

DJHELLSTEPS

VERSION 1.01

User Manual

Jerome DESMOULINS - January 13, 2018

Contents

1	Introduction	4
1.1	Features	4
1.2	Requirements	5
1.3	How it works	5
1.4	Supported Operating systems	6
2	Installation	6
3	Configuration	6
3.1	DjeShellSteps configuration file	6
3.2	DjeShellSteps directories	7
4	Your first scripts	8
4.1	Example 1: First script with 3 steps	8
5	DjeShellSteps Internal Variables	10
5.1	Specify ShellName to have a particular log file name	10
5.2	Help message for command line	10
6	Command line arguments	11
6.1	Activate debug mode (debug)	12
6.2	Restart from a particular step (fromstep)	12
6.3	Command line help (help)	12
6.4	Restart from scratch (fromscratch)	12
6.5	Display last log file contents (lastlog)	13
6.6	Terminate a running script (kill)	13
6.7	Force output to stdout (stdout)	13
6.8	Force summary output to stdout (stdoutsummary)	13
6.9	List all Shell Steps (liststeps)	14
7	Configuration file for your shell script	14
8	Methods of work	15
8.1	Exiting DjeShellSteps before the end of the script	15
9	DjeShellSteps error codes	15
10	User Manual update history	16

1 Introduction

DjeShellSteps is a bash shell library, allowing to create shell scripts with steps, to simplify and improve your exploitation scripts.

If the program detect a problem during execution, script will stop, giving a returned code and an error message.

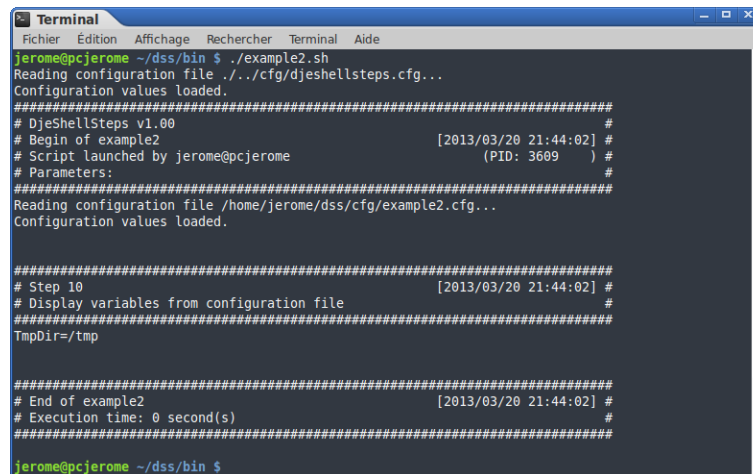
When a user restart it, program, will start from last error step, and continue its work.

All script execution will generate a log file, with all outputs from commands (catching stdout and stderr).

DjeShellSteps is very useful to automate some process, and secure shell scripts. Each shell can be restarted at last error step, from a particular step, or from beginning. Full log can be displayed to stdout in realtime, or hidden. A live summary of shell script execution is also available.

I've started to work on DjeShellSteps for my needs, on my website's exploitation scripts, in 2007. As I've seen a lot of same needs in forums or internet, I've decided to publish it, as an OpenSource product.

This manual works with DjeShellSteps version 1.01.



```

Terminal
Fichier Edition Affichage Rechercher Terminal Aide
jerome@pcjerome ~/dss/bin $ ./example2.sh
Reading configuration file ../cfg/djeshellsteps.cfg...
Configuration values loaded.
#####
# DjeShellSteps v1.00
# Begin of example2 [2013/03/20 21:44:02] #
# Script launched by jerome@pcjerome (PID: 3609 ) #
# Parameters:
#####
Reading configuration file /home/jerome/dss/cfg/example2.cfg...
Configuration values loaded.

#####
# Step 10 [2013/03/20 21:44:02] #
# Display variables from configuration file
#####
TmpDir=/tmp

#####
# End of example2 [2013/03/20 21:44:02] #
# Execution time: 0 second(s)
#####
jerome@pcjerome ~/dss/bin $

```

You can find more informations on DjeShellScripts website:
<http://www.desmoulins.fr/projets/djeshellsteps/>

1.1 Features

Here is a list of DjeShellSteps main features:

- All you Shell scripts can be manage in a single tree
- Common command line for your shell scripts
- Allow to execute a shell script step by step. In case of problem, the script can be restarted after last successful step

- Easy restart method, script can be restarted from beginning or a specific step
- Easy way to find last errors
- All informations written to stdout are displayed to a log file
- Only a short summary of script execution can be shown on stdout (or nothing)
- Last execution log can be displayed from sample command
- Script can be killed by a simple command
- You can check the script status in real time (via command line)
- Easy configuration file processing
- No need to take care about debugging, DjeShellSteps will do it for you
- Display all your script steps from command line

1.2 Requirements

DjeShellSteps require an operating system with Bash command line interpreter installed. Some basic OS tools are used by DjeShellSteps . Your system must have *bash* and *tee* command available.

1.3 How it works

When you launch a shell script, DjeShellSteps will

- Initiate a log file with the name of your script and datetime
- Catch all outputs into logfile and stdout (by default)
- Create a pid file during all script execution
- Create a rst file during all script execution
- If no problem happens, pid, and rst file will be removed after execution
- If a problem occurs, pid will be removed, your log file will be renamed with *.E*. rst file will stay, to allow you a rerun from last failed step.

1.4 Supported Operating systems

DjeShellSteps works with

- AIX
- BSD (OpenBSD, FreeBSD)
- Linux
- Solaris
- Windows (using Babun for example <http://babun.github.io/>)

2 Installation

To install DjeShellSteps you have to uncompress the .tar.gz file in the directory of your choice. Example, to install DjeShellSteps on /opt/exploitation directory:

```
mkdir /opt/exploitation
cd /opt/exploitation
gzip -d /tmp/djeshellsteps.tar.gz
tar xvf /tmp/djeshellsteps.tar
rm /tmp/djeshellsteps.tar
```

The following tree has been created:

```
opt
├── exploitation
│   ├── bin
│   │   ├── djeshellsteps.sh
│   │   └── example1.sh
│   ├── cfg
│   │   ├── djeshellsteps.cfg.default .3 data
│   ├── doc
│   ├── logs
│   └── tmp
```

DjeShellSteps is now installed. You have to read next chapter, to configure it.

3 Configuration

3.1 DjeShellSteps configuration file

djeshellsteps.cfg file contains all DjeShellSteps configuration. By default, this file is located in the *cfg* directory. *bin* directory can be used. To setup the file for first time, simply copy the file *djeshellsteps.cfg.default* to *djeshellsteps.cfg*

```
cd /opt/exploitation/cfg
copy djeshellsteps.cfg.default djeshellsteps.cfg
```

Default configuration file

```
#####
# File:      djeshellsteps.cfg                #
# Version:   1.00                            #
# Description:                                #
# Configuration file for DjeShellSteps        #
#####

#Specify bellow all the directories for DjeShellTools
BaseDir=/opt/exploitation                    # Update with your installation path
ShDir=$BaseDir/bin
LogDir=$BaseDir/logs
DataDir=$BaseDir/datas
SQLDir=$BaseDir/sql
IncludeDir=$BaseDir/bin
CfgDir=$BaseDir/cfg

### DjeShellSteps Options ###
DJSOption_StdoutEnabled=1                   # 0: Write only in logfile, 1: Write all informat
DJSOption_SummaryInfosToStdout=0           # 0: Do nothing, 1: Write summary informations to
```

Write log contents into Stdout (DJSOption_StdoutEnabled)

This variable can be set to 0, or 1.

Value	Description
0	Nothing displayed on Stdout. Logfile is only used.
1	All output informations are displayed into logfile. and StdOut

Write summary informations to Stdout (DJSOption_SummaryInfosToStdout)

This variable can be set to 0, or 1.

Value	Description
0	No summary informations displayed into stdout.
1	Summary informations are written into stdout

This option is used if you don't want all log contents info stdout, but only some short informations to know when a new step is starting, if program failed, or has been restarted, ...

3.2 DjeShellSteps directories

bin

Contains DjeShellSteps and your shell scripts

cfg

Contains DjeShellSteps configuration file, and the one for your shell scripts

data

Can be used to store your scripts data

doc

DjeShellSteps manual

logs

Log files for all your shells scripts

tmp

Can be used to store your temporary data (for your scripts)

4 Your first scripts

This chapter contains simple scripts, to explain DjeShellSteps programming method

4.1 Example 1: First script with 3 steps

Script explanation

Here is a first example of DjeShellSteps which is just a simple script with three steps.

To start a script, we must start starting bash as a Shell engine for our script

```
#!/bin/bash
```

Then, best way to remember what the script is and who do what, we can put a banner with our informations

```
#####
# File: example1.sh                                     #
# Author: Jerome DESMOULINS - djeshellsteps@desmoulins.fr #
# Description:                                         #
#   DjeShellSteps example 1: sample shell script      #
#   with 3 steps.                                     #
#####
```

Each script must start calling DjeShellSteps main library, using:

```
. `dirname $0`/djeshellsteps.sh
```

This call will load all DjeShellSteps variables and procedures in memory, allowing to initialise steps.

One library loaded, DjeShellSteps must be initialized, using:

BeginSteps

This will initiate DjeShellSteps , create the log file, pid file, restart file (.rst), and start tracking commands

We can go now with our steps. To declare a step, we just include it with a number:

```
Step 10 "List all files on current directory"
```

Then, we have to check if this step has to be launched (depending on restart file, or parameters provided from command line):

```
if [ $RunThisStep -eq 1 ]; then
```

Here we are. From here, we can put all the commands we want to launch, loops, variable declaration, It's a shell script here. For this example, we will just list the files in the current directory:

```
ls
```

Then, we have to end the step, closing if statement:

```
fi
```

The step is now defined. We can define more steps if needed. After the last step, nothing has to be done. You can save your script, give it the execution rights, and launch it.

The full script source

Create a file called example1.sh, in DjeShellSteps bin directory:

```
#!/bin/bash
#####
# File: example1.sh #
# Author: Jerome DESMOULINS - djeshellsteps@desmoulins.fr #
# Description: #
# DjeShellSteps example 1: sample shell script #
# with 3 steps. #
#####

### Calling DjeShellSteps ###
. `dirname $0`/djeshellsteps.sh
BeginSteps

#####
Step 10 "List all files on current directory"
#####
if [ $RunThisStep -eq 1 ]; then
ls
```

```

fi

#####
Step 20 "Sleep for two seconds"
#####
if [ $RunThisStep -eq 1 ]; then
    sleep 2
fi

#####
Step 30 "Display kernel version"
#####
if [ $RunThisStep -eq 1 ]; then
    uname -a
fi

### End of Shell Script ###
EndSteps

```

Launching your script

To launch your script, you have to make your script executable

```
chmod 755 example1.sh
```

then, you just have to launch it:

```
./example1.sh
```

Your script will be launched. A log file has been generated in logs directory.

5 DjeShellSteps Internal Variables

5.1 Specify ShellName to have a particular log file name

DjeShellSteps allow to specify your ShellName, as a variable. If *ShellName* is set, your script will be called by this value, instead of default name (your shell, without extension). For example, you shell is called `apache_control.sh`, and you want to have a log file called `apache_.log`. You have to specify:

```
ShellName=apache
```

and DjeShellSteps will create logfiles as `apache_YYYYMMDD_HHMMSS.log`

5.2 Help message for command line

DjeShellSteps allow to manage command line help, using a very simple method. To add your help message (parameters) you just have to fill `$Help` variable in the beginning of your shell script. Example:

```
Help="<input>;Input file name|<output>;Output file name"
```

When you will launch your shell using `-help` parameter from command line, two new parameters will be displayed. `input` and `output`. Result:

```
$/my_script.sh --help
./my_script.sh                                     (powered by DjeShellSteps v1.01)
                                                    (c) Jerome DESMOULINS
```

```
Usage:
```

```
./my_script.sh <input> <output>
```

```
Where:
```

```
    <input>           Input file name
    <output>          Output file name
```

```
...
```

```
...
```

You can then control if two parameters has been specified on command line, using the following code:

```
Help="<input>;Input file name|<output>;Output file name"
```

```
### Calling DjeShellSteps ###
. `dirname $0`/djeshellsteps.sh
```

```
### Checking parameters ###
if [ $# -ne 2 ]
then
    ShowHelp
fi
...
```

In this case, if the script is not launched with two parameters on command line, it will stop and display the help message.

6 Command line arguments

This chapter explain all available command line arguments. You can use your own command line arguments, by the same way than you manage command line arguments in a shell script. DjeShellSteps add common command line arguments, allowing to interact with a DjeShellSteps script. Available arguments are:

- debug
- fromstep
- fromscratch
- help

- lastlog
- kill
- stdout
- stdoutsummary
- liststeps

6.1 Activate debug mode (debug)

debug parameter will activate debugging mode. With this mode, all commands/instructions from your shell script will be displayed in the log file. Example:

```
./myshell --debug
```

to activate debug mode

6.2 Restart from a particular step (fromstep)

fromstep parameter allow DjeShellSteps to restart from a specified step. Example:

```
./myshell --fromstep 20
```

to restart myshell from step 20

6.3 Command line help (help)

help parameter will display all command line options. Example:

```
./myshell --help
```

to display all available options (Common and the ones from your shell).

To add a help message in your shell, refer to: 5.2 Help message for command line.

6.4 Restart from scratch (fromscratch)

fromscratch parameter allow DjeShellSteps to restart from first step. Example:

```
./myshell --fromscratch
```

to restart myshell from first step.

This parameter is usefull if last execution of your script failed and you want to restart from beginning.

6.5 Display last log file contents (lastlog)

lastlog will not start your shell script. It will just display last execution log file, and restart file contents, if exists. Example:

```
./myshell --lastlog
```

to view last execution log file.

This parameter is usefull if last execution of your script failed and you want to display last log file.

6.6 Terminate a running script (kill)

kill option will terminate your script execution. It will be stopped, and a restart file will be created with the current step. When you will try to rerun your script, it will start on the last execution step. Example:

```
./myshell --kill
```

to terminate your script execution.

This parameter is usefull if your shell is stuck, or frozen (feature mainly used for develoment purpose).

6.7 Force output to stdout (stdout)

stdout option will force shell script to display all informations to stdout. Log file is written in anycase. Example:

```
./myshell --stdout
```

to force full stdout output (and keep log file).

This parameter is usefull when you want to launch a job manually if `DJSOption.StdoutEnabled` option is set to 0).

6.8 Force summary output to stdout (stdoutsummary)

stdoutsummary option will force shell script to display summary informations to stdout. This option should be used when `DJSOption.StdoutEnabled` option is set to 0, but you want to have minimal informations about your shell execution into stdout. Example:

```
./myshell --stdoutsummary
```

to force summary informations to stdout (and keep log file).

This parameter is usefull when you want to launch a job manually if `DJSOption.StdoutEnabled` option is set to 0).

6.9 List all Shell Steps (liststeps)

liststeps option will display all Shell steps. This option should be used by operator, to understand what the shell is doing in case of problems, to know what has been done during last execution, and what to be done when it will be restarted. Example:

```
./myshell --liststeps
```

to list all Shell steps.

This parameter is usefull in operational mode. Example:

```
./myshell --liststeps
```

Steps for myshell:

```
-Step 10:  Checking for new invoice file
-Step 20:  Converting file from CSV to JSON
-Step 30:  Moving converted file to destination folder
```

7 Configuration file for your shell script

Your shell script can use a configuration file. To use this feature, you have to create a configuration file in the configuration directory of *DjeShellSteps* . All variables in the configuration file can be read, and you can use them directly from your shell script.

To read your configuration file, you have to use *ReadConfigFile* function.

Example:

```
ReadConfigFile
```

to read your shell script configuration file content, and load variables into your shell. All configuration variable will be available after using *ReadConfigFile*

To read another configuration file than you shell, you can specify its name calling the function Example:

```
ReadConfigFile my_configuration_file.cfg
```

to read this particular configuration file content, and load variables into your shell.

Your can put some comments on your configuration file. All variables must be set with the folowing method:

```
YourVariable=VariableValue
```

Configuration file example:

```
#####
# File: example2.cfg #
# Author: Jerome DESMOULINS - djeshellsteps@desmoulins.fr #
# Description: #
```

```
#   DjeShellSteps example 2: sample configuration file   #
#####
TmpDir=/tmp
```

8 Methods of work

This chapter describes some methods of work with DjeShellSteps .

8.1 Exiting DjeShellSteps before the end of the script

Exiting and generate and error

To exit, and generate an error in DjeShellSteps , you just have to use system exit command. Example:

```
exit 1
```

Will end your shell script execution, generate a restart file in the current step. Your script will restart from this step in the next execution.

Exiting without error

To exit DjeShellSteps without generating an error, you can use the following code. Example:

```
# Force DjeShellSteps to terminate without playing next steps
EndSteps
exit 1
```

DjeShellSteps will end, without generating a restart file. Next execution will start from first step.

9 DjeShellSteps error codes

This chapter contains all DjeShellSteps internal error codes.

Error Code	Explanation
501	Failed to write restart file contents. DjeShellSteps failed to write contents into restart file. Check .rst file rights.
502	Failed to remove restart file. DjeShellSteps failed to remove .rst file. Check .rst file rights.
503	Failed to write log file. DjeShellSteps failed to write log file contents. Check log directory rights for user which is launching your shell script.
504	Another instance of your shell script is already running. Check PID in your .script.pid, to check. If nothing is running, and pid file still there, you can remove it (this should not happens as DjeShellSteps remove the pid file after shell script execution
505	Failed to write pid file. DjeShellSteps writes a file with the name of your script dot pid, to prevent many launches from the same script at a same time. DjeShellSteps was not able to write your .script.pid in the log directory. Check log directory rights.
506	Failed to remove pid file. DjeShellSteps was not able to remove your .script.pid in the log directory. Check log directory rights.

10 User Manual update history

Date	Update details
2018/01/13	Improve english translation of the manual