ForEachBox 1.0.3

ForEachBox is a scripting and automation tool for various Cisco devices. It supports the command line interface of the Cisco Nexus, IOS and ASA series. It is implemented as a Java GUI and requires the Plink tool that is included in PuTTY.

It features command macros, which allow Expect-Send clauses and dynamic parameters in commands. Results can be saved to files or condensed in data collections.

Contact

Website: http://www.semken.com/projekte

Email: Volker@semken.com

Disclaimer:

ForEachBox is an automation tool, which can change settings of many devices at one click. Special care should be taken using this software and the test option be considered. I shall not be held responsible for any harm or damage caused by this software, including system or network outages, costs for restoring to a previous state or damage to the management systems. This software is provides as-is, use at your own risk.

The following is the legal warning quote from the PuTTY web site.

***LEGAL WARNING**: Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. I believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England and Wales and in many other countries, but I am not a lawyer and so if in doubt you should seek legal advice before downloading it. You may find this site useful (it's a survey of cryptography laws in many countries) but I can't vouch for its correctness.*

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Licenses

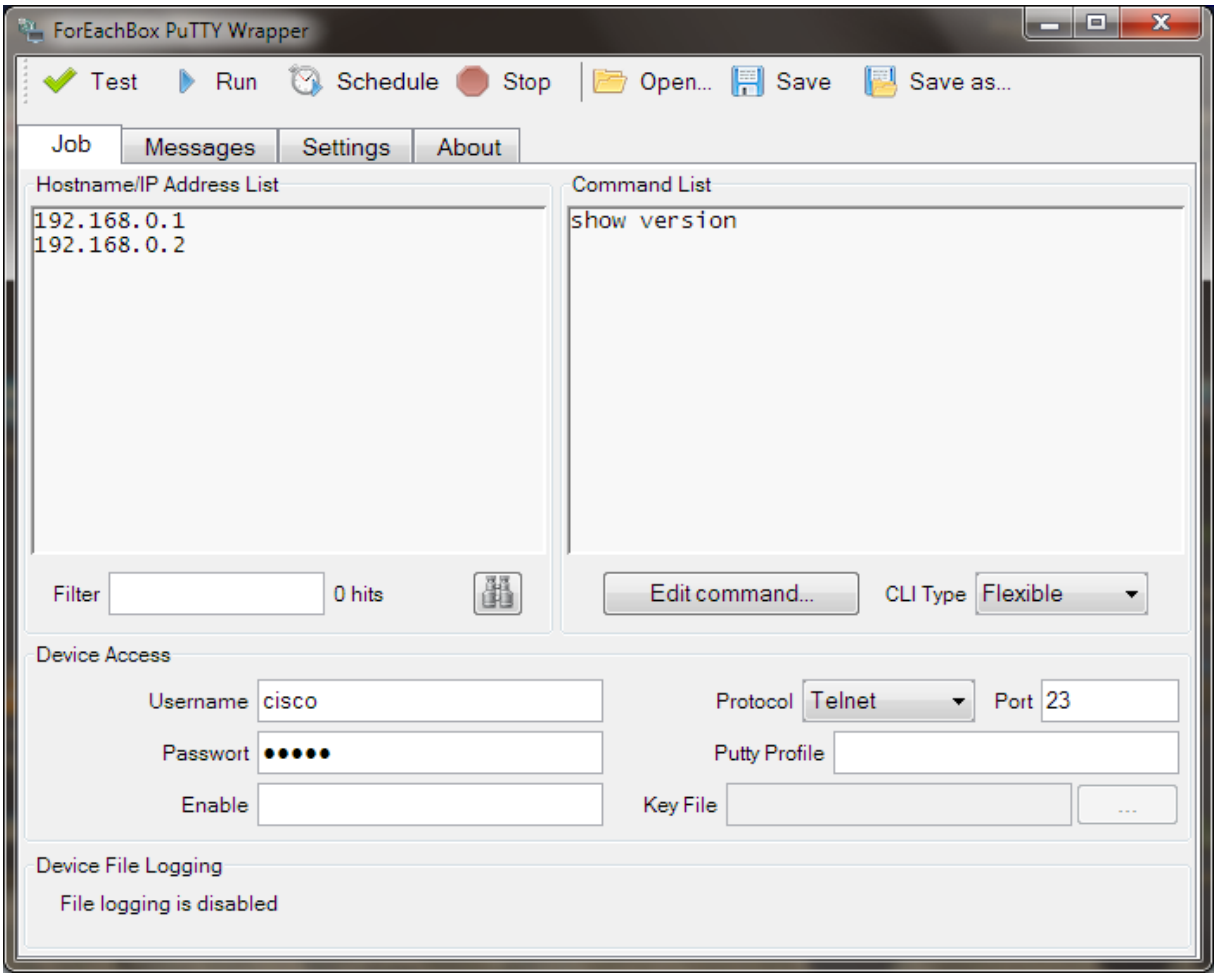
Security consideration

ForEachBox does not save any passwords, pins or the information about the private key file to a project file or any logging file. The username is only saved to the settings file, which is stored in the user directory.

Passwords and keys from the device output are not filtered and are saved to log files with any other information. The used file names are disclosed in this document. Besides the logging files, that are configured to be generated no additional temporary file is used. All information in the Messages log are keep in memory.

Passwords and Pins provided to the application are stored in memory. Attempts are undertaken to wipe these from memory after use or before the application closes. Garbage collection and code optimization can build copies of these memory areas, where it becomes difficult to ensure, that passwords are not retained in memory, even after the application is terminated.

For SSH authentication the password can optionally be included in the command line for Plink. Any other application with administrative rights can query this command line and gain access to the username and password used for authentication. This information is only available during the runtime or Plink.



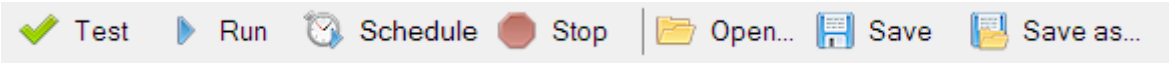
Installation

ForEachBox does not use a setup; just copy the two files ForEachBox.jar and Plink.exe to a working directory. All generated logging and projects files are also automatically stored in this directory.

ForEachBox requires Java JRE 1.4. It has been tested to work on Windows and Linux, but should work on any platform, where Java and Putty are working.

If you start ForEachBox the first time, it comes up with a few defaults. These settings are stored in the file *foreachbox.ini* in the home directory.

Buttons



Test Button

This will run the normal job on the first active device in the host list only. All Log files are generated the same way.

Run Button

This will start the processing of all devices included in the host list.

Schedule Button

Start the scheduler. Settings can be changed under the Settings tab.

Stop Button

This will allow stopping the job. On the first press, this will stop the process after the current device; the current device will be completed first. If the scheduler has been started, it will also be stopped.

The second press will immediately stop processing the current device. This will also happen on a test run.

Open Button

This opens a file open dialog to pick and load a previously saved job file.

Save Button

Saves the current settings to a previously opened or saved file. If no file has been selected before, the Save-As file dialog will allow saving the settings to a new file.

This stores most settings of the Job and Settings tab and also the messages filter to a project file. Usernames, passwords, the private key file and Plink path are not saved.

Save As Button

The current settings are saved to a new file.

Job definition

The tab Job allows defining all settings for one job, like the hostnames, the commands to execute and the device access.

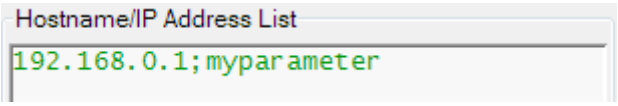
List of hostnames

This contains the hostnames or IP addresses, where ForEachBox should work on. Hostnames can contain domain suffixes, but they must be resolvable to IP addresses on the local system. Enter one system per line without any spaces or other characters.

Hosts can be disabled with a prepending "!" or "#", which are used as remark characters. The hostbox used the following color key.

Color	Usage
Black	Active hosts
Grey	Remark or host not matching filter
Blue	Current host and host matching host filter
Green	Host was finished successful
Orange	Host finished with warnings, not all commands could be delivered
Red	Connection or login failed

The host list can also contain a second parameter, separated with a semicolon. This parameter will be used to preload the variable, which then can be used in the macros. This allows using an individual setting for each host. White spaces ahd quote signs are supported.



A typical use is to use the host as a jump box and initiate a Telnet to the hostname in the variable parameter. It can also be used to use this value in the logging file name via the placeholder %variable%.

Filter Host List

This allows the list of hosts to be filtered. Only hosts that match the filter criteria will be processed. Hosts, that match will be colored black, hosts, that don't match will be colored grey.

This allows from a longer list of hosts to select only devices from a specific network or location or to select the test device. The filter can contain regular expression elements like "|", "?", "[1-3]" and others.

Find Host

Select a host in the host list and press the Find button  . It will select the corresponding begin of the messages for this host. If that host is in the messages more than once, the find button in the messages tab can be used to find the other sections for this host.

Commands

This contains all device specific commands to be sent to the device. This list allows the use of "#" as a remark and disables this line. The command can include the "|" to filter the output on the device and any regular expression can be appended. There is no syntax checking done on the commands, use care on entering the commands.

Please see the section *Command Macros* on using specific switches.

CLI Type

The type of the device or the CLI type it uses can be selected here. It is used to send specific commands to the device to initialize the session. Currently it will disable page breaks ("more" prompt), so that all output is delivered in one piece.

The option "Flexible" allows the command prompt to change or to access other devices during working on one host. Usually ForEachBox expects the prompt to begin with the hostname and end with an arrow, hash or dollar sign. With this option enabled, only the end section is checked. This also allows accessing other command line interfaces, like Linux servers.

Do not revert this command or select the wrong device type. ForEachBox will not handle these interruptions well and commands will timeout.

Username

Enter the username that should be used to authenticate with the device. SSH requires this, but can obtain this from a provided Putty profile. Telnet supports to authenticate using just a password. If the login doesn't work check the messages log for more information.

The username is stored in the programs setting files in the user's directory.

Password

This is the password used to authenticate with the device. It is not saved in the project or settings files. This field can be left empty if the devices don't require a password.

Enable

This is the enable password for the device. If left empty ForEachBox will not attempt to enter privileged mode. If it's set it will try to get into privilege level 15. If the useraccount already provides level 15 access, there is no change in providing the enable password. Some platforms don't support an enable password, like Nexus switches, where this command will be rejected. If a different procedure is required to gain privileges it can also be implemented as a command macro.

Protocol

This allows selecting Telnet, SSH version 1 and 2 from the list. For Telnet, it is possible to login just with the password and omitting the username. If "SSH" is selected, both versions are allowed and negotiated by Plink at session setup.

Port

Enter any non-default TCP port here. If the port is left empty, ForEachBox will use port 22 for SSH and 23 for Telnet.

Putty profile

This is a putty profile, which can contain other settings and parameters for the device access. Settings provided in ForEachBox have precedence. This can also be used to provide a username for a SSH session and is a method to use different usernames in job files.

Key file

Beginning with 15.0 Cisco routers support public key authentication. Provide the private key here to login to a device where the public key has been provided to. This can also work in conjunction with Putty's Pageant which provides key management.

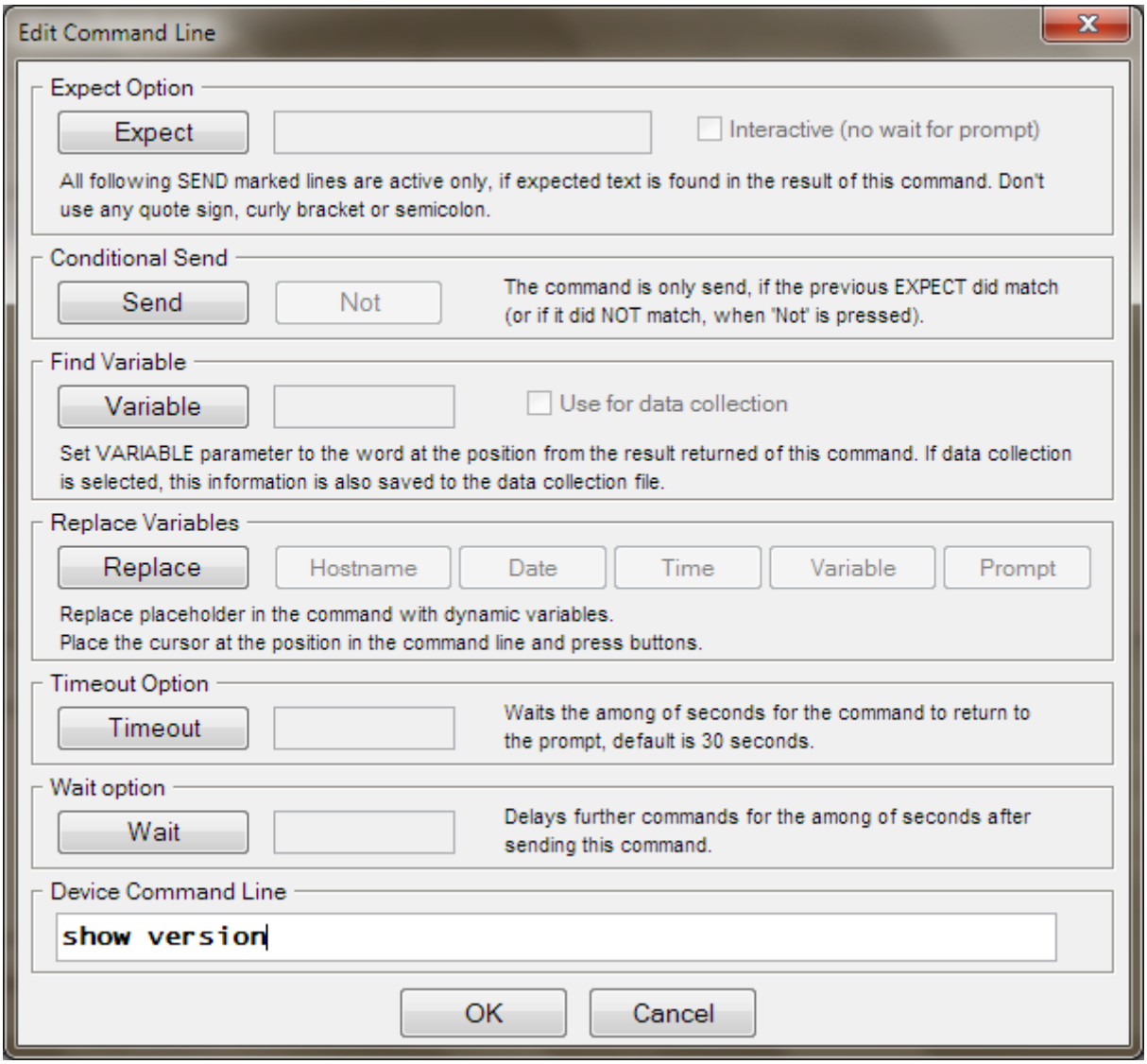
Command Macros

ForEachBox supports Command Macros, which allow commands to be sent conditionally, filled with dynamic information and to collect information. They are prepended to each command and evaluated while the commands are sent to the device. Macros are always applied to one command, but the result can be used in the scope of the individual device.

Macros have their own syntax, but are expressed as simple text. You can edit copy or delete them directly in the command box. They appear to a Cisco device as a remark, but are filtered out before the command is send to the device. Macros start with an exclamation mark and opening curly bracket at the beginning of a line and end with the first closing bracket. The additional white space is added for better readability. A typical macro looks like the following.

```
!{T=10;W=2} sh ver
```

Macros can also be edited with the much easier “Command Line Editor”. Select a command by placing the cursor on it and press the button “Edit Command”. Only one command at a time can be edited and if more lines are selected, the first selected line will be edited. The following dialog will show.



Expect

The EXPECT macro checks the command result for a provided text. If the result contains the text an internal flag is set and any following line with SEND macro is executed. If the text cannot be found, all following lines with the SEND macro are not executed. The flag is kept, until a further EXPECT macro is used. The flag is cleared for each device and the default is to not execute SEND macro lines.

The search text can also include regular expressions. This allows matching more than one prompt or other pattern. The provided text is extended with other parameters to build a valid regular expression. This allows using the regex features without caring too much about the syntax. Typical examples for the seach text are *[pP]assword:* or *username|login*. The used regex has the following format:

(?s).(expression).**

Both EXPECT and SEND can be used on the same command, where the SEND defines, if the line is executed. If the line is executed, the EXPECT is applied to the result. If it is not executed, the flag keeps the previous state.

By default EXPECT also waits for the device to return to the command prompt. This also allows the VARIABLE macro to see all the resulting output. For interactive prompts like software updates, this would not work and here the checkbox “Interactive” should be ticked. As the expected text the column sign or the complete prompt like “filename:” can be used. If the text is found, the command will be send immediately, which in this example could contain the actual filename. The internal flag for conditionally executing the SEND macro is also set.

The text cannot contain a curly bracket or a semicolon and regular expressions are not yet supported. Text with white spaces doesn’t need to be surrounded by quote signs.

```
Syntax: X[i]=text
```

Send

If this macro is present the internal EXPECT flag will be checked. The command will only be executed, if a previous EXPECT macro could find the provided text in the output.

The “not” option inverses the behaviour and executes the command only, if the SEND macro would not send the command as a previous EXPECT macro didn’t succeed. With this option an IF-THEN-ELSE clauses can be build, where the “not” option is equivalent to an ELSE block.

Multiple SEND macros can be used for one device and they don’t need to follow directly.

```
Syntax: S[n]
```

Variable

The VARIABLE macro allows storing some information from the command output for later use. This can be used in a later command where the REPLACE macro is applied to. This macro can be used with the EXPECT macro but if that is used with the interactive option, the output might be cut off at the point, when the expected text has been found. It might be useful to filter the output on the device by using the pipe to only get one line of output.

As the parameter enter the position of a word in the command output. This also works for multiple lines, which are appended to the list of words. If the number of words is smaller, than the parameter, the variable will contain an empty text. The variable is kept for the duration of the device and set to an empty text for each new device. It is also kept, if used on a SEND macro, if the command is not executed.

The EXPECT and VARIABLE macros add an additional delay of 200 ms if one or both are used on a command. This should ensure that all output is collected first, before the two macros are applied to it.

If the checkbox “Data collection” is ticked, the content of the variable will be used for data collection and added to the device’s data set. See Data Collection for more information.

```
Syntax: V[c]=number
```

Replace

Replace placeholder in the command with dynamic values. The following are supported.

%hostname%	The hostname or IP address used in the host list
%date%	Current date in the format YYYY-MM-DD
%time%	Current time when executing this command, in the format HH-MM
%variable%	A dynamic value obtained with the VARIABLE macro
%prompt%	The detected device’s prompt without any delimiter like “#”. This should be identical to the configure hostname.

The macro itself is just enables the replacement. The placeholders are added to the actual command line. The editor allows selecting a position in the command line and pressing any of the buttons to add the placeholder to the selected position.

```
Syntax: R
```

Timeout

This can be used to extend the expected time a command can take. By default ForEachBox will wait for 30 seconds for a command to complete. If you start an image download, you want the script to extend this time to send further commands not while the device is busy.

The value is the timeout in seconds. The default timeout for every command is 30 seconds.

```
Syntax: T=value
```

Wait

This stops for the number of seconds after executing the command. This can be useful if the device needs some time to apply the change. The command can also be left empty if the timeout should be applied before the next command is executed.

The value is the delay in seconds.

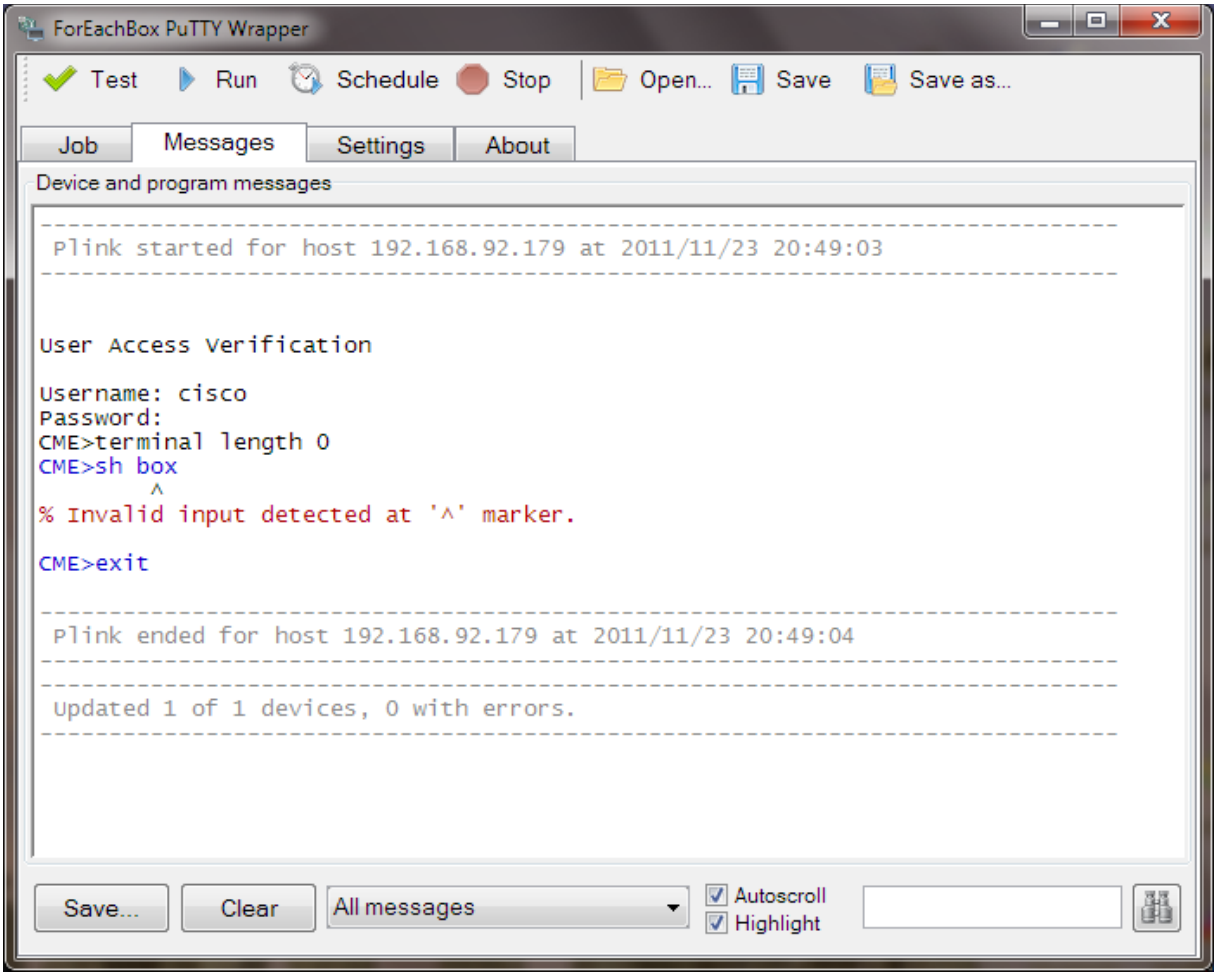
```
Syntax: W=value
```

Messages logging

The tab Messages will collect various messages from different sources apply a colour key and append them to the text area. Newest messages are added to the bottom.

Internally the messages logging uses a fast buffer. This allows it to fetch all messages from the Plink program without delay. It also avoids problems with collisions while different components send messages to this buffer. The buffer is independently (and slower) synchronized with the messages log. Applying filters and highlights will pull the information in the buffer again and apply the new rules while filling the messages box.

By default all information is sent to the messages log, which can at a certain point cause the program to run out of memory. See the Limits section for further solutions how to remedy this.



Color key

The following colour key is used. This can also be seen while hovering the mouse over the borders of the box.

Color	Usage
Black	All information sent by the device, like login and command results
Grey	Status information from ForEachBox
Blue	Output from the Plink application, like also be errors and messages during the login
Red	Error messages generated by ForEachBox, which should not happen

Button Clear

This clears all messages from the box and frees the memory. It also clears the internal buffer.

If the mouse is hovered over the button, the tool tip shows the current size of the text information in the message box. A saved log file will have the same file size. This can be used as an indicator for the memory consumption.

Button Save

This will save the content of the messages box to a file. If filters are applied, only the messages conforming to the filter will be saved to the file.

If the filename has the extention ".html" it will be saved in HTML format. This will preserve the color key of the messages.

Filter

This will apply a filter to the messages box and show only messages that match the filter. The messages are obtained from the internal buffer. Applying a filter does not remove messages, but hides them.

If the selected filter is "All messages" any result from the device is added to the log. This is the default setting. Any other filter will change the process, that normal command output is NOT added to the internal buffer. This does not apply to messages generated during the login phase. This also does not change anything for the device file log, where any messages as selected are saved.

This setting is useful if there is a lot of output generated and this causes the application to run out of memory. Keep in mind that this filters the messages while they enter the buffer, if they're already in the buffer, they will remain there.

The filter "No device output" will show all messages, except any device output. All other filters will only show one type of message.

Autoscroll

If selected, the log will always scroll to the last message that was added to the box. If unselected, the current scroll position will be kept.

Highlight

If ticked, the device output will be parsed for both syntax errors, which are lines starting with a "%" and lines containing an executed command. The following colour key will be applied.

Color	Usage
Blue	Executed commands, which includes configuration commands
Red	Syntax errors reported by the device

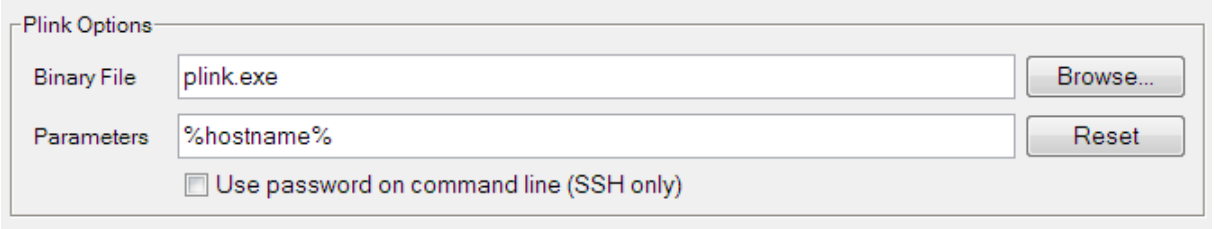
This colour coding will also be saved to a HTML file. Filters won't apply to this; the output is still considered device output.

Find

This allows searching the messages for the entered text. It will start searching from the current cursor position or selection in the messages box. If Autoscroll is active, this is always the last position and the search will start at the beginning. The starting position can be chosen by selecting a bit of text.

Plink Options

Under the settings tab the filename for the Plink application can be changed. The default does not contain any directory information. In this case ForEachBox tries to find the application in the current directory. If the file can't be found, an error message will be raised.



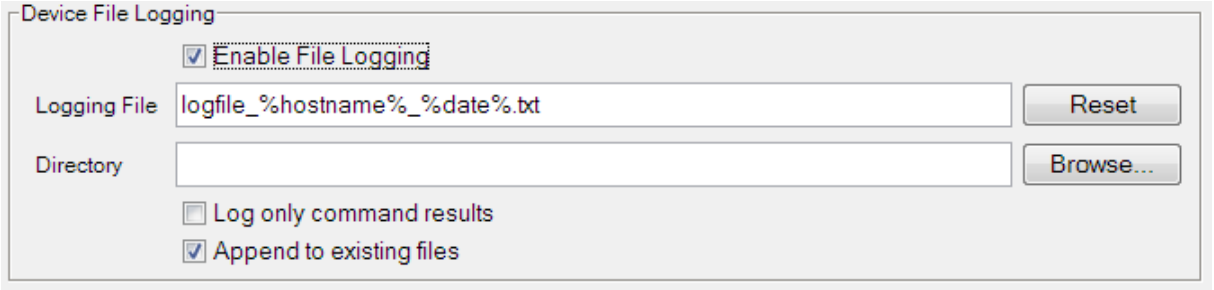
The parameter text allows adding further command line options to Plink. This can be the option to use IPv6 or the Putty Pageant. For normal operation no further options are required. If options are added, the Plink syntax must remain valid. Enter "Plink" without parameters at the command line to see all options. If the file is empty, the hostname will be appended in the same way, as shown above.

Using the password on the command line is a security thread, as the password has to be included in clear text. The password can be obtained with the Windows Taskmanager or other Tools. This option can help to login to Linux systems or

systems, that don’t use a standard login prompt. The option is disabled by default and in this case the password is send to the Plink tool via a system pipe.

Device File Logging

Under the settings tab further settings for device file logging can be found. The logging file defines the format of the file that is generated for each device.



Enable File Logging

Enable or disable Device file logging. The following settings are only effective, if File logging is enabled.

If disabled no file is written to disk, all information and messages are stored in the messages log under the tab “Messages”. The only exception is the data collection file, which is generated independently. See data collection section.

Logging file

The image above shows the default settings that are also applied, if the reset button is pressed. It supports various placeholders, which are similar to those in the REPLACE macro.

%hostname%	The hostname or IP address used in the host list
%date%	Current date in the format YYYY-MM-DD
%time%	Current time when executing this command, in the format HH-MM
%prompt%	The device prompt that has been detected. If an error occurred, the hostname will be used and prepended with a “#”.
%variable%	The value of the variable, which has been loaded with the VARIABLE macro.

Usually all messages are written to the file, at the time they’re generated. The usage of the prompt or variable placeholder causes the device output to be saved, after all commands have been sent to the device and the connection has been closed. This feature will use more memory to hold these messages.

If there is no placeholder hostname or prompt in the file name, ALL messages are saved to the same file. All messages of one job run are appended to this file. Also date and time placeholder are applied only at the start of the job.

Some characters in hostname, date or time that are not allowed in filenames are replaced. Columns in IPv6 addresses are replaced with underscores and time formats will use a hyphen as separator.

Directory

This sets the base directory for all device log files and also the data collection file. The directory name can contain white spaces, but also placeholders like in the filename.

If the directory is left empty, the current directory will be used. The device file logging in the Jobs tab shows, which directory this will be.

Data Collection

The data collection feature of ForEachBox is intended to collect information for all devices of a job in one condensed file.

It works by ticking the "Data collection" checkbox for at least one command. It works independently of the device file logging. It will also work, if device file logging is disabled.

The output file is a CSV file with no header line. First column is the hostname for the device, where the information was collected. The following columns are filled with information from the VARIABLE macro. Every use of that macro will result in an additional column.

The filename will be the same as the device file logging, where the %hostname% placeholder is replaced with the text *data_collection*. If the placeholder is missing, the text will be prepended. The file extension will be ".csv", where a trailing ".txt" will be replaced. The file will be saved in the same directory, as the device files. The setting for appending device log files is also applied to the data collection file.

Example:

The following command has been used to display version information and to pick the actual version number.

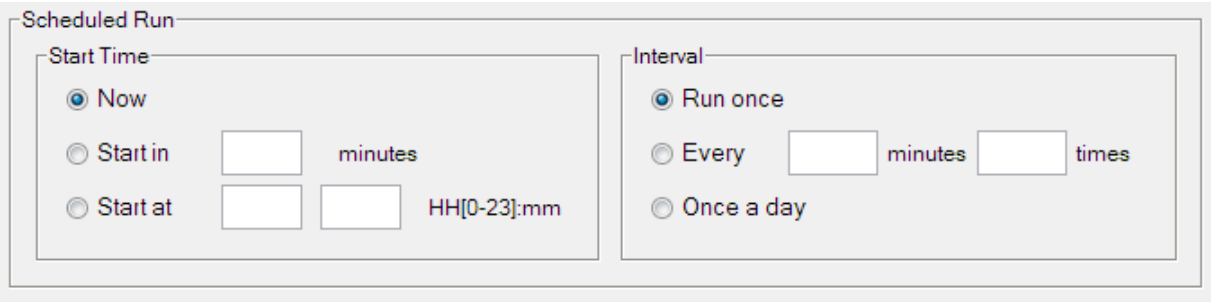
```
!(Vc=8) sh version
```

The following data has been collected.

```
192.168.92.169;12.4(15)T8,
192.168.92.170;12.4(15)T8,
```

Scheduler

Also under the Settings tab settings for a scheduler can be set. This allows to run the job at a later time and to repeat this more times.



The start time section should be self-explaining. Keep in mind, that the hour must be entered in 24 hour format.

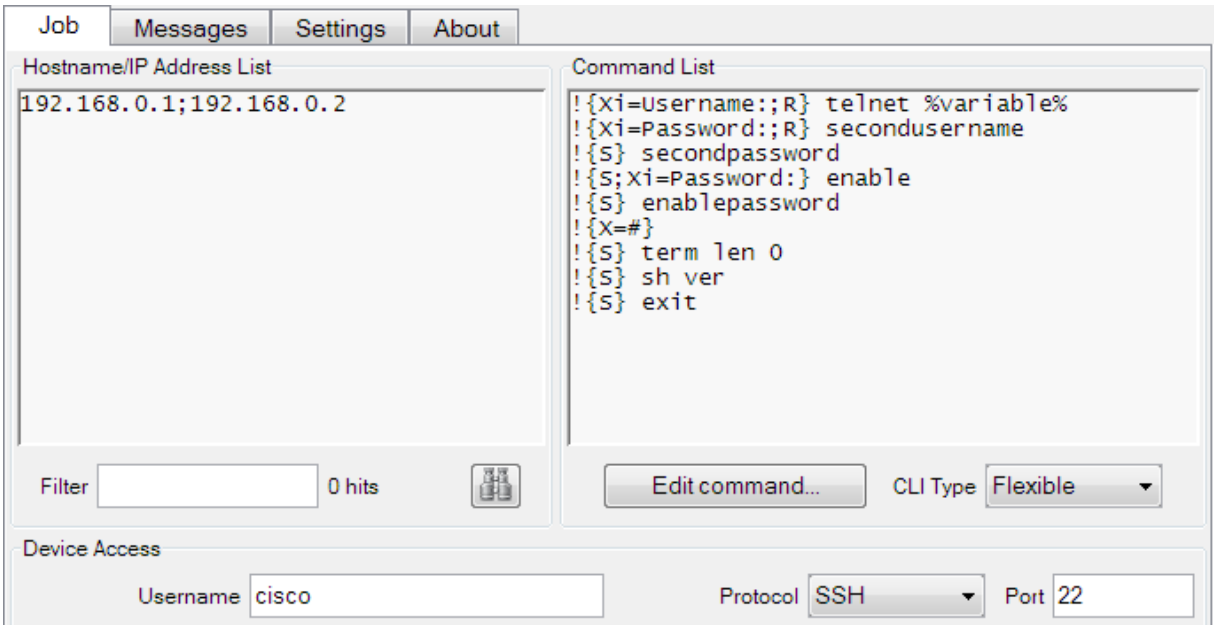
"Interval Every" allows the job to run more than once at the interval in minutes. The "times" field can be left empty, in which case the scheduler runs, until it is manually stopped.

Press the "Schedule" button in the command line to start the scheduler. Additional messages are added to the log.

Jump box support

There are scenarios where to gain access to a device, a login to an intermediate device is necessary. This can be the network management station or a network device within a remote network which has access to the other devices in that network.

ForEachBox has no direct support for this scenario, but with some specific settings this can be accomplished. Two examples in the ZIP file contain the required settings and demonstrate how a jump box can be used.



In the example above the host **192.168.0.1** is the jump box. The username and protocol settings in the section Device Access are the account information, that are used to login to this device. This host is provided with the parameter **192.168.0.2**, which is the IP address of the second device to be accessed.

The CLI type must be set to "Volatile" as the command prompt of the first device can't be seen while accessing the second device. All commands would time out and generate error messages.

After the login to the first device the command list is executed as usual. The first command initiates a telnet connection to the variable that has been loaded with the IP address 192.168.0.2. It will pause until it receives back the string "Username:" and then send back the text "secondusername". Now the password prompt is expected and the password "secondpassword" is send back upon reception.

Similar the privileged mode is entered by sending the enable command and waiting for the password prompt. The line following the enable password actually checks, if the privileged mode has been entered by checking for a hash in the prompt. The following commands are only sent, if this was successful.

Note, that the command "term len 0" has to be executed on the second device. The "show version" command is the place where the commands for that device are to be entered. All usual tools can be used here. Keep in mind, that the prompt variable still contains the value of the jump box. If the hostname or prompt are used as placeholders in the commands or in names for log files they will be the same for all devices in the list. For the logging file names use the variable placeholder to set a distinctive name.

The exit is required to terminate the connection to the second device. Even if all devices use the same jump box, ForEachBox will always terminate the session to the jump box and establish a new session for every device.

If errors occur during the connection setup to the second device, these are not reported correctly with the color key. Problems and errors are only reported in the messages logging.

Limits

ForEachBox is a lightweight application with low memory footprint. For holding all dynamic information it acquires addition memory. Especially the message logging can be quite memory intensive.

All messages are first written to a buffer, which is periodically written to the messages box. This explains why the setting "No device output" can be switched on and off – it reparses the information in the buffer again.

If the JAR file is started with default settings, it gets a small maximum memory allocation. With this setting, the messages log can hold about 6 MB of text information before the application runs out of memory. Unfortunately Java behaves a bit unpredictable if this happens and will most probably crash sooner or later.

An easy way to prevent this is, to disable the device output box in the messages tab. This will also prevent the messages to be written to the internal buffer. All messages are still written to file if selected. Clearing the messages log will also empty both log box and internal buffer.

Alternatively Java can be started with additional command line parameters that assign more memory to the application. The included Batch file does this, also the binary file as well. With the parameters

```
-Xmx512m -XX:MaxPermSize=512m
```

At least 20 MB of text information could be logged and at this time the application slowed down considerably. An out of memory error occurred and crash at about 30 MB.

Known Problems

Banners

Some login banners can interfere with ForEachBox. If you have characters in the banner, that look like a device prompt, the login will fail. Especially if a line starts with a letter and a hash symbol follows without any white space, like the example below. Only line B will cause problems.

```
banner motd ^line_A####### OK, as it is the first line
line_B###### not OK
line C##### OK as it contains a space
#####line C OK, as it doesn't contain leading characters
^
```

SSH on Linux

If you use SSH to login to a device, but forgot to enter the password, Plink will use the console to output the password prompt. You won't see any message in ForEachBox and the login will just timeout. This seems to be a bug in Plink or the way it uses the output pipe. Workaround is to enter all required authentication details.

Version of Plink

ForEachBox includes a special version of the Plink binary. This has been slightly altered to force it to send any message without delay to ForEachBox. The standard version works as well, but some output from the Plink application itself will be shown just when the process has ended. This is text usually shown in blue colour key. Normal device output is unaffected and will show in the log immediately.

The following lines of the source code for version 0.62 have been altered:

```
File WINPLINK.C Line 288
      /*
       * disable any output buffering
       * This is required to get messages from
       * Plink directly and not after exit.
       */
      setvbuf( stdout, NULL, _IONBF, 0 );
      setvbuf( stderr, NULL, _IONBF, 0 );


File WINMISC.C Lines 8 - 10:

#ifndef SECURITY_WIN32
#define SECURITY_WIN32
#endif

This was required to avoid a compiler error in MS Visual Basic 2005.
```