no effect.

4.13.2 'Disable Nagle's algorithm'

Nagle's algorithm is a detail of TCP/IP implementations that tries to minimise the number of small data packets sent down a network connection. With Nagle's algorithm enabled, PuTTY's bandwidth usage will be slightly more efficient; with it disabled, you may find you get a faster response to your keystrokes when connecting to some types of server.

The Nagle algorithm is disabled by default for interactive connections.

4.13.3 'Enable TCP keepalives'

NOTE: TCP keepalives should not be confused with the application-level keepalives described in section 4.13.1. If in doubt, you probably want application-level keepalives; TCP keepalives are provided for completeness.

The idea of TCP keepalives is similar to application-level keepalives, and the same caveats apply. The main differences are:

- TCP keepalives are available on *all* connection types, including Raw and Rlogin.
- The interval between TCP keepalives is usually much longer, typically two hours; this is set by the operating system, and cannot be configured within PuTTY.
- If the operating system does not receive a response to a keepalive, it may send out more in quick succession and terminate the connection if no response is received.

TCP keepalives may be more useful for ensuring that half-open connections are terminated than for keeping a connection alive.

TCP keepalives are disabled by default.

4.13.4 'Internet protocol'

This option allows the user to select between the old and new Internet protocols and addressing schemes (IPv4 and IPv6). The default setting is 'Auto', which means PuTTY will do something sensible and try to guess which protocol you wanted. (If you specify a literal Internet address, it will use whichever protocol that address implies. If you provide a hostname, it will see what kinds of address exist for that hostname; it will use IPv6 if there is an IPv6 address available, and fall back to IPv4 if not.)

If you need to force PuTTY to use a particular protocol, you can explicitly set this to 'IPv4' or 'IPv6'.

4.14 The Data panel

The Data panel allows you to configure various pieces of data which can be sent to the server to affect your connection at the far end.

Each option on this panel applies to more than one protocol. Options which apply to only one protocol appear on that protocol's configuration panels.

4.14.1 'Auto-login username'

All three of the SSH, Telnet and Rlogin protocols allow you to specify what user name you want to log in as, without having to type it explicitly every time. (Some Telnet servers don't support this.)

In this box you can type that user name.

4.14.2 'Terminal-type string'

Most servers you might connect to with PuTTY are designed to be connected to from lots of different types of terminal. In order to send the right control sequences to each one, the server will need to know what type of terminal it is dealing with. Therefore, each of the SSH, Telnet and Rlogin protocols allow a text string to be sent down the connection describing the terminal. On a Unix server, this selects an entry from the `termcap` or `terminfo` database that tells applications what control sequences to send to the terminal, and what character sequences to expect the keyboard to generate.

PuTTY attempts to emulate the Unix `xterm` program, and by default it reflects this by sending `xterm` as a terminal-type string. If you find this is not doing what you want - perhaps the remote system reports 'Unknown terminal type' - you could try setting this to something different, such as `vt220`.

If you're not sure whether a problem is due to the terminal type setting or not, you probably need to consult the manual for your application or your server.

4.14.3 'Terminal speeds'

The Telnet, Rlogin, and SSH protocols allow the client to specify terminal speeds to the server.

This parameter does *not* affect the actual speed of the connection, which is always 'as fast as possible'; it is just a hint that is sometimes used by server software to modify its behaviour. For instance, if a slow speed is indicated, the server may switch to a less bandwidth-hungry display mode.

The value is usually meaningless in a network environment, but PuTTY lets you configure it, in case you find the server is reacting badly to the default value.

The format is a pair of numbers separated by a comma, for instance, `38400,38400`. The first number represents the output speed (*from* the server) in bits per second, and the second is the input speed (*to* the server). (Only the first is used in the Rlogin protocol.)

This option has no effect on Raw connections.

4.14.4 Setting environment variables on the server

The Telnet protocol provides a means for the client to pass environment variables to the server. Many Telnet servers have stopped supporting this feature due to security flaws, but PuTTY still supports it for the benefit of any servers which have found other ways around the security problems than just disabling the whole mechanism.

Version 2 of the SSH protocol also provides a similar mechanism, which is easier to implement without security flaws. Newer SSH-2 servers are more likely to support it than older ones.

This configuration data is not used in the SSH-1, rlogin or raw protocols.

To add an environment variable to the list transmitted down the connection, you enter the variable name in the 'Variable' box, enter its value in the 'Value' box, and press the 'Add' button. To remove one from the list, select it in the list box and press 'Remove'.

4.15 The Proxy panel

The Proxy panel allows you to configure PuTTY to use various types of proxy in order to make its network connections. The settings in this panel affect the primary network connection forming your PuTTY session, and also any extra connections made as a result of SSH port forwarding (see section 3.5).

4.15.1 Setting the proxy type

The 'Proxy type' radio buttons allow you to configure what type of proxy you want PuTTY to use for its network connections. The default setting is 'None'; in this mode no proxy is used for any connection.

- Selecting 'HTTP' allows you to proxy your connections through a web server supporting the HTTP `CONNECT` command, as documented in RFC 2817.
- Selecting 'SOCKS 4' or 'SOCKS 5' allows you to proxy your connections through a SOCKS server.
- Many firewalls implement a less formal type of proxy in which a user can make a Telnet connection directly to the firewall machine and enter a command such as `connect myhost.com 22` to connect through to an external host. Selecting 'Telnet' allows you to tell PuTTY to use this type of proxy.
- Selecting 'Local' allows you to specify an arbitrary command on the local machine to act as a proxy. When the session is started, instead of creating a TCP connection, PuTTY runs the command (specified in section 4.15.5), and uses its standard input and output streams.

  This could be used, for instance, to talk to some kind of network proxy that PuTTY does not natively support; or you could tunnel a connection over something other than TCP/IP entirely.

  If you want your local proxy command to make a secondary SSH connection to a proxy host and then tunnel the primary connection over that, you might well want the `-nc` command-line option in Plink. See section 3.8.3.14 for more information.

4.15.2 Excluding parts of the network from proxying

Typically you will only need to use a proxy to connect to non-local parts of your network; for example, your proxy might be required for connections outside your company's internal network. In the 'Exclude Hosts/IPs' box you can enter ranges of IP addresses, or ranges of DNS names, for which PuTTY will avoid using the proxy and make a direct connection instead.

The 'Exclude Hosts/IPs' box may contain more than one exclusion range, separated by commas. Each range can be an IP address or a DNS name, with a `*` character allowing wildcards. For example:

```
*.example.com
```

This excludes any host with a name ending in `.example.com` from proxying.

```
192.168.88.*
```

This excludes any host with an IP address starting with 192.168.88 from proxying.

```
192.168.88.*,*.example.com
```

This excludes both of the above ranges at once.

Connections to the local host (the host name `localhost`, and any loopback IP address) are never proxied, even if the proxy exclude list does not explicitly contain them. It is very unlikely that this behaviour would ever cause problems, but if it does you can change it by enabling 'Consider proxying local host connections'.

Note that if you are doing DNS at the proxy (see section 4.15.3), you should make sure that your proxy exclusion settings do not depend on knowing the IP address of a host. If the name is passed on to the proxy without PuTTY looking it up, it will never know the IP address and cannot check it against your list.

4.15.3 Name resolution when using a proxy

If you are using a proxy to access a private network, it can make a difference whether DNS name resolution is performed by PuTTY itself (on the client machine) or performed by the proxy.

The 'Do DNS name lookup at proxy end' configuration option allows you to control this. If you set it to 'No', PuTTY will always do its own DNS, and will always pass an IP address to the proxy. If you set it to 'Yes', PuTTY will always pass host names straight to the proxy without trying to look them up first.

If you set this option to 'Auto' (the default), PuTTY will do something it considers appropriate for each type of proxy. Telnet, HTTP, and SOCKS5 proxies will have host names passed straight to them; SOCKS4 proxies will not.

Note that if you are doing DNS at the proxy, you should make sure that your proxy exclusion settings (see section 4.15.2) do not depend on knowing the IP address of a host. If the name is passed on to the proxy without PuTTY looking it up, it will never know the IP address and cannot check it against your list.

The original SOCKS 4 protocol does not support proxy-side DNS. There is a protocol extension (SOCKS 4A) which does support it, but not all SOCKS 4 servers provide this extension. If you enable proxy DNS and your SOCKS 4 server cannot deal with it, this might be why.

4.15.4 Username and password

If your proxy requires authentication, you can enter a username and a password in the 'Username' and 'Password' boxes.

Note that if you save your session, the proxy password will be saved in plain text, so anyone who can access your PuTTY configuration data will be able to discover it.

Authentication is not fully supported for all forms of proxy:

- Username and password authentication is supported for HTTP proxies and SOCKS 5 proxies.
  - With SOCKS 5, authentication is via CHAP if the proxy supports it (this is not supported in PuTTYtel); otherwise the password is sent to the proxy in plain text.
  - With HTTP proxying, the only currently supported authentication method is 'basic', where the password is sent to the proxy in plain text.
- SOCKS 4 can use the 'Username' field, but does not support passwords.
- You can specify a way to include a username and password in the Telnet/Local proxy command (see section 4.15.5).

4.15.5 Specifying the Telnet or Local proxy command

If you are using the Telnet proxy type, the usual command required by the firewall's Telnet server is `connect`, followed by a host name and a port number. If your proxy needs a different command, you can enter an alternative here.

If you are using the Local proxy type, the local command to run is specified here.

In this string, you can use `\n` to represent a new-line, `\r` to represent a carriage return, `\t` to represent a tab character, and `\x` followed by two hex digits to represent any other character. `\\` is used to encode the `\` character itself.

Also, the special strings `%host` and `%port` will be replaced by the host name and port number you want to connect to. The strings `%user` and `%pass` will be replaced by the proxy username and password you specify. The strings

`%proxyhost` and `%proxyport` will be replaced by the host details specified on the *Proxy* panel, if any (this is most likely to be useful for the Local proxy type). To get a literal `%` sign, enter `%%`.

If a Telnet proxy server prompts for a username and password before commands can be sent, you can use a command such as:

```
%user\n%pass\nconnect %host %port\n
```

This will send your username and password as the first two lines to the proxy, followed by a command to connect to the desired host and port. Note that if you do not include the `%user` or `%pass` tokens in the Telnet command, then the 'Username' and 'Password' configuration fields will be ignored.

4.16 The Telnet panel

The Telnet panel allows you to configure options that only apply to Telnet sessions.

4.16.1 'Handling of OLD_ENVIRON ambiguity'

The original Telnet mechanism for passing environment variables was badly specified. At the time the standard (RFC 1408) was written, BSD telnet implementations were already supporting the feature, and the intention of the standard was to describe the behaviour the BSD implementations were already using.

Sadly there was a typing error in the standard when it was issued, and two vital function codes were specified the wrong way round. BSD implementations did not change, and the standard was not corrected. Therefore, it's possible you might find either BSD or RFC-compliant implementations out there. This switch allows you to choose which one PuTTY claims to be.

The problem was solved by issuing a second standard, defining a new Telnet mechanism called `NEW_ENVIRON`, which behaved exactly like the original `OLD_ENVIRON` but was not encumbered by existing implementations. Most Telnet servers now support this, and it's unambiguous. This feature should only be needed if you have trouble passing environment variables to quite an old server.

4.16.2 Passive and active Telnet negotiation modes

In a Telnet connection, there are two types of data passed between the client and the server: actual text, and *negotiations* about which Telnet extra features to use.

PuTTY can use two different strategies for negotiation:

- In *active* mode, PuTTY starts to send negotiations as soon as the connection is opened.
- In *passive* mode, PuTTY will wait to negotiate until it sees a negotiation from the server.

The obvious disadvantage of passive mode is that if the server is also operating in a passive mode, then negotiation will never begin at all. For this reason PuTTY defaults to active mode.

However, sometimes passive mode is required in order to successfully get through certain types of firewall and Telnet proxy server. If you have confusing trouble with a firewall, you could try enabling passive mode to see if it helps.

4.16.3 'Keyboard sends Telnet special commands'

If this box is checked, several key sequences will have their normal actions modified:

- the Backspace key on the keyboard will send the Telnet special backspace code;

- Control-C will send the Telnet special Interrupt Process code;
- Control-Z will send the Telnet special Suspend Process code.

You probably shouldn't enable this unless you know what you're doing.

4.16.4 'Return key sends Telnet New Line instead of ^M'

Unlike most other remote login protocols, the Telnet protocol has a special 'new line' code that is not the same as the usual line endings of Control-M or Control-J. By default, PuTTY sends the Telnet New Line code when you press Return, instead of sending Control-M as it does in most other protocols.

Most Unix-style Telnet servers don't mind whether they receive Telnet New Line or Control-M; some servers do expect New Line, and some servers prefer to see ^M. If you are seeing surprising behaviour when you press Return in a Telnet session, you might try turning this option off to see if it helps.

4.17 The Rlogin panel

The Rlogin panel allows you to configure options that only apply to Rlogin sessions.

4.17.1 'Local username'

Rlogin allows an automated (password-free) form of login by means of a file called `.rhosts` on the server. You put a line in your `.rhosts` file saying something like `jbloggs@pc1.example.com`, and then when you make an Rlogin connection the client transmits the username of the user running the Rlogin client. The server checks the username and hostname against `.rhosts`, and if they match it does not ask for a password.

This only works because Unix systems contain a safeguard to stop a user from pretending to be another user in an Rlogin connection. Rlogin connections have to come from port numbers below 1024, and Unix systems prohibit this to unprivileged processes; so when the server sees a connection from a low-numbered port, it assumes the client end of the connection is held by a privileged (and therefore trusted) process, so it believes the claim of who the user is.

Windows does not have this restriction: *any* user can initiate an outgoing connection from a low-numbered port. Hence, the Rlogin `.rhosts` mechanism is completely useless for securely distinguishing several different users on a Windows machine. If you have a `.rhosts` entry pointing at a Windows PC, you should assume that *anyone* using that PC can spoof your username in an Rlogin connection and access your account on the server.

The 'Local username' control allows you to specify what user name PuTTY should claim you have, in case it doesn't match your Windows user name (or in case you didn't bother to set up a Windows user name).

4.18 The SSH panel

The SSH panel allows you to configure options that only apply to SSH sessions.

4.18.1 Executing a specific command on the server

In SSH, you don't have to run a general shell session on the server. Instead, you can choose to run a single specific command (such as a mail user agent, for example). If you want to do this, enter the command in the 'Remote command' box.

Note that most servers will close the session after executing the command.

4.18.2 'Don't start a shell or command at all'

If you tick this box, PuTTY will not attempt to run a shell or command after connecting to the remote server. You might want to use this option if you are only using the SSH connection for port forwarding, and your user account on the server does not have the ability to run a shell.

This feature is only available in SSH protocol version 2 (since the version 1 protocol assumes you will always want to run a shell).

This feature can also be enabled using the `-N` command-line option; see section 3.8.3.13.

If you use this feature in Plink, you will not be able to terminate the Plink process by any graceful means; the only way to kill it will be by pressing Control-C or sending a kill signal from another program.

4.18.3 'Enable compression'

This enables data compression in the SSH connection: data sent by the server is compressed before sending, and decompressed at the client end. Likewise, data sent by PuTTY to the server is compressed first and the server decompresses it at the other end. This can help make the most of a low-bandwidth connection.

4.18.4 'Preferred SSH protocol version'

This allows you to select whether you would like to use SSH protocol version 1 or version 2.

PuTTY will attempt to use protocol 1 if the server you connect to does not offer protocol 2, and vice versa.

If you select '1 only' or '2 only' here, PuTTY will only connect if the server you connect to offers the SSH protocol version you have specified.

4.18.5 Encryption algorithm selection

PuTTY supports a variety of different encryption algorithms, and allows you to choose which one you prefer to use. You can do this by dragging the algorithms up and down in the list box (or moving them using the Up and Down buttons) to specify a preference order. When you make an SSH connection, PuTTY will search down the list from the top until it finds an algorithm supported by the server, and then use that.

PuTTY currently supports the following algorithms:

- AES (Rijndael) - 256, 192, or 128-bit SDCTR or CBC (SSH-2 only)
- Arcfour (RC4) - 256 or 128-bit stream cipher (SSH-2 only)
- Blowfish - 256-bit SDCTR (SSH-2 only) or 128-bit CBC
- Triple-DES - 168-bit SDCTR (SSH-2 only) or CBC
- Single-DES - 56-bit CBC (see below for SSH-2)

If the algorithm PuTTY finds is below the 'warn below here' line, you will see a warning box when you make the connection:

```
The first cipher supported by the server
is single-DES, which is below the configured
warning threshold.
Do you want to continue with this connection?
```

This warns you that the first available encryption is not a very secure one. Typically you would put the 'warn below here' line between the encryptions you consider secure and the ones you consider substandard. By

default, PuTTY supplies a preference order intended to reflect a reasonable preference in terms of security and speed.

In SSH-2, the encryption algorithm is negotiated independently for each direction of the connection, although PuTTY does not support separate configuration of the preference orders. As a result you may get two warnings similar to the one above, possibly with different encryptions.

Single-DES is not recommended in the SSH-2 protocol standards, but one or two server implementations do support it. PuTTY can use single-DES to interoperate with these servers if you enable the 'Enable legacy use of single-DES in SSH-2' option; by default this is disabled and PuTTY will stick to recommended ciphers.

4.19 The Kex panel

The Kex panel (short for 'key exchange') allows you to configure options related to SSH-2 key exchange.

Key exchange occurs at the start of an SSH connection (and occasionally thereafter); it establishes a shared secret that is used as the basis for all of SSH's security features. It is therefore very important for the security of the connection that the key exchange is secure.

Key exchange is a cryptographically intensive process; if either the client or the server is a relatively slow machine, the slower methods may take several tens of seconds to complete.

If connection startup is too slow, or the connection hangs periodically, you may want to try changing these settings.

If you don't understand what any of this means, it's safe to leave these settings alone.

This entire panel is only relevant to SSH protocol version 2; none of these settings affect SSH-1 at all.

4.19.1 Key exchange algorithm selection

PuTTY supports a variety of SSH-2 key exchange methods, and allows you to choose which one you prefer to use; configuration is similar to cipher selection (see section 4.18.5).

PuTTY currently supports the following varieties of Diffie-Hellman key exchange:

- 'Group 14': a well-known 2048-bit group.
- 'Group 1': a well-known 1024-bit group. This is less secure than group 14, but may be faster with slow client or server machines, and may be the only method supported by older server software.
- 'Group exchange': with this method, instead of using a fixed group, PuTTY requests that the server suggest a group to use for key exchange; the server can avoid groups known to be weak, and possibly invent new ones over time, without any changes required to PuTTY's configuration. We recommend use of this method, if possible.

If the first algorithm PuTTY finds is below the 'warn below here' line, you will see a warning box when you make the connection, similar to that for cipher selection (see section 4.18.5).

4.19.2 Repeat key exchange

If the session key negotiated at connection startup is used too much or for too long, it may become feasible to mount attacks against the SSH connection. Therefore, the SSH-2 protocol specifies that a new key exchange should take place every so often; this can be initiated by either the client or the server.

While this renegotiation is taking place, no data can pass through the SSH connection, so it may appear to 'freeze'. (The occurrence of repeat key exchange is noted in the Event Log; see section 3.1.3.1.) Usually the same algorithm is used as at the start of the connection, with a similar overhead.

These options control how often PuTTY will initiate a repeat key exchange ('rekey'). You can also force a key exchange at any time from the Special Commands menu (see section 3.1.3.2).

- 'Max minutes before rekey' specifies the amount of time that is allowed to elapse before a rekey is initiated. If this is set to zero, PuTTY will not rekey due to elapsed time. The SSH-2 protocol specification recommends a timeout of at most 60 minutes.

You might have a need to disable time-based rekeys completely for the same reasons that keepalives aren't always helpful. If you anticipate suffering a network dropout of several hours in the middle of an SSH connection, but were not actually planning to send *data* down that connection during those hours, then an attempted rekey in the middle of the dropout will probably cause the connection to be abandoned, whereas if rekeys are disabled then the connection should in principle survive (in the absence of interfering firewalls). See section 4.13.1 for more discussion of these issues; for these purposes, rekeys have much the same properties as keepalives. (Except that rekeys have cryptographic value in themselves, so you should bear that in mind when deciding whether to turn them off.) Note, however, the the SSH *server* can still initiate rekeys.

- 'Max data before rekey' specifies the amount of data (in bytes) that is permitted to flow in either direction before a rekey is initiated. If this is set to zero, PuTTY will not rekey due to transferred data. The SSH-2 protocol specification recommends a limit of at most 1 gigabyte.

  As well as specifying a value in bytes, the following shorthand can be used:

  - '`1k`' specifies 1 kilobyte (1024 bytes).
  - '`1M`' specifies 1 megabyte (1024 kilobytes).
  - '`1G`' specifies 1 gigabyte (1024 megabytes).

Disabling data-based rekeys entirely is a bad idea. The integrity, and to a lesser extent, confidentiality of the SSH-2 protocol depend in part on rekeys occuring before a 32-bit packet sequence number wraps around. Unlike time-based rekeys, data-based rekeys won't occur when the SSH connection is idle, so they shouldn't cause the same problems. The SSH-1 protocol, incidentally, has even weaker integrity protection than SSH-2 without rekeys.

4.20 The Auth panel

The Auth panel allows you to configure authentication options for SSH sessions.

4.20.1 'Bypass authentication entirely'

In SSH-2, it is possible to establish a connection without using SSH's mechanisms to identify or authenticate oneself to the server. Some servers may prefer to handle authentication in the data channel, for instance, or may simply require no authentication whatsoever.

By default, PuTTY assumes the server requires authentication (most do), and thus must provide a username. If you find you are getting unwanted username prompts, you could try checking this option.

This option only affects SSH-2 connections. SSH-1 connections always require an authentication step.

4.20.2 'Attempt authentication using Pageant'

If this option is enabled, then PuTTY will look for Pageant (the SSH private-key storage agent) and attempt to authenticate with any suitable public keys Pageant currently holds.

This behaviour is almost always desirable, and is therefore enabled by default. In rare cases you might need to turn it off in order to force authentication by some non-public-key method such as passwords.

This option can also be controlled using the `-noagent` command-line option. See section 3.8.3.9.

See chapter 9 for more information about Pageant in general.

4.20.3 'Attempt TIS or CryptoCard authentication'

TIS and CryptoCard authentication are (despite their names) generic forms of simple challenge/response authentication available in SSH protocol version 1 only. You might use them if you were using S/Key one-time passwords, for example, or if you had a physical security token that generated responses to authentication challenges.

With this switch enabled, PuTTY will attempt these forms of authentication if the server is willing to try them. You will be presented with a challenge string (which will be different every time) and must supply the correct response in order to log in. If your server supports this, you should talk to your system administrator about precisely what form these challenges and responses take.

4.20.4 'Attempt keyboard-interactive authentication'

The SSH-2 equivalent of TIS authentication is called 'keyboard-interactive'. It is a flexible authentication method using an arbitrary sequence of requests and responses; so it is not only useful for challenge/response mechanisms such as S/Key, but it can also be used for (for example) asking the user for a new password when the old one has expired.

PuTTY leaves this option enabled by default, but supplies a switch to turn it off in case you should have trouble with it.

4.20.5 'Allow agent forwarding'

This option allows the SSH server to open forwarded connections back to your local copy of Pageant. If you are not running Pageant, this option will do nothing.

See chapter 9 for general information on Pageant, and section 9.4 for information on agent forwarding. Note that there is a security risk involved with enabling this option; see section 9.5 for details.

4.20.6 'Allow attempted changes of username in SSH-2'

In the SSH-1 protocol, it is impossible to change username after failing to authenticate. So if you mis-type your username at the PuTTY 'login as:' prompt, you will not be able to change it except by restarting PuTTY.

The SSH-2 protocol *does* allow changes of username, in principle, but does not make it mandatory for SSH-2 servers to accept them. In particular, OpenSSH does not accept a change of username; once you have sent one username, it will reject attempts to try to authenticate as another user. (Depending on the version of OpenSSH, it may quietly return failure for all login attempts, or it may send an error message.)

For this reason, PuTTY will by default not prompt you for your username more than once, in case the server complains. If you know your server can cope with it, you can enable the 'Allow attempted changes of username' option to modify PuTTY's behaviour.

4.20.7 'Private key file for authentication'

This box is where you enter the name of your private key file if you are using public key authentication. See chapter 8 for information about public key authentication in SSH.

This key must be in PuTTY's native format (`*.PPK`). If you have a private key in another format that you want to use with PuTTY, see section 8.2.12.

If a key file is specified here, and Pageant is running (see chapter 9), PuTTY will first try asking Pageant to authenticate with that key, and ignore any other keys Pageant may have. If that fails, PuTTY will ask for a passphrase as normal.

4.21 The TTY panel

The TTY panel lets you configure the remote pseudo-terminal.

4.21.1 'Don't allocate a pseudo-terminal'

When connecting to a Unix system, most interactive shell sessions are run in a *pseudo-terminal*, which allows the Unix system to pretend it's talking to a real physical terminal device but allows the SSH server to catch all the data coming from that fake device and send it back to the client.

Occasionally you might find you have a need to run a session *not* in a pseudo-terminal. In PuTTY, this is generally only useful for very specialist purposes; although in Plink (see chapter 7) it is the usual way of working.

4.21.2 Sending terminal modes

The SSH protocol allows the client to send 'terminal modes' for the remote pseudo-terminal. These usually control the server's expectation of the local terminal's behaviour.

If your server does not have sensible defaults for these modes, you may find that changing them here helps. If you don't understand any of this, it's safe to leave these settings alone.

(None of these settings will have any effect if no pseudo-terminal is requested or allocated.)

You can add or modify a mode by selecting it from the drop-down list, choosing whether it's set automatically or to a specific value with the radio buttons and edit box, and hitting 'Add'. A mode (or several) can be removed from the list by selecting them and hitting 'Remove'. The effect of the mode list is as follows:

- If a mode is not on the list, it will not be specified to the server under any circumstances.
- If a mode is on the list:
  - If the 'Auto' option is selected, the PuTTY tools will decide whether to specify that mode to the server, and if so, will send a sensible value.

    PuTTY proper will send modes that it has an opinion on (currently only the code for the Backspace key, `ERASE`). Plink on Unix will propagate appropriate modes from the local terminal, if any.

  - If a value is specified, it will be sent to the server under all circumstances. The precise syntax of the value box depends on the mode.

By default, all of the available modes are listed as 'Auto', which should do the right thing in most circumstances.

The precise effect of each setting, if any, is up to the server. Their names come from POSIX and other Unix systems, and they are most likely to have a useful effect on such systems. (These are the same settings that can usually be changed using the `stty` command once logged in to such servers.)

Some notable modes are described below; for fuller explanations, see your server documentation.

- `ERASE` is the character that when typed by the user will delete one space to the left. When set to 'Auto' (the default setting), this follows the setting of the local Backspace key in PuTTY (see section 4.4.1).

  This and other special characters are specified using `^C` notation for Ctrl-C, and so on. Use `^<27>` or `^<0x1B>` to specify a character numerically, and `^~` to get a literal `^`. Other non-control characters are denoted by themselves. Leaving the box entirely blank indicates that *no* character should be assigned to the specified function, although this may not be supported by all servers.

- `QUIT` is a special character that usually forcefully ends the current process on the server (`SIGQUIT`). On many servers its default setting is Ctrl-backslash (`^\`), which is easy to accidentally invoke on many keyboards. If this is getting in your way, you may want to change it to another character or turn it off entirely.
- Boolean modes such as `ECHO` and `ICANON` can be specified in PuTTY in a variety of ways, such as `true`/`false`, `yes`/`no`, and `0`/`1`.
- Terminal speeds are configured elsewhere; see section 4.14.3.

4.22 The X11 panel

The X11 panel allows you to configure forwarding of X11 over an SSH connection.

If your server lets you run X Window System applications, X11 forwarding allows you to securely give those applications access to a local X display on your PC.

To enable X11 forwarding, check the 'Enable X11 forwarding' box. If your X display is somewhere unusual, you will need to enter its location in the 'X display location' box; if this is left blank, PuTTY will try to find a sensible default in the environment, or use the primary local display (`:0`) if that fails.

See section 3.4 for more information about X11 forwarding.

4.22.1 Remote X11 authentication

If you are using X11 forwarding, the virtual X server created on the SSH server machine will be protected by authorisation data. This data is invented, and checked, by PuTTY.

The usual authorisation method used for this is called `MIT-MAGIC-COOKIE-1`. This is a simple password-style protocol: the X client sends some cookie data to the server, and the server checks that it matches the real cookie. The cookie data is sent over an unencrypted X11 connection; so if you allow a client on a third machine to access the virtual X server, then the cookie will be sent in the clear.

PuTTY offers the alternative protocol `XDM-AUTHORIZATION-1`. This is a cryptographically authenticated protocol: the data sent by the X client is different every time, and it depends on the IP address and port of the client's end of the connection and is also stamped with the current time. So an eavesdropper who captures an `XDM-AUTHORIZATION-1` string cannot immediately re-use it for their own X connection.

PuTTY's support for `XDM-AUTHORIZATION-1` is a somewhat experimental feature, and may encounter several problems:

- Some X clients probably do not even support `XDM-AUTHORIZATION-1`, so they will not know what to do with the data PuTTY has provided.
- This authentication mechanism will only work in SSH-2. In SSH-1, the SSH server does not tell the client the source address of a forwarded connection in a machine-readable format, so it's impossible to verify the `XDM-AUTHORIZATION-1` data.
- You may find this feature causes problems with some SSH servers, which will not clean up `XDM-AUTHORIZATION-1` data after a session, so that if you then connect to the same server using a client which only does `MIT-MAGIC-COOKIE-1` and are allocated the same remote display number, you might find that out-of-date authentication data is still present on your server and your X connections fail.

PuTTY's default is `MIT-MAGIC-COOKIE-1`. If you change it, you should be sure you know what you're doing.

4.23 The Tunnels panel

The Tunnels panel allows you to configure tunnelling of arbitrary connection types through an SSH connection.

Port forwarding allows you to tunnel other types of network connection down an SSH session. See section 3.5 for a general discussion of port forwarding and how it works.

The port forwarding section in the Tunnels panel shows a list of all the port forwardings that PuTTY will try to set up when it connects to the server. By default no port forwardings are set up, so this list is empty.

To add a port forwarding:

- Set one of the ‘Local’ or ‘Remote’ radio buttons, depending on whether you want to forward a local port to a remote destination (‘Local’) or forward a remote port to a local destination (‘Remote’). Alternatively, select ‘Dynamic’ if you want PuTTY to provide a local SOCKS 4/4A/5 proxy on a local port (note that this proxy only supports TCP connections; the SSH protocol does not support forwarding UDP).
- Enter a source port number into the ‘Source port’ box. For local forwardings, PuTTY will listen on this port of your PC. For remote forwardings, your SSH server will listen on this port of the remote machine. Note that most servers will not allow you to listen on port numbers less than 1024.
- If you have selected ‘Local’ or ‘Remote’ (this step is not needed with ‘Dynamic’), enter a hostname and port number separated by a colon, in the ‘Destination’ box. Connections received on the source port will be directed to this destination. For example, to connect to a POP-3 server, you might enter `popserver.example.com:110`.
- Click the ‘Add’ button. Your forwarding details should appear in the list box.

To remove a port forwarding, simply select its details in the list box, and click the ‘Remove’ button.

In the ‘Source port’ box, you can also optionally enter an IP address to listen on, by specifying (for instance) `127.0.0.5:79`. See section 3.5 for more information on how this works and its restrictions.

In place of port numbers, you can enter service names, if they are known to the local system. For instance, in the ‘Destination’ box, you could enter `popserver.example.com:pop3`.

You can modify the currently active set of port forwardings in mid-session using ‘Change Settings’ (see section 3.1.3.4). If you delete a local or dynamic port forwarding in mid-session, PuTTY will stop listening for connections on that port, so it can be re-used by another program. If you delete a remote port forwarding, note that:

- The SSH-1 protocol contains no mechanism for asking the server to stop listening on a remote port.
- The SSH-2 protocol does contain such a mechanism, but not all SSH servers support it. (In particular, OpenSSH does not support it in any version earlier than 3.9.)

If you ask to delete a remote port forwarding and PuTTY cannot make the server actually stop listening on the port, it will instead just start refusing incoming connections on that port. Therefore, although the port cannot be reused by another program, you can at least be reasonably sure that server-side programs can no longer access the service at your end of the port forwarding.

If you delete a forwarding, any existing connections established using that forwarding remain open. Similarly, changes to global settings such as 'Local ports accept connections from other hosts' only take effect on new forwardings.

4.23.1 Controlling the visibility of forwarded ports

The source port for a forwarded connection usually does not accept connections from any machine except the SSH client or server machine itself (for local and remote forwardings respectively). There are controls in the Tunnels panel to change this:

- The 'Local ports accept connections from other hosts' option allows you to set up local-to-remote port forwardings in such a way that machines other than your client PC can connect to the forwarded port. (This also applies to dynamic SOCKS forwarding.)
- The 'Remote ports do the same' option does the same thing for remote-to-local port forwardings (so that machines other than the SSH server machine can connect to the forwarded port.) Note that this feature is only available in the SSH-2 protocol, and not all SSH-2 servers support it (OpenSSH 3.0 does not, for example).

4.23.2 Selecting Internet protocol version for forwarded ports

This switch allows you to select a specific Internet protocol (IPv4 or IPv6) for the local end of a forwarded port. By default, it is set on 'Auto', which means that:

- for a local-to-remote port forwarding, PuTTY will listen for incoming connections in both IPv4 and (if available) IPv6
- for a remote-to-local port forwarding, PuTTY will choose a sensible protocol for the outgoing connection.

Note that some operating systems may listen for incoming connections in IPv4 even if you specifically asked for IPv6, because their IPv4 and IPv6 protocol stacks are linked together. Apparently Linux does this, and Windows does not. So if you're running PuTTY on Windows and you tick 'IPv6' for a local or dynamic port forwarding, it will *only* be usable by connecting to it using IPv6; whereas if you do the same on Linux, you can also use it with IPv4. However, ticking 'Auto' should always give you a port which you can connect to using either protocol.

4.24 The Bugs panel

Not all SSH servers work properly. Various existing servers have bugs in them, which can make it impossible for a client to talk to them unless it knows about the bug and works around it.

Since most servers announce their software version number at the beginning of the SSH connection, PuTTY will attempt to detect which bugs it can expect to see in the server and automatically enable workarounds. However, sometimes it will make mistakes; if the server has been deliberately configured to conceal its version number, or if the server is a version which PuTTY's bug database does not know about, then PuTTY will not know what bugs to expect.

The Bugs panel allows you to manually configure the bugs PuTTY expects to see in the server. Each bug can be configured in three states:

- 'Off': PuTTY will assume the server does not have the bug.

- ‘On’: PuTTY will assume the server *does* have the bug.
- ‘Auto’: PuTTY will use the server's version number announcement to try to guess whether or not the server has the bug.

4.24.1 ‘Chokes on SSH-1 ignore messages’

An ignore message (SSH_MSG_IGNORE) is a message in the SSH protocol which can be sent from the client to the server, or from the server to the client, at any time. Either side is required to ignore the message whenever it receives it. PuTTY uses ignore messages to hide the password packet in SSH-1, so that a listener cannot tell the length of the user's password; it also uses ignore messages for connection keepalives (see section 4.13.1).

If this bug is detected, PuTTY will stop using ignore messages. This means that keepalives will stop working, and PuTTY will have to fall back to a secondary defence against SSH-1 password-length eavesdropping. See section 4.24.2. If this bug is enabled when talking to a correct server, the session will succeed, but keepalives will not work and the session might be more vulnerable to eavesdroppers than it could be.

This is an SSH-1-specific bug. No known SSH-2 server fails to deal with SSH-2 ignore messages.

4.24.2 ‘Refuses all SSH-1 password camouflage’

When talking to an SSH-1 server which cannot deal with ignore messages (see section 4.24.1), PuTTY will attempt to disguise the length of the user's password by sending additional padding *within* the password packet. This is technically a violation of the SSH-1 specification, and so PuTTY will only do it when it cannot use standards-compliant ignore messages as camouflage. In this sense, for a server to refuse to accept a padded password packet is not really a bug, but it does make life inconvenient if the server can also not handle ignore messages.

If this ‘bug’ is detected, PuTTY will assume that neither ignore messages nor padding are acceptable, and that it thus has no choice but to send the user's password with no form of camouflage, so that an eavesdropping user will be easily able to find out the exact length of the password. If this bug is enabled when talking to a correct server, the session will succeed, but will be more vulnerable to eavesdroppers than it could be.

This is an SSH-1-specific bug. SSH-2 is secure against this type of attack.

4.24.3 ‘Chokes on SSH-1 RSA authentication’

Some SSH-1 servers cannot deal with RSA authentication messages at all. If Pageant is running and contains any SSH-1 keys, PuTTY will normally automatically try RSA authentication before falling back to passwords, so these servers will crash when they see the RSA attempt.

If this bug is detected, PuTTY will go straight to password authentication. If this bug is enabled when talking to a correct server, the session will succeed, but of course RSA authentication will be impossible.

This is an SSH-1-specific bug.

4.24.4 ‘Miscomputes SSH-2 HMAC keys’

Versions 2.3.0 and below of the SSH server software from `ssh.com` compute the keys for their HMAC message authentication codes incorrectly. A typical symptom of this problem is that PuTTY dies unexpectedly at the beginning of the session, saying ‘Incorrect MAC received on packet’.

If this bug is detected, PuTTY will compute its HMAC keys in the same way as the buggy server, so that communication will still be possible. If this bug is enabled when talking to a correct server, communication will fail.

This is an SSH-2-specific bug.

4.24.5 'Miscomputes SSH-2 encryption keys'

Versions below 2.0.11 of the SSH server software from `ssh.com` compute the keys for the session encryption incorrectly. This problem can cause various error messages, such as 'Incoming packet was garbled on decryption', or possibly even 'Out of memory'.

If this bug is detected, PuTTY will compute its encryption keys in the same way as the buggy server, so that communication will still be possible. If this bug is enabled when talking to a correct server, communication will fail.

This is an SSH-2-specific bug.

4.24.6 'Requires padding on SSH-2 RSA signatures'

Versions below 3.3 of OpenSSH require SSH-2 RSA signatures to be padded with zero bytes to the same length as the RSA key modulus. The SSH-2 specification says that an unpadded signature MUST be accepted, so this is a bug. A typical symptom of this problem is that PuTTY mysteriously fails RSA authentication once in every few hundred attempts, and falls back to passwords.

If this bug is detected, PuTTY will pad its signatures in the way OpenSSH expects. If this bug is enabled when talking to a correct server, it is likely that no damage will be done, since correct servers usually still accept padded signatures because they're used to talking to OpenSSH.

This is an SSH-2-specific bug.

4.24.7 'Misuses the session ID in SSH-2 PK auth'

Versions below 2.3 of OpenSSH require SSH-2 public-key authentication to be done slightly differently: the data to be signed by the client contains the session ID formatted in a different way. If public-key authentication mysteriously does not work but the Event Log (see section 3.1.3.1) thinks it has successfully sent a signature, it might be worth enabling the workaround for this bug to see if it helps.

If this bug is detected, PuTTY will sign data in the way OpenSSH expects. If this bug is enabled when talking to a correct server, SSH-2 public-key authentication will fail.

This is an SSH-2-specific bug.

4.24.8 'Handles SSH-2 key re-exchange badly'

Some SSH servers cannot cope with repeat key exchange at all, and will ignore attempts by the client to start one. Since PuTTY pauses the session while performing a repeat key exchange, the effect of this would be to cause the session to hang after an hour (unless you have your rekey timeout set differently; see section 4.19.2 for more about rekeys). Other, very old, SSH servers handle repeat key exchange even more badly, and disconnect upon receiving a repeat key exchange request.

If this bug is detected, PuTTY will never initiate a repeat key exchange. If this bug is enabled when talking to a correct server, the session should still function, but may be less secure than you would expect.

This is an SSH-2-specific bug.

4.25 The Serial panel

The Serial panel allows you to configure options that only apply when PuTTY is connecting to a local serial line.

4.25.1 Selecting a serial line to connect to

The 'Serial line to connect to' box allows you to choose which serial line you want PuTTY to talk to, if your computer has more than one serial port.

On Windows, the first serial line is called `COM1`, and if there is a second it is called `COM2`, and so on.

This configuration setting is also visible on the Session panel, where it replaces the 'Host Name' box (see section 4.1.1) if the connection type is set to 'Serial'.

4.25.2 Selecting the speed of your serial line

The 'Speed' box allows you to choose the speed (or 'baud rate') at which to talk to the serial line. Typical values might be 9600, 19200, 38400 or 57600. Which one you need will depend on the device at the other end of the serial cable; consult the manual for that device if you are in doubt.

This configuration setting is also visible on the Session panel, where it replaces the 'Port' box (see section 4.1.1) if the connection type is set to 'Serial'.

4.25.3 Selecting the number of data bits

The 'Data bits' box allows you to choose how many data bits are transmitted in each byte sent or received through the serial line. Typical values are 7 or 8.

4.25.4 Selecting the number of stop bits

The 'Stop bits' box allows you to choose how many stop bits are used in the serial line protocol. Typical values are 1, 1.5 or 2.

4.25.5 Selecting the serial parity checking scheme

The 'Parity' box allows you to choose what type of parity checking is used on the serial line. The settings are:

- 'None': no parity bit is sent at all.
- 'Odd': an extra parity bit is sent alongside each byte, and arranged so that the total number of 1 bits is odd.
- 'Even': an extra parity bit is sent alongside each byte, and arranged so that the total number of 1 bits is even.
- 'Mark': an extra parity bit is sent alongside each byte, and always set to 1.
- 'Space': an extra parity bit is sent alongside each byte, and always set to 0.

4.25.6 Selecting the serial flow control scheme

The 'Flow control' box allows you to choose what type of flow control checking is used on the serial line. The settings are:

- 'None': no flow control is done. Data may be lost if either side attempts to send faster than the serial line permits.
- 'XON/XOFF': flow control is done by sending XON and XOFF characters within the data stream.
- 'RTS/CTS': flow control is done using the RTS and CTS wires on the serial line.
- 'DSR/DTR': flow control is done using the DSR and DTR wires on the serial line.

4.26 Storing configuration in a file

PuTTY does not currently support storing its configuration in a file instead of the Registry. However, you can work around this with a couple of batch files.

You will need a file called (say) `PUTTY.BAT` which imports the contents of a file into the Registry, then runs PuTTY, exports the contents of the Registry back into the file, and deletes the Registry entries. This can all be done using the Regedit command line options, so it's all automatic. Here is what you need in `PUTTY.BAT`:

```
@ECHO OFF
regedit /s putty.reg
regedit /s puttyrnd.reg
start /w putty.exe
regedit /ea new.reg HKEY_CURRENT_USER\Software\SimonTatham\PuTTY
copy new.reg putty.reg
del new.reg
regedit /s puttydel.reg
```

This batch file needs two auxiliary files: `PUTTYRND.REG` which sets up an initial safe location for the `PUTTY.RND` random seed file, and `PUTTYDEL.REG` which destroys everything in the Registry once it's been successfully saved back to the file.

Here is `PUTTYDEL.REG`:

```
REGEDIT4

[-HKEY_CURRENT_USER\Software\SimonTatham\PuTTY]
```

Here is an example `PUTTYRND.REG` file:

```
REGEDIT4

[HKEY_CURRENT_USER\Software\SimonTatham\PuTTY]
"RandSeedFile"="a:\\putty.rnd"
```

You should replace `a:\putty.rnd` with the location where you want to store your random number data. If the aim is to carry around PuTTY and its settings on one floppy, you probably want to store it on the floppy.

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

[PuTTY release 0.60]