
VNCDoTool Documentation

Release 0.8.0

Marc Sibson

August 06, 2013

CONTENTS

1	Introduction	1
1.1	Quick Start	1
1.2	Feedback	1
1.3	Acknowledgements	1
2	Installation	3
3	Usage	5
3.1	Basic Usage	5
3.2	Running Scripts	6
3.3	Creating Scripts	6
4	Library API	7
5	Release History	9
5.1	0.8.0 (2013-08-06)	9
5.2	0.3.0 (2012-12-22)	9
5.3	0.2.0 (2012-08-07)	10
5.4	0.1.1 (2011-05-18)	10
5.5	0.1.0 (2011-03-03)	10
6	Code Documentation	11
6.1	<code>client</code> Module	11
7	Indices and tables	13
	Python Module Index	15

INTRODUCTION

vncdotool is a command line VNC client. It can be useful to automating interactions with virtual machines or hardware devices that are otherwise difficult to control.

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, you can use RealVNC, TightVNC or clone you 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 at [ReadtheDocs](#).

1.2 Feedback

Comments, suggestions and patches are welcome and appreciated. They can be sent to sibson+vncdotool@gmail.com or via [GitHub](#). 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

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 .
```


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 ([PIL](#)) installed you can also make screen captures of the session:

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

With [PIL](#) 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
```

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.

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.:

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

If its too hard to remember which port to use you can tell vncdotool to launch a viewer that will already be connected to the vnclog session.:

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

By running with `--forever` vncdotool 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
```

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
```

LIBRARY API

As `vncdotool` is built on the `Twisted` framework it best intergrates into other Twisted Applications That isn't always an option so a synchronous API is under development. It uses a seperate thread to run the Twisted reactor and communicates with the main program using a threadsafe Queue.

To use the synchronous 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:123')
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
```


RELEASE HISTORY

5.1 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 synchronous API for easier intergration with non-Twisted apps

5.2 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

5.3 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

5.4 0.1.1 (2011-05-18)

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

5.5 0.1.0 (2011-03-03)

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

CODE DOCUMENTATION

6.1 client Module

Twisted based VNC client protocol and factory

3. 2010 Marc Sibson

MIT License

`vncdotool.client.ImageFactory()`

Wrap importing PIL.Image so vncdotool can be used without PIL being installed. Of course capture and expect won't work but at least we can still offer key, type, press and move.

class `vncdotool.client.VNCDoToolClient`

Bases: `vncdotool.rfb.RFBClient`

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*)

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*)

clientConnectionMade (*protocol*)

force_caps = False

nocursor = False

password = None

protocol

alias of `VNCDoToolClient`

pseudocursor = False

shared = True

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

V

`vncdotool.client`, [11](#)