Kostis Netzwerkberatung
Talstr. 25, D-63322 Rödermark
Tel. +49 6074 881056
http://www.kostis.net/
kosta@kostis.net

HowTo
do-device.py

created by

Kostis Netzwerkberatung
Konstantinos Kostis

do-device.py HowTo

Document Details

Title:	do-device.py..HowTo
Filename:	do-device.py.HowTo.doc
Author:	Konstantinos Kostis, kosta@kostis.net
Manager:	
Version:	V1.01
Date:	18.08.2020

Audience:

`do-device` script framework users

History (CTRL + click on page number to access text mark)

Page	Date	Editor	Reason
All	2018-09-27	kosta@kostis.net	removed “location”
All	2018-06-25	kosta@kostis.net	corrected typos
All	2017-08-02	kosta@kostis.net	initial documentation

do-device.py HowTo

Table of Contents (CTRL + click to access chapter)

1 Preface

You can use `do-device.py` (and scripts included) free of charge at your own risk.

This document describes how to use `do-device.py` allowing easy automated show commands for (Cisco) network devices. Log files created by `do-device.py` can easily be analysed.

`do-device.py` takes care of logging on to a device using `ssh` (via `netmiko`) and then executes a CLI script written by the user. User credentials must be stored in a personal "`credentials.txt`" file. The environment variable "`DO_DEVICE`" must point to the directory containing "`credentials.txt`" and "`devicelist.txt`".

For every device and script, a log file is created. User CLI scripts may contain comments.

`do-device.py` originally was written in Python 3 (3.6.1) and uses the Python module `netmiko` (containing `cryptography.fernet`).

If you haven't already, install `netmiko` using `pip`:

```
pip install netmiko
```

`do-device.py` was created using Microsoft Windows. It should however work fine with Linux and the like as well.

Prerequisites
`do-device.py` requires Python 3 (3.6.1 or better is recommended) and the Python `netmiko` module.

https://www.python.org/downloads/

1.1 Personal Configuration Files

Personal configuration files should be located in a directory and the environment variable `DO_DEVICE` should contain that directory path. If you don't already have such a directory consider creating it like this:

```
REM Windows
c:
cd \
mkdir "%LOCALAPPDATA%\Kostis Netzwerkberatung"
mkdir "%LOCALAPPDATA%\Kostis Netzwerkberatung\do-device"
```

Set environment variable `DO_DEVICE` to

```
%LOCALAPPDATA%\Kostis Netzwerkberatung\do-device
```

Expand `%LOCALAPPDATA%` before including it to `DO_DEVICE`. Create the files below there.

1.2 credentials.txt

Create a file `credentials.txt`.

```
################################################################################
# %DO_DEVICE%\credentials.txt example
################################################################################
#
# realm;username;password;secret
#
################################################################################
# realm=* means default realm, all realms must start with '*'
# secret=* means same as password
# secret=- means no secret (no enable)
# password or secret starting with " " is encrypted
################################################################################

*;username;encrypted-password;*
```

Replace "`username`" by the account name used to log on to devices.

Replace "`encrypted-password`" by your account password used to login to devices encrypted using `pwencrypt.py`.

If `enable` is not the same as the `encrypted-password`, replace `*` by the enable secret encrypted using `pwencrypt.py`.

Make sure to update this file every time after changing your device account passwords!

1.3 deviceinfo.txt

`deviceinfo.txt` contains information about the device OS and credentials and may also be used to map hostnames to IP addresses if there is no DNS resolution of said hostnames and also no permission to modify your `hosts` file.

```
################################################################################
# %DO_DEVICE%\deviceinfo.txt example
################################################################################
#
# device;ipaddr;type;username;password;secret
#
################################################################################
#
# ipaddr is optional (when there's no hosts/DNS available)
# type is a netmiko type (e.g. cisco_ios, cisco_nxos)
# username/password/secret=* means default (realm=*) value from credentials.txt
# *realm means realm (not default) value from credentials.txt
# otherwise specify different username/password/secret for device
# password/secret must be encrypted
# secret=- means no secret (no enable)
#
################################################################################

hostname1;;cisco_nxos;*;*;*
hostname2;;cisco_nxos;*;*;*
hostname3;;cisco_nxos;*;*;*
hostname4;;cisco_nxos;*;*;*
hostname5;;cisco_nxos;*;*;*
hostname6;;cisco_nxos;*;*;*
hostname7;;cisco_nxos;*;*;*
hostname8;;cisco_nxos;*;*;*
hostname9;;cisco_nxos;*;*;*
```

If no `ipaddr` is given, DNS/hosts will have to be available for the given `hostname`.
`type` (such as "`cisco nxos`") are defined by the Python `netmiko` module.
'`*`' for `username`, `password`, `secret` means use the values from "`credentials.txt`".
You can use a different `username`, `password`, `secret` for each device…

2 Script File Location

Put the following files in a directory included in your `PATH` environment variable:

- `do-devoce.py`
- `pwdecrypt.py`
- `pwdecrypt.py`

If you don't have such a directory, place the files in a directory of your choice and add that directory path to your `PATH` environment variable.

2.1 CLI Script File Syntax

CLI script files can contain any (Cisco) CLI command returning a prompt ("`>`" or "`#`").

CLI script files can contain (Cisco) comment lines (starting with "`!`").
Comment lines will not be sent to the device.

If you need to pass time between CLI lines, use the "`!!sleep n`" command (`n` is the number of seconds to wait).

Before your CLI script files are executed `do-device.py` performs a login via `ssh` using the user credentials given in "`credentials.txt`" or "`devicelist.txt`".

Your CLI script file is executed line by line waiting for a prompt after each line. Waiting for a prompt is handled by the Python module `netmiko`. It may not always be 100% reliable.

2.1.1 Example CLI script-status

```
terminal length 0
show interface trunk
show interface status
show interface description
show port-channel summary
show mac address-table
show clock
```

2.1.2 Example CLI script-show-config

```
terminal length 0
show run
show clock
```

2.2 do-devices Limitations/Caveats

This limitation may hit you arguably with any command line automation.
Device output containing either "`#`" or "`>`" is "cut" at the first occurrence of "`#`" or "`>`".

Never use any prompt character in any (Cisco) configuration!

Using either "`#`" or "`>`" you will end up in a lot of trouble not just with the `do-devices` script framework but with products such as CiscoWorks and DeviceExpert as well.
Please also refrain from using the character "$" if possible.

2.3 Syntax

2.3.1 do-device.py

Run a given script on a given device.

Syntax:

```
do-device.py device script
```

`do-device.py` logs in to device executing `script`. A log file is created.

2.3.2 pwencrypt.py

Encrypt a password read from the console for use in "`credentials.txt`" and "`devicelist.txt`".

Syntax:

```
pwencrypt.py
```

2.3.3 pwdecrypt.py

`pwdecrypt.py` is a module imported by `do-device.py`. It is not meant to be run directly.

3 Appendix A: On Password Encryption

You may ask yourself: why encrypt a password? Or if you're beyond this, how do I make sure the decrypted password cannot be seen by anyone else?

3.1 Why Encrypt A Password?

The main reason why passwords should never be stored in clear text in configuration files, permanent scripts or anywhere else is that people might find those files and use the credentials in there to do harm. Even if they don't get access to the file directly they might look over your shoulder and see passwords while you display or edit them in your scripts.

If you think that's paranoid, think again. It happens all the time and unless you can be really sure nobody will be ever be able to look at your screen, your devices may not be safe.

3.2 What to Avoid Using Decodable Passwords

One mistake people tend to make is to make files containing password information readable by more than the owner. Make sure your file permissions are such that nobody else can read your files if at all possible.

Another problem is that if you extract a password and then use it as a parameter for a script or program the clear text password is shown in the process list. Never use clear text passwords in command line parameters.

3.3 Make Decodable Passwords More Safe

Use a mechanism that looks up the credentials in your scripts. Decrypt them inside your script (use the Force of Python, Luke).