PUTTY *DRIVER*

INTRODUCTION

PuttyDriver aims to assist systems administration and legacy application testing (e.g., regression testing) using automation and robotics via an easy to use Secure Shell Protocol (SSH) interface. Planned future versions of PuttyDriver, include Windows Form applications (Visual Basic and C#) for integration of legacy applications with modern digital systems.

PuttyDriver has been built using Excel and Visual Basic and interfaces with the popular open source PuTTY SSH and Telnet client developed by Simon Tatham and others.

PuttyDriver is free and open source software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Putty is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

The project files with source code can be found at https://github.com/christyler80/PuttyDriver.

PuttyDriver includes both record and replay functionality and uses the Microsoft Windows Message Queue to communicate interactively with one or more PuTTY sessions.

PuttyDriver currently consists of two application files - **PuttyDriver.xlsb** (Excel 32 bit) and **putty.exe,** together with a SQLite database. ODBC driver for SQLite must be installed and can be downloaded from here.

putty.exe is a slightly modified 0.81 version of PuTTY with a small amount of additional C code, to provide the message queue interface.

So far, the project scope has been 'proof of concept' for building and testing a robust communication interface with PuTTY.

Basic testing has successfully been undertaken with servers running a variety of Unix/Linux platforms. A sample script for a standard Raspberry Pi OS installation is included and is referenced in this document.

Please also see DB Excel Data Manager for additional database interface documentation

Important: At this time:

- PuttyDriver has only been tested with 32-bit versions of Microsoft Excel 2016 or later.
- Microsoft Excel 64-bit versions are not currently supported.
- 32-bit or 64-bit versions of Microsoft Windows 10 operating system or later are supported.
- Other operating systems (e.g., Apple, Linux) are not supported.

NEXT STEPS

Q2 2025 - .NET Windows Form application for integration of legacy applications with modern digital systems, via easy to use automation and robotics.
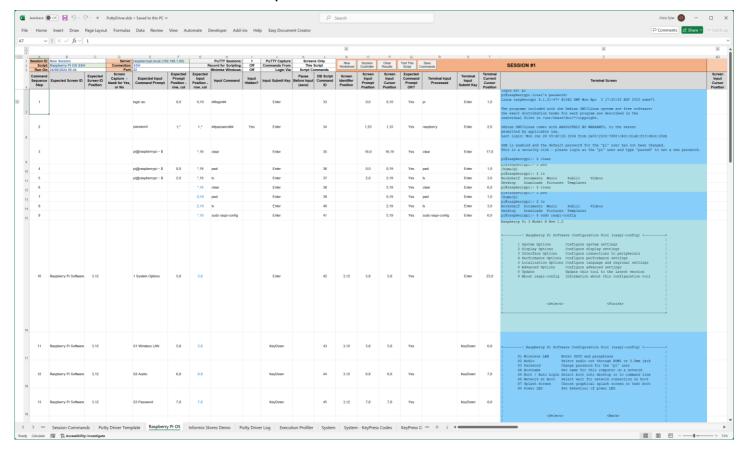
Please provide feedback via GitHub and/or by contacting me using the details below.

Chris Tyler, June 2024

chris@christyler.uk

www.christyler.uk

1. Download the Zip file from [GitHub](#) and extract into a folder.
2. Run putty.exe and connect to a server using SSH (or Telnet), using its IP address or Server Name.
3. Check that login and password are both OK and close putty.exe.
4. Open the PuttyDriver.xlsb workbook and using the 'Example Template' worksheet, type the **IP address** or **Server Network Name** into cell E1. Leave the **Script Name** in cell B2 blank.

5. Press the 'Record New Script' button. PuTTY (**putty.exe**) should open.
6. Login into PuTTY as usual, i.e., using a valid user ID and password.
7. These commands and the PuTTY screen should be captured automatically, in rows 6 and 7 of the spreadsheet.
8. Type 2 or 3 additional commands Into PuTTY - e.g., '*pwd*' or '*ls*'. These should also be captured in the spreadsheet.
9. When finished, close PuTTY as normal.
10. Passwords are usually often 'hidden'. If so, type '*Yes*' Into the '*Input Hidden?*' column 'G' on row 7 if not already present.
11. To replay these commands, press the 'Clear Results' button, followed by the 'Test This Script' button.
12. As before, PuTTY (putty.exe) should open but this time, PuttyDriver should log in automatically using the credentials used in step 6 and automatically run the commands typed in step 8.
13. The PuTTY screen(s) should be captured automatically, both in Excel and as text files, in a new 'Capture' folder.
14. Script execution can be controlled using the '*Expected Screen ID*', '*Expected Screen ID Position*', '*Expected Input Command Prompt*', '*Expected Cursor Position*' and '*Expected Cursor Position*' columns.
15. See the 'Raspberry PI OS' worksheet for examples of how the '*Expected*' fields can be set and use of '*' wildcard for Cursor position where X or Y are known, but not both (e.g., after running list files command).
16. After recording a new script, Use the 'Save Script Commands' function to save the script commands for future use.
17. Use the 'Session Controller' to manage servers and to access/re-run existing scripts.
18. Scripts can be also modified and changes saved using these worksheets (e.g., add screen identification text/numbers).