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 .NET controller programs (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.

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 0.78 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

Q3 2023 - Full Documentation.

Q4 2023 - VB.NET and C# programs 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.

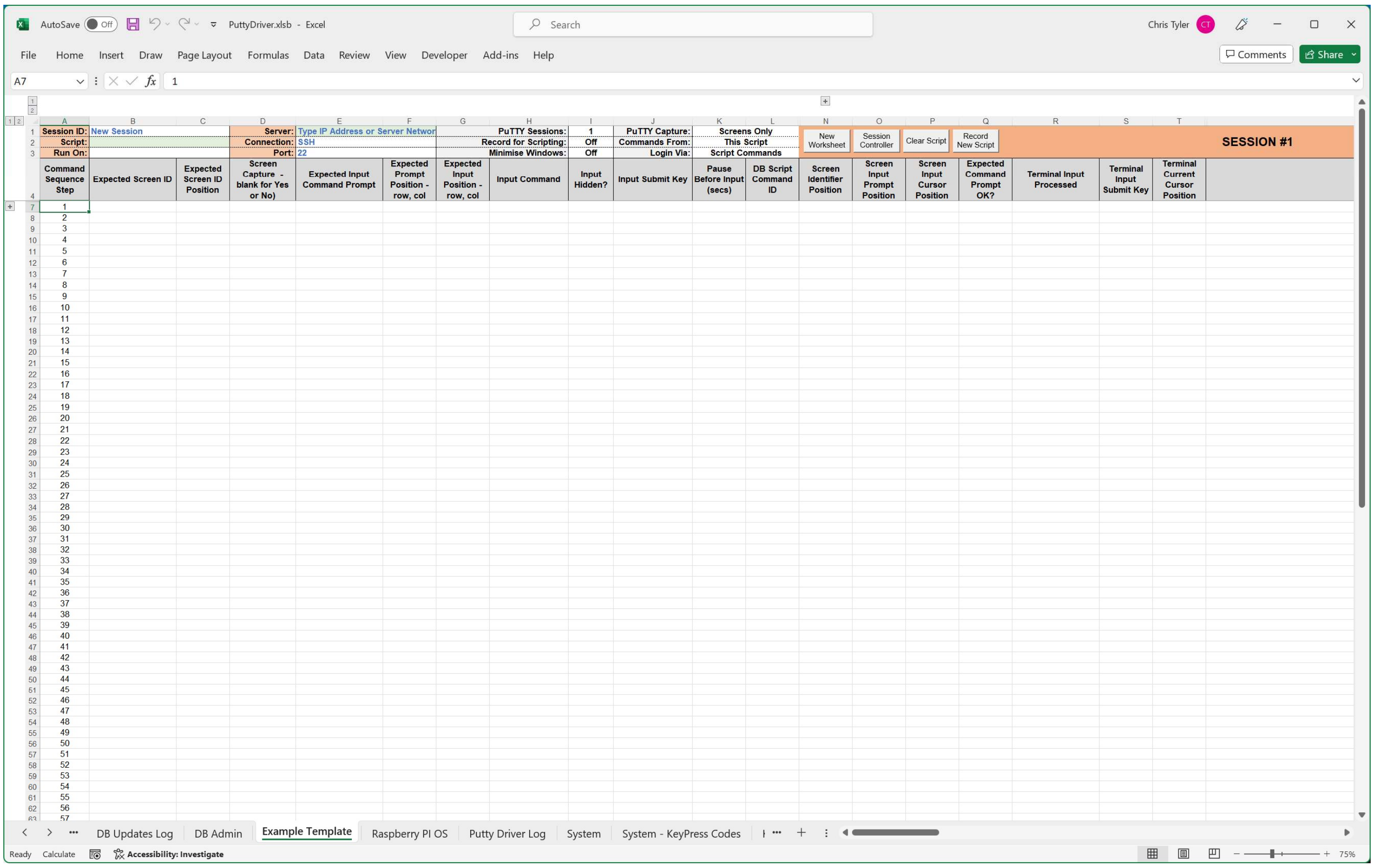
Chris Tyler, July 2023

chris@christyler.uk

www.christyler.uk

GETTING STARTED

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.



Session ID:	New Session	Server:	Type IP Address or Server Networ	PuTTY Sessions:	1	PuTTY Capture:	Screens Only
Script:		Connection:	SSH	Record for Scripting:	Off	Commands From:	This Script
Run On:		Port:	22	Minimise Windows:	Off	Login Via:	Script Commands

Command Sequence Step	Expected Screen ID	Expected Screen ID Position	Screen Capture - blank for Yes or No)	Expected Input Command Prompt	Expected Prompt Position - row, col	Expected Input Position - row, col	Input Command	Input Hidden?	Input Submit Key	Pause Before Input (secs)	DB Script Command ID	Screen Identifier Position	Screen Input Prompt Position	Screen Input Cursor Position	Expected Command Prompt OK?	Terminal Input Processed	Terminal Input Submit Key	Terminal Current Cursor Position
1																		
2																		
3																		
4																		
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		
19																		
20																		
21																		
22																		
23																		
24																		
25																		
26																		
27																		
28																		
29																		
30																		
31																		
32																		
33																		
34																		
35																		
36																		
37																		
38																		
39																		
40																		
41																		
42																		
43																		
44																		
45																		
46																		
47																		
48																		
49																		
50																		
51																		
52																		
53																		
54																		
55																		
56																		
57																		

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 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 Input Command Prompt*' 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).