VNCDoTool Documentation

Release 0.9.0.dev0

Marc Sibson

March 03, 2016

Contents

CHAPTER 1

Introduction

It's under active development and seems to be working, but please report any problems you have.

1.1 Quick Start

To use vncdotool you will need a VNC server. Most virtualization products include one, or use RealVNC, TightVNC or clone your Desktop using x11vnc.

Once, you have a server running you can install vncdotool from pypi:

```
pip install vncdotool
```

and then send a message to the vncserver with:

```
vncdo -s vncserveraddress type "hello world"
```

You can also take a screen capture with:

```
vncdo -s vncservername capture screen.png
```

More documentation can be found on Read the Docs.

1.2 Feedback

Comments, suggestions and patches are welcome and appreciated. They can be sent to via GitHub, vncdotool@googlegroups.com or sibson+vncdotool@gmail.com.

If you are reporting a bug or issue please include the version of both vncdotool and the VNC server you are using it with.

1.3 Acknowledgements

Thanks to Chris Liechti, techtonik and Todd Whiteman for developing the RFB and DES implementations used by vncdotool. Also, to the TigerVNC project for creating a community focus RFB specification document

CHAPTER 2

Installation

vncdotool is available on PyPI, so in most cases you should be able to simply run:

```
pip install vncdotool
```

vncdotool relies on a number of libraries, the two major ones are PIL, the Python Imaging Library and Twisted, an asynchronous networking library. While vncdotool should work with any recent version of these libraries sometimes things break. If you are having issues getting things to work you can try using a stable set of libraries and if you aren't already using it, and you should be, use a virtualenv.:

```
pip install virtualenv
virtualenv venv-vncdotool
# XXX requirements.txt from vncdotool source tree
pip install -r requirements.txt
pip install -e .
```

2.1 Windows

If you are not familiar with Python, the most reliable way to install vncdotool is to use binary packages. Currently, (Oct 2013) PIL only provides 32bit binary packages for Windows, so you will need to install a 32bit python.

1. Download and install python-2.7.5.exe from the Python Downloads website
2. Open up Powershell, and paste in the following:

```
    [Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\Python27\;C:\Python27\Scripts\", "Us
```

3. Restart your Windows Machine
4. Upon Restart, go to the Twisted Downloads and get and install 32bit Twisted, Twisted-13.1.0.win32-py2.7.exe
5. Download and install PIL-1.1.7.win32-py2.7.exe, from PIL Downloads.
6. Download ez_setup.py and get_pip.py and save them to your *Python/Scripts* folder, `C:\Python27\Scripts`
7. Open up Powershell and type the following:

```
    pip install pip --upgrade
    pip install distribute
    pip install setuptools --upgrade
    pip install Twisted --upgrade
    pip install vncdotool < -- Finally install vncdotool
```

8. At a Powershell prompt:

```
vncdo.exe --server som.eip.add.res type "Hello World"
```

9. If Hello World shows up on the remote machine that has a VNC server running then its time to celebrate. Otherwise, first check you can connect from your local machine to the remote using a normal GUI VNC Client. Once you get the normal GUI client working try vncdotool again and if you still have problems contact sibson+vncdotool@gmail.com.

CHAPTER 3

Usage

3.1 Basic Usage

Once installed you can use the vncdotool command to send key-presses. Alphanumerics are straightforward just specify the character. For other keys longer names are used:

```
> vncdo key a
> vncdo key 5
> vncdo key .
> vncdo key enter
> vncdo key shift-a
> vncdo key ctrl-C
> vncdo key ctrl-alt-del
```

To type longer strings when entering data or commands you can use the type c command, which does not support special characters:

```
> vncdo type "hello world"
```

You can control the mouse pointer with move and click commands. NOTE, you should almost always issue a move before a click, as in:

```
> vncdo move 100 100 click 1
```

The following would seem to be equivalent but would actually click at (0, 0). This occurs due to how click events are encoded by VNC, meaning you need to initialise the position of the mouse.:

```
> vncdo move 100 100
> vncdo click 1
```

If you have the Python Imaging Library (Pillow) installed you can also make screen captures of the session:

```
> vncdo capture screenshot.png
```

With Pillow installed, you can wait for the screen to match a known image:

```
> vncdo expect somescreen.png 0
```

Putting it all together you can specify multiple actions on a single command line. You could automate a login with the following:

```
> vncdo type username key enter expect password_prompt.png
> vncdo type password move 100 150 click 1 expect welcome_screen.png
```

Sometimes you only care about a portion of the screen, in which case you can use rcapture and rexpect. For instance, if your login window appears at x=100, y=200 and is 400 pixels wide by 250 high you could do:

```
> vncdo rcapture region.png 100 200 400 250
> vncdo rexpect region.png 100 200 0
```

3.2 Running Scripts

For more complex automation you can read commands from stdin or a file. The file format is simply a collection of actions:

```
> echo "type hello" | vncdo -
```

Or if you had a file called login.vdo with the following content:

```
# select the name text box, enter your name and submit
move 100 100 click 1 type "my name" key tab key enter

# grab the result
capture screenshot.png
```

You could run it with the following command:

```
> vncdo login.vdo
```

3.3 Creating Scripts

While you can create scripts by hand it can often be a time consuming process. To make the process easier vncdotool provides a log mode that allows a user to record a VNC session to a script which is playable by vncdo. vnclog act as a man-in-the-middle to record the VNC commands you issue with a client. So you will have your vnclog connect to your server and your viewer connect to vnclog

vncviewer —> vnclog —> vncserver

For best results be sure to set your vncviewer client to use the RAW encoding. Others encoding may work but are not fully supported at this time.

The quickest way to get started is to run:

```
> vnclog --viewer vncviewer keylog.vdo
```

For more control you can launch the viewer seperately but be sure to connect to the correct ports:

```
> vnclog keylog.vdo
> vncviewer localhost:2  # do something and then exit viewer
> vncdo keylog.vdo
```

By running with –forever vnclog will create a new file for every client connection and record each clients activity. This can be useful for quickly recording a number of testcases.:

```
> vnclog --forever --listen 6000 /tmp
> vncviewer localhost::6000
# do some stuff then exit and start new session
> vncviewer localhost::6000
# do some other stuff
> ls /tmp/*.vdo
```

CHAPTER 4

Embedding in Python Applications

vncdotool is built with the Twisted framework, as such it best intergrates with other Twisted Applications Rewriting your application to use Twisted may not be an option, so vncdotool provides a compatability layer. It uses a seperate thread to run the Twisted reactor and communitcates with the main program using a threadsafe Queue.

To use the syncronous API you can do the following:

```
from vncdotool import api
client = api.connect('vnchost:display')
```

You can then call any of the methods available on *vncdotool.client.VNCDoToolClient* and they will block until completion. For example:

```
client.captureScreen('screenshot.png')
client.keyPress('enter')
client.expectScreen('login_success.png', maxrms=10)
```

This can be used to automate the starting of an Virtual Machine or other application:

```
vmtool.start('myvirtualmachine.img')
client.connect('vmaddress::5950')
client.expectScreen('booted.png')
for k in 'username':
    client.keyPress(k)
client.keyPress('enter')
for k in 'password':
    client.keyPress(k)
client.keyPress('enter')
client.expectScreen('loggedin.png')

# continue with your testing session or other work
```

CHAPTER 5

Command Reference

5.1 click BUTTON

5.2 capture FILENAME.PNG

5.3 drag X Y

5.4 expect FILENAME.PNG FUZZ

5.5 key KEY

5.6 keydown KEY

5.7 keyup KEY

5.8 move X Y

5.9 mousedown BUTTON

5.10 mousemove X Y

5.11 mouseup BUTTON

5.12 pause SECONDS

5.13 rcapture FILENAME.PNG X Y W H

5.14 rexpect FILENAME.PNG X Y FUZZ

5.15 type STRING

CHAPTER 6

Release History

6.1 0.10.0 (2016-03-03)

- drop offical 2.6 support, it'll probably work for a while still
- use frombytes rather than fromstring for compatibility with PIL
- vnclog works with password protected servers using –password-required
- exit more reliably after an error
- use increatmental frameBufferUpdateRequests, appears to be compatible with more servers
- include basic version negotiation with servers, thanks Ezra Bühler

6.2 0.9.0 (2015-05-08)

- add special keys [~!@#$%^&*()_+{}|:"<>?] to –force-caps, for servers that don't handle them, Tyler Oderkirk, Aragats Amirkhanyan
- improve vnclog performance with TCP_NODELAY, Ian Britten
- by default pause 10ms between sending commands, better compatability with servers
- better handle screen resizing, Daniel Stelter-Gliese
- API, fix deadlocks due to threaded init of PIL, thanks Antti Kervinen
- API, support password protected server, thanks Antti Kervinen
- API, able to connect to multiple servers, Daniel Stelter-Gliese
- drop official support for py2.4 and py2.5
- use Pillow rather than PIL

Thanks to Jan Sedlák, Daniel Stelter-Gliese, Antti Kervinen, Anatoly Techtonik, Tyler Oderkirk and Aragats Amirkhanyan for helping make this release possible

6.3 0.8.0 (2013-08-06)

- improved documentation using sphinx

- regional capture and expect that operate on a portion of the display
- –force-caps, better compatibility when sending UPPERCASE to servers
- –timeout, exit with an error after a given number of seconds
- experimental syncronous API for easier intergration with non-Twisted apps

6.4 0.3.0 (2012-12-22)

- main program renamed to vncdo, vncdotool continues an alias for now
- use host:display, host::port syntax like other vnc tools, removed -d
- read/play commands from stdin or file
- vnclog, creates scripts from captured interactive sessions
- better control over mouse in screen captures with –nocursor and –localcursor
- mousemove, sleep command aliases to match xdotool
- keyup/keydown commands for more control over keypresses
- send SetEncodings on connect, thanks Matias Suarez for fix
- debian packaging
- type “Hello World” now preserves capitalization
- basic compatibility with VNC 4.0 servers, found in some KVMs
- improved frameUpdate handling
- –warp to replay script faster than real-time
- –delay, insert a delay between sending commands

6.5 0.2.0 (2012-08-07)

- add pause, mouseup, mousedown, drag commands
- only require TWisted 11.1.0, so we can have py2.4 support
- **bugfixes, thanks Christopher Holm for reporting**
  - vncdotool type -something now works
  - no longer silently fail for unsupported image formats

6.6 0.1.1 (2011-05-18)

- add PIL to requires
- fix bug where incorrect mouse button is sent

6.7 0.1.0 (2011-03-03)

- first release
- commands: press, type, move, click, capture, expect

CHAPTER 7

Contributing

Code and Issue tracking is provided by Github. There is also a mailing list setup via Google Groups.

CHAPTER 8

Code Documentation

8.1 client Module

Twisted based VNC client protocol and factory

3. 2010 Marc Sibson

MIT License

exception vncdotool.client.**AuthenticationError**
Bases: exceptions.Exception

VNC Server requires Authentication

class vncdotool.client.**VNCDoToolClient**
Bases: vncdotool.rfb.RFBClient

SPECIAL_KEYS_US = '~!@#$%^&*()_+{}|:"<>?'

bell()

buttons = 0

captureRegion(*filename*, *x*, *y*, *w*, *h*)
Save a region of the current display to filename

captureScreen(*filename*)
Save the current display to filename

cmask = None

commitUpdate(*rectangles*)

connectionMade()

copy_text(*text*)

cursor = None

deferred = None

drawCursor()

expectRegion(*filename*, *x*, *y*, *maxrms=0*)
Wait until a portion of the screen matches the target image

The region compared is defined by the box (x, y), (x + image.width, y + image.height)

expectScreen(*filename*, *maxrms=0*)
Wait until the display matches a target image

filename: an image file to read and compare against maxrms: the maximum root mean square between histograms of the

screen and target image

keyDown(*key*)

keyPress(*key*)
Send a key press to the server

key: string: either [a-z] or a from KEYMAP

keyUp(*key*)

mouseDown(*button*)
Send a mouse button down at the last set position

button: int: [1-n]

mouseDrag(*x*, *y*, *step=1*)
Move the mouse point to position (x, y) in increments of step

mouseMove(*x*, *y*)
Move the mouse pointer to position (x, y)

mousePress(*button*)
Send a mouse click at the last set position

button: int: [1-n]

mouseUp(*button*)
Send mouse button released at the last set position

button: int: [1-n]

paste(*message*)

pause(*duration*)

screen = None

updateCursor(*x*, *y*, *width*, *height*, *image*, *mask*)

updateRectangle(*x*, *y*, *width*, *height*, *data*)

vncConnectionMade()

vncRequestPassword()

x = 0

y = 0

class vncdotool.client.**VNCDoToolFactory**
Bases: vncdotool.rfb.RFBFactory

clientConnectionFailed(*connector*, *reason*)

clientConnectionLost(*connector*, *reason*)

clientConnectionMade(*protocol*)

force_caps = False

nocursor = False

password = None

protocol
alias of *VNCDoToolClient*

pseudocursor = False

shared = True

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

V

A

B

C

D

E

F

K

M

N

P

S

U

V

X

Y