opConfig Configuration Guide (Version 1.x)

opConfig has 3 primary configuration files that make it run.

1. credential_sets.nmis
2. connections.nmis
3. command_sets.nmis

Below is a description of how to configure them and a little explanation of what they do.

Credentials (for connecting to devices)

conf/credential_sets.nmis holds the credential sets that are used when connecting to a device. Even if auto discovery is not use the credentials still need to live in this file.

To setup credentials edit conf/credential_sets.nmis

```
#add/remove/change the lines with default username/password info to match credentials for the devices you want
to discover
%hash = (
 'empty' => { username => '', password => '' },
 'myCredentialSetHere' => { username => 'YourUsername', password => 'YourPassword, password_privileged =>
'YourPassword_supersecret' }
);
```

Make very sure this file is only readable by you / root (opfixperms.pl will do this for you)

```
chmod 600 conf/credential_sets.nmis
```

Connections

Connections tell opConfig how to connect to the devices you would like to gather configuration data from. Connections can be auto-discovered if opConfig is attached to an NMIS configuration.

NB: opConfig will only attempt to discover devices from NMIS that are **active** and are currently being **collected. To force all NMIS devices into the list (even if they are not active) pass force_active=true**

```
bin/opconfig-cli.pl act=discover
```

This will attempt to use the credentials given in the credential_sets against every active & collected device in NMIS using different transport types (SSH and Telnet) and will output the commands it has found to connections.nmis. If a connection for a device already exists in connections.nmis opConfig will leave the settings as they are and not attempt to discover them. If your list of credentials is long and your list of devices is long I suggest not using this method as it will take many many cups of coffee to complete.

NOTE: to skip testing each credential set pass disable_test=true, if this is done the connections file will need to have the credential entries for each device set, and the transport value checked (as it's only a guess)

Here is a sample connections.nmis file, if you use auto discovery opConfig will produce output in this format (which is the required format):

```
%hash = (
 'asgard' => {
 'connection_info' => {
 'transport' => 'Telnet',
 'credential_set' => 'myCredentialSetHere',
 'personality' => 'ios',
 'node' => 'asgard',
 'host' => '192.168.88.254'
 },
 'os_info' => {
 'featureset' => 'Unknown',
 'version' => '12.4(25f)',
 'platform' => '1841',
 'train' => '12.4',
 'major' => '12.4',
 'os' => 'IOS',
 'image' => 'C1841-ADVENTERPRISEK9-M'
 }
 },
 'thor' => {
 'connection_info' => {
 'transport' => 'SSH',
 'credential_set' => 'set4',
 'personality' => 'bash',
 'node' => 'thor',
 'host' => '192.168.88.8',
 'priveleged_credential_set' => 'set3'
 },
 'os_info' => {
 'featureset' => 'N/A',
 'version' => '2.6.32-131.21.1.el6.x86_64',
 'platform' => 'x86_64',
 'train' => '2.6',
 'major' => '2.6',
 'os' => 'Linux',
 'image' => 'N/A'
 }
 }
);
```

The important settings here are in the connection_info.

NOTE: If the command_sets you want to run filter based on os_info then you will need to define the required data in order for opConfig-cli.pl to match the connections you require. At the very least os_info->os will need to be defined but for many devices it is likely you will want to define more than that so your command sets can target the device with better commands.

Command Sets

A default command_sets.nmis file is provided. It defines a list of "command sets" to be run on devices that match the criteria laid out by each specific command set (usually by the os_info hash inside the command set). As many sets as you like can be added, with as many commands as you like.

If you are running the lastest versions of IOS, at the time of writing 15.1 was new, you would need to modify the command set to include 15.1 in the version list, this could be done by changing

```
'version' => '/12.2|12.4|15.0/',
```

to

```
'version' => '/12.2|12.4|15.0|15.1/',
```

or

```
'version' => '/12.2|12.4|15.\d+/',
```

Changing it to 15.\d+ will match any version of IOS 15 from now on.

```
%hash = (
 'IOS_DAILY' => {
 'os_info' => {
 'version' => '/12.2|12.4|15.0/',
 'os' => 'IOS'
 },
 'aging_info' => {
 'age' => 'forever'
 },
 'scheduling_info' => {
 'run_commands_on_separate_connection' => 'false'
 },
 commands => [
 {
 'tags' => 'config,version,troubleshooting, detect-change',
 'command' => 'show version',
 'privileged' => 'false',
 'multipage' => 'true',
 'run_command_on_separate_connection' => 'false',
 'command_filters' => [
 '/uptime is/'
 ]
 }
}
);
```

A quick note, every opConfig try and bundle as many commands for the same device together into a single session (or connection if you like). If you would like the command set, or the individual command to be run on it's own connection (a good idea for long running commands), you can set run_commands_on_separate_connection => 'true' to run each command in that command set on it's own, or run_command_on_separate_connection => 'true' to run that specific command on it's own.

Running command sets

```
bin/opconfig-cli.pl act=run_command_sets
```

This command will run all command sets against all matching connections.

If you would like to run only specific command set/s: (comma separated, no spaces)

```
bin/opconfig-cli.pl act=run_command_sets names=IOS_DAILY,LINUX_DAILY
```

This will run only the IOS_DAILY and LINUX_DAILY command sets.

There is currently no way to run a command set against a specific connection, and no way to run only 1 specific command.

NOTE: Only nodes that are marked as "active" and "collect" in NMIS are run, to force them to run add 'force_active' => 'true' to the connection

If you would like to run it against only specific nodes: (comma separated, no spaces):

```
bin/opconfig-cli.pl act=run_command_sets nodes=node1,node2
```