Fing

Fing Limited
1st Floor Minerva House
Simmonscourt Road
Dublin 4, Ireland

fing.com/business

Fing CLI - Network Tools and Device Recognition

Fing Command Line Interface - User Guide

Fing Kit for Device Recognition
Last Update: 07 March 2020
Document Version : 1.3.1

Table of Contents

1. Fing Command Line Interface

The Fing Command Line Interface (the **Fing CLI** from this point forward) is a simple but powerful tool to integrate Fing in your daily work. It's a stand-alone executable delivering a suite of network tools for troubleshooting (NIC information, scan, ping, traceroute and so on). It is available for desktop (Windows, Linux, OSx) and embedded (OpenWRT) platforms. It also can be installed on a Raspberry PI - mounting a standard Raspbian - and on any Linux system with Docker containerization system.

2. Requirements

Fing CLI runs on the following Operating Systems:

- Windows:
    - Windows Vista,
    - Windows 7,
    - Windows 8.x,
    - Windows 10
- Apple:
    - OS X Lion (10.7),
    - OS X Mountain Lion (10.8),
    - OS X Maverick (10.9),
    - OS X Yosemite (10.10),
    - OS X El Capitan (10.11),
    - macOS Sierra (10.12),
    - macOS High Sierra (10.13),
    - macOS Mojave (10.14),
    - macOS Catalina (10.15
- Debian >= 5.0,
- Raspbian,
- Ubuntu >=8.x,
- CentOS >=5.0,
- Fedora >=10,
- Slackware,
- Gentoo
- OpenWRT Chaos Calmer / LeDe on standard target/subtarget architectures
- Docker running on Linux Systems[1]

You don't need to be neither a Network Administrator nor a Domain Administrator to run Fing CLI.

An administrative user account is required to install and run the software; this is a security restriction enforced on every operating system to make sure the network is appropriately

[1] We need the docker to run with host networking. As per Docker documentation: "The host networking driver only works on Linux hosts, and is not supported on Docker Desktop for Mac, Docker Desktop for Windows, or Docker EE for Windows Server"

protected. If you are unsure about the type of user account you are using on a machine, please refer to the list below.

Windows

On Windows, you must be logged as a User having Local Administrator privileges. If you are unsure of your user type, click on:

```
Start | Settings | Control Panel | User Accounts
```

A dialog will pop-up and you shall see the account type (Standard or Administrator). You shall be listed as 'Administrator'. To run the software, a Command-line tool cmd.exe shall be executed as a User Administrator.

OS X / macOS

On OS X / macOS, you must be logged as a User having Administrator privileges.
If you are unsure of your user type, click on your User Name in the menu-bar and select "Account Preferences…".
A dialog will prompt a list of all users and types (Standard, Admin, System). You shall be listed as either 'Admin' or 'System'.

Linux - OpenWRT

You must be logged as the root user. Other equally-powerful accounts and the installation through sudo command are correct, too.

N.B. Find more about privileged access here.

Docker

You already have root permission inside the docker. You must have the permission to start docker instances on the host system.

3. Installation / Uninstallation

The installation is straightforward on all platforms: download the package and install it. Below some specification on how to install on the different operating system.

Please note that while Windows and OSx/macOS are single architecture operating system, so there is only one installer for them, Linux and OpenWRT support several architectures. You can use the following command to verify the architecture of your system:

```
uname -m
```

Alternatively, on OpenWRT, you should find all the information on files:

```
cat /etc/openwrt_version
cat /etc/openwrt_release
```

Windows

Double click on the .exe file to trigger the installer and follow the mandatory steps. If you do not have WinPcap packet capture library in your system, you will be also prompted to install it. When asked to enable the WinPcap 'NPF' Windows service, you can decide whether:

- to install and start the NPF service, thus allowing fing (and other packet capture tools) to be used by any non-admin user in the system
- not to install the NFP service, so that only admin user accounts will be allowed to use fing

To uninstall Fing on Windows platform, use the Start menu to locate the Fing folder. An Uninstall menu item is available; selecting it will run the uninstaller application that will remove the program from the system.

OSx / macOS

Download and open the .pkg file, and follow the instructions provided by the installer.
For the uninstallation, there is a script to run:

```
sudo /usr/local/bin/fing-uninstall.sh
```

Linux

The software has been packed using Debian (DPKG) and Read Hat (RPM) package manager.
It’s also available as tarball (TGZ) for Linux distros exploiting different package management systems.

Architecture supported:

- i686 and amd64 (intel 32/64 bit)
- arm and arm64 (arm soft float 32/64 bit)
- armhf (arm hard float 32 bit)
- mips and mips64 (mips big endian 32/64 bit)
- mipsel and mipsel64 (mips little endian 32/64 bit)

Download the package most suitable for your distribution and open a Terminal window.

You may refer to this list of Linux Distributions to find out the package format your platform requires:

- Debian / Ubuntu / ...
    - Installation

```
sudo dpkg -i fing-<version>-<architecture>.deb
```

- Uninstallation

```
sudo dpkg -r fing
```

- Fedora / Gentoo / ...
    - Installation

```
sudo rpm -i fing-<version>-<architecture>.rpm
```

- Uninstallation

```
sudo rpm -e fing
```

- Generic
    - Installation

```
sudo tar -zxvf fing-<version>-<architecture>.tar.gz \
         --strip-components 1 -C /
```

- Uninstallation

```
sudo /usr/local/bin/fing-uninstall.sh
```

N.B. This script is available on all Linux distribution

OpenWrt

Given the number of target and sub-targets of OpenWRT operating system.
The package can be installed using:

```
sudo opkg install fing-<target>-<subtarget>.ipk
```

On the others side, the package can be uninstalled with:

```
sudo opkg remove fing
```

Docker

Uncompress the tarball using the command

```
gunzip fing-<version>-docker_image.tar.gz
```

Then create locally the fing image using the following command:

```
docker load -i fing-<version>-docker_image.tar
```

For the removal, you have to be sure that there are not running instances of fing image:

```
docker rm -f $(docker ps -a | grep registry.fing.io/docker/kit | awk '{print
$1}')
```

then type:

```
docker rmi -f registry.fing.io/docker/kit:<version>
```

Please refer to Docker official documentation[2] for any deepening.

Available Packages

Below the full list of available packages:

Platform	Version	Package
Windows	5.5.0	https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing.exe
OSx / macOS	5.5.0	https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-osX.pkg
Linux (Debian)	5.5.0	https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-amd64.deb https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64.deb https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-i686.deb https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-armhf.deb https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64el.deb https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mipsel.deb https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-arm.deb https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips.deb
Linux (RedHat)	5.5.0	https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-amd64.rpm https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64.rpm https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-i686.rpm https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-armhf.rpm https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64el.rpm https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mipsel.rpm https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-arm.rpm https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips.rpm
Linux (Generic)	5.5.0	https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-amd64.tar.Z https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64.tar.Z https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-i686.tar.Z https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-armhf.tar.Z https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64el.tar.Z https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mipsel.tar.Z https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-arm.tar.Z

[2] https://docs.docker.com/engine/reference/commandline/docker/

		https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips.tar.Z https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-amd64.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-i686.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-armhf.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64el.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mipsel.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-arm.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips.sh https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-amd64.tar.gz https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64.tar.gz https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-i686.tar.gz https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-armhf.tar.gz https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips64el.tar.gz https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mipsel.tar.gz https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-arm.tar.gz https://s3-eu-west-1.amazonaws.com/fing-sdk/FingCLI/fing-mips.tar.gz

4. Configuration

Fing CLI comes along several configuration files that are stored in the package data directory and copied at the first run on the standard configuration folder of the operating system.

Folder

The application data folder is:

- Windows

```
%APPDATA%\Fing\conf
```

You also have a shortcut to that folder using:

```
Start | Fing | Fing Configuration
```

- OS X - Linux - OpenWRT - Docker

```
/etc/fing
```

Files

The files are all in the ".properties" format. Further information can be found here [3].

There are a lot of possible customizations that may be performed using the Fing properties file. The properties should be self explanatory, below the scope of each file:

[3] https://en.wikipedia.org/wiki/.properties

File	Description
`conf.properties`	Fing configuration management. **Please do not edit this file.**
`fing.properties`	Fing system properties. It contains: ● log configuration ● discovery configuration and defaults ● date and time format configuration (all, month, day, time) ● reverse DNS Lookup feature ● CSV separator configuration ● TCP service scan configuration: tcp syn and connect scan ● ICMP ping configuration ● HTTP proxy configuration ● HTTP Basic auth is supported, if needed by the proxy, to enable it set auth=true and fill in account
`discovery.properties`	Profile for the discovery: ● default profile: ○ native (data-link) discovery configuration for default profile ○ network discovery configuration for default profile ● slow network profile: ○ native (data-link) discovery configuration for slow network profile ○ network discovery configuration for slow network profile ● fast network profile: ○ native (data-link) discovery configuration for fast network profile ○ network discovery configuration for fast network profile
`hosts.properties`	File to declare host custom names
`ip-services.properties`	IP services for TCP and UDP protocols. **Please do not edit this file.**
`ethernet-ouis.properties`	List of MAC Vendor OUI. **Please do not edit this file.**
`kit.properties`	Fing
`timezones.csv`	Time zones configuration. **Please do not edit this file.**

5. Usage

Every tool available from the Fing CLI can be triggered using the proper command options after the command name.

The command `'fing'` without any additional parameter executes a discovery on your Local Area Network with a plain text output sent to console. This is only the simplest of reports Fing may generate.

Windows

As soon as installation is complete, you can open a command prompt and launch the software:

```
fing
```

OS X / macOS - Linux - OpenWRT

After the installation has successfully completed, open a Terminal and type:

```
sudo fing
```

Docker

You can run the container with the following command[4]:

```
docker run --tty --interactive --network host \
        registry.fing.io/docker/kit:<version> \
        /usr/bin/fing
```

6. Tools

The tools available in the Fing CLI are listed below:

Name	Description	Short Option	Long Option	Mandatory Arguments
`Info`	Show network information	-i	--info	-
`Discovery`	Run a network discovery	-n	--netdiscover	Network/Host List
`Services Scan`	Scan services on a host/network	-s	--servicescan	Network/Host List
`Ping`	Ping a list of host	-p	--ping	Hosts List
`Traceroute`	Trace the route to the target	-T	--traceroute	Hosts List
`Wake on Lan`	Wake a local device	-w	--wol	Hosts List

Since now afterwards, we refer as <cmd> to the command described in section 5.

Show Network Information

The Fing Kit can provide detailed information about the status of your network card and other useful information on your local network:

[4] Your users must to belong to docker group (i.e. it needs the rights to run docker command)

- Local Network Interfaces:
    - Type (Ethernet, WiFi, etc.)
    - Hardware Address
    - IPv4
    - IPv6
- Default gateway
- DNS Server

```
<cmd> --info
```

Scan Networks

An administrative user account is required to run the software on all platforms. The network discovery provides a complete view of any network: fing engine automatically detects the underlying network type and uses the best technique to take the picture of the target network.

The best results are achieved on Local Area Networks, both on wired Ethernet connection and wireless Wi-Fi connections, where Fing can make use of the dedicated data-link layer discovery engine which is faster one more accurate. Fing's engine can detect all the hosts present in the network, even behind a Firewall!

Discoveries performed on non-local networks (or non-ethernet networks) are handled by a network layer discovery engine, which relies on TCP/IP network layer, i.e. ICMP (ping) and TCP queries. When you start a discovery fing tells you the actual engine which is being used; in case of specific needs, it is possible to configure and tune each engine for optimal results, creating dedicated discovery profiles in the related configuration properties file:

```
discovery.properties
```

When you start Fing without arguments, it takes the nearest of your available networks and starts performing a discovery on it, reporting stuff directly on console. But you can perform discovery on any network, by providing your target network to fing in the command line:

```
<cmd> -n 192.168.1.0/24
<cmd> -n www.domotz.com/24
```

If you do not provide any output parameter, fing uses default ones as specified in its configuration file:

```
fing.properties
```

But for a running discovery you can setup as many output formats as you need, by providing a command line argument like:

```
<cmd> -o setupFormat1 setupFormat2 ... setupFormatN
```

The output format setup syntax is pretty simple: there are 2 main categories of output flows, table and log. The table flow produces a network table dump every time a discovery round is completed, while the log flow logs each network event as soon as it's detected.
If you want to change the discovery round frequency, edit the related round.interval setting in discovery.properties configuration file. Note that for each profile you declare you must provide both configurations for data-link and network layer discovery classes.
The log flow allows you to log network events in real-time, on the console itself or in a specific file. Currently there are two formats supported for log flows: text and CSV.

E.g. to start fing producing textual log in console and a CSV log in a file:

```
<cmd> -o text,console csv,fing.txt
```

The table flow instead produces a network table view refreshed each time a discovery round finishes. The most popular formats are text and HTML but here it is the complete list: stext (short text for 80-columns console), text (plain text), html, csv, and xml.

E.g. on Windows to make fing report network discovery to an HTML file on your:

```
<cmd> -o table,text table,html,"%USERPROFILE%\Desktop\network.html"
```

The network table report contains details for each host found in the network, and it's refreshed in real-time at each round: IP address, MAC address, hostname and host friendly name. The latter is a friendly name you can associate to the hosts by means of the hosts.properties configuration file, where you are able to define your custom names for hosts (by IP address or MAC address) and for networks.

By default, when you close fing the discovery session is lost, unless you want to save session data into a session file; in this case fing can be closed and restarted when you need, without losing any discovery session data. To exploit discovery session feature you have to simply provide fing the session file name to use.

E.g. on Windows to make fing generate an HTML report and save session data in a folder named report:

```
<cmd> -n 192.168.1.0/24 -o table,text,C:\report\lan.html \
            --session C:\report\lansessiondata
```

Discovering the services

The service discovery feature, also known as service scan, quickly detects active TCP services on a target host or network. Service discovery also gives its best with ethernet-based networks, where TCP SYN scan technique can be applied to audit active services on any host in a few seconds.

You can scan a local or remote host but also entire networks. You may specify a maximum number of ports for the scan to make it faster:

```
<cmd> -s 192.168.1.1               # (single host)
<cmd> -s 192.168.1.1 -m 1000       # (single host with max ports)
<cmd> -s www.fing.com              # (domain)
<cmd> -s www.google.com/24         # (entire network)
```

By default discovered services are reported on console as a plain text output, but you can choose between different output formats: like text, CSV, XML and HTML. So it is actually possible not only to use it as a command line administrative tool, but also integrated with your 3rd party applications:

```
<cmd> -s 192.168.1.1 -o html,report.html
<cmd> -s www.fing.com -o xml,scan.xml
```

Measuring the round trip time

The ping feature is capable of performing multiple quick pings to a list of target hosts. You can scan a set of local or remote hosts.

```
<cmd> -p 192.168.1.1 www.fing.com
```

By default ping summary is reported on console as a plain text output, but you can choose between different output formats: like text, CSV, XML and HTML. So it is actually possible not only to use it as a command line administrative tool, but also integrated with your 3rd party applications.

The output format setup syntax is pretty simple: there are 2 main categories of output flows, table and log. The table flow produces a network table dump every time a discovery round is completed, while the log flow logs each network event as soon as it's detected. The log flow can be removed using the option `--silent` :

```
<cmd> -p 192.168.1.1 -o html,ping.html text,console
<cmd> -p 192.168.1.1 --silent --output xml,ping.xml
```

Tracing the route of packets to a destination host

The traceroute feature is capable of for displaying the route (path) and measuring transit delays of packets sent to a target host. You can scan a set of local or remote hosts.

```
<cmd> -T 8.8.8.8
```

The summary is reported only on console as a plain text output.

```
+-----------------------------------------------------------------+
|                   === Fing 5.4.0 - www.fing.io ===              |
+-----------------------------------------------------------------+
| #hop |      host      |    avg    |   loss    |   min    |  max   |
|------+--------------+-----------+-----------+----------+--------+
|  1   | 192.168.7.1  |  1.831    |     -     |  1.371   | 2.474  |
|  2   | 10.0.1.1     |  2.223    |     -     |  2.007   | 2.514  |
                  ****************************
| 11   | 8.8.8.8      |  30.49    |     -     | 30.22    | 30.92  |
 ----------------------------------------------------------------
```

Waking device on LAN or WAN

Wake-on-LAN (WOL) is an Ethernet computer networking standard that allows a computer to be turned on - or woken up - by a network message.

WOL is implemented at hardware level by the motherboard of a computer (BIOS) and the network interface (firmware). Consequently, it is independent of the Operating System (and NIC drivers) running on the hardware, so it works on Windows, OS X and Linux machines. Some operating systems can control Wake-on-LAN behaviour via hardware drivers. The WOL message must be sent on the local network. To wake up machines from an external network (Wake-on-WAN), WOL Gateways must be configured to receive WOL packets from outside networks, and forward them to the local network.

Fing is able to send WOL magic packets to wake local or remote computers that have the WOL feature enabled. The feature works with devices that are in your connected local network, but you may also send remote WOL signals outside your local network to hosts/routers configured to be WOL gateway services by specifying the host address and port.

The syntax is very simple:

```
<cmd> --wol
```

Each target must have the following syntax:

```
MAC[@network[:port]]
```

or:

```
MAC[@host[:port]]
```

Examples

On docker, to send WOL to MAC 010203040506 in current LAN:

```
docker run --network host registry.fing.io/docker/kit --wol \
     010203040506
```

On OS X, to send WOL to MAC 010203040506 and 112233445566 in current LAN, silently:

```
sudo fing --wol 010203040506 112233445566 -silent
```

On Raspberry Pi, to send WOL to MAC 01:02:03:04:05:06 in network 192.168.0.1/24:

```
sudo fing --wol 01:02:03:04:05:06@192.168.0.1/24
```

On Linux, to send a remote WOL for MAC 01:02:03:04:05:06 to the host myremoterouter.com configured on UDP port number 9:

```
sudo fing --wol 010203040506@myremoterouter.com:9
```

7. Interactive mode

The interactive mode is will guide you through Fing tools with a step-by-step procedure. Fing is a command-line tool, and you would need to provide a (possibly long) list of arguments to specify your settings. In the interactive mode, the tool itself will guide through the available features and configurations, from basic operations to complex discoveries, in just few seconds.

The feature configures Fing and starts your discovery and additionally displays the whole command-line arguments, according to your chosen options.

Windows

You can start interactive mode from fing shortcut in:

```
Start menu | Applications
```

Alternatively, you may execute a Command-Line shell using:

```
Start menu | Execute
```

and type:

```
cmd
```

Then, a Shell window will open, allowing you to enter the command:

```
fing --interactive
```

OS X - Linux - OpenWRT

You shall open a Terminal window and execute:

```
sudo fing --interactive
```

Docker

You shall open a shell on the host system and execute:

```
docker run -it registry.fing.io/docker/kit:5.4.0 \
       /usr/bin/fing --interactive
```

Once started, the interactive mode will provide the following options:

- (D)iscover
- (S)can
- (P)ing
- (T)raceroute
- display (I)nfos

Each letter will start the corresponding Fing tool, guiding you through a series of question regarding the networks to scan, the output formats, the output destinations and such. At the end of each tool, the interactive mode will print the full set of parameters - should you need to run the same command directly in the future - and will ask you if you wish to execute it now.

```
=== You have completed the procedure ===
The equivalent command is:
fing -p localhost -o csv,console
```

8. Kit Mode for Device Recognition

Fing CLI can operates in so-called “kit” mode providing the ultimate Fing technology for network scanning and device recognition. Even if it’s an option as the other tools, we dedicate a separate section as it’s used by Fing customer to exploit the Device Recognition technology.

To enable the functionalities delivered by the kit, you must first obtain a license key and validate it. The validation requires access to the Internet. A missing or failed validation disables the features of the kit.

Configuration

Fing Kit requires a configuration file. The file is created after the first execution of fing from the template. You can find it at this location and update:

- Windows: `'%APPDATA%\Fing\conf\kit.properties'`
- Linux/Unix/OSx: `'/etc/fing/kit.properties'`

Properties

The file has to be filled with these fields at least:

```
license = 08d26f6xxxxxxxxxxe0e40
output.folder = /var/data/fing/kit
```

Below the full list:

Parameter	Description
network.interface	**Optional**. Default = "default". If available it indicate the interface name of the network to scan. By default, it runs on the default NIC.
license	**Required**. The unique license key that enable the usage of Fing Kit. The key is used to identify the Kit owner, assess the services that are enabled for a given license and to ensure the usage of the functionalities within the agreed terms. **N.B. If not set, the kit run in DEMO mode, without access to Fing technology.**
enrichment.enabled	**Optional**. **Default = true** Flag to disable the enrichment. If **licenseKey** is not set, this parameter is set to **false** automatically
refresh.interval	**Optional**. **Default = 60000** Interval between two consecutive discovery run. The value is in milliseconds. We recommend a value between half an hour (1800000) and two hours (7200000).
rounds	**Optional**. **Default = 0**. The number of discovery to perform before quitting. A 0 value means a never ending process.
output.folder	**Required**. The folder in which the kit put the output file and a copy of the configuration file. **N.B. If not set, the kit run in DEMO mode, without access to Fing technology.**

Dynamic Configuration and Usage Token

The Fing CLI in 'kit' mode requires to store some variables dynamically. These properties are stored in a file inside the output folder in the same format of the configuration file.

Below the full list:

Parameter	Description

license.usage.token	**Optional**. Default = “”. A token assigned to the running device for the present month. It is used for billing purpose to count the active installation monthly. The usage token is stored and attach to any request for enrichment to count each installation only once.

Operating Mode

Fing CLI can run in “kit” mode both in foreground or as a service, meaning that it runs as a background process, rather than being under the direct control of an interactive user.

Foreground

The command to launch the FingCLI in kit mode is (as administrative user!):

```
<cmd> --kit <kit-configuration-file-path>
```

The workflow can be divided in three phases, as shown in the stdout:

- License Parse

The license key is parsed to collect and show basic information on the license (owner, id, activation date, ...)

```
Fing Kit License:
      Company:     Fing
      Customer-ID: fing
      License-ID:  test-kit-embedded
      Act. date:   2018/07/02 10:01:26
```

- Configuration Load

The properties listed in the configuration above are printed. The refresh interval is printed in milliseconds. The location of the configuration file is reported twice as explained above.

```
Fing Kit starting on:
      Network:             en0 (Ethernet)
      Refresh:             3600
      Rounds:              -
      Output Folder:       /var/data/fing/kit/
      Kit Properties File (Read-Only): /etc/fing/kit.properties
      Kit Properties File (Operating): /var/data/fing/kit//kit.properties
```

- Discovery Process

A discovery run started on the specified interface, with a determined network and the corresponding layer. The output reports periodic updates on the progress.
When the discovery terminates, the following information are available:

- License:
  - Verification elapsed
  - Expiration time
  - Usage Token, which is A token assigned to the running device for the present month
  - Customer ID
  - Usage Counted (i.e. whether the enrichment has trigger the usage count for billing)
  - Kit Grants
- Devices:
  - IP
  - MAC Address
  - Device Name
  - Type (see Appendix 1 for further detail)
  - Make
  - Model
  - OS Name, Version and Build Number
  - Rank (see Appendix 2 for further detail)
- Network
  - Gateway
  - DNS Server
  - Internet Service Provider

```
- Network discovery started on en0: 192.168.1.0/24 (data-link layer)
 - Discovery progress: 25%
 - Discovery progress: 50%
 - Discovery progress: 75%
 - Discovery progress: 100%
 - Internet info refreshed: CONNECTED - Telecom Italia Mobile
 - Kit License verified in 00:00.476
 - Kit Session expiration: 2020/03/10 15:55:23.454
 - Kit Usage Token: 'Bc1Ey2vOhItI/5XZ383eQA=='
 - Kit Customer ID: fing
 - Kit Usage Counted: NO
 - Kit Grants: [DISCOVERY] [ENRICHMENT] [ACCOUNT]
 - Enrichment completed in 00:00.365
 > Device - ...
 > Device - ...
 > Device - ...
```

```
  > Gateway: 192.168.43.1 (0C:2C:54:0A:CB:75)
  > DNS Server: 192.168.43.1
  > Internet: YES via [Telecom Italia Mobile] Location [Italy/Treviso]
- Network discovery completed on 192.168.43.0/24: 2 devices (in 00:17.914)
```

Background

With few steps, it's easy to perform network discoveries in the background and generate reports at regular intervals.

- Windows

If executed as a Windows Service, Fing automatically starts when you power on your computer, and automatically stops when you shut it down. As all Windows services, Fing Services can be managed from the Windows Service Control Panel.
You must use absolute paths for every file you provide in the command line (both output files and session file). Fing Service will run as LocalSystem user and will use the Home Folders of the user that registers the service to read and store configuration files, log files and any other accessory file. Fing CLI provides an utility to manually install it:

```
fing --installservice FingKit --kit "%APPDATA%\Fing\conf\kit.properties"
```

The corresponding Fing network discovery with device recognition is installed as service and immediately started. Later, if you want to later uninstall the service, just type:

```
fing --uninstallservice FingKit
```

Note that, at the moment, Fing service removal is not supported when completely uninstalling Fing. Hence, if you are in the process of upgrading or removing Fing software from your computer, you should stop and uninstall running Fing services as an initial step.

- OS X / macOS

Copy or link the launched script from the installation folder to OSx users' dameon folder:

```
sudo cp /usr/local/lib/fing/launchd/com.fing.fingkit.FingKit.plist \
              /Library/LaunchDaemons
```

To manually load and start the service in background, type:

```
sudo launchctl load /Library/LaunchDaemons/com.fing.fingkit.FingKit.plist
```

To unload:

```
sudo launchctl unload /Library/LaunchDaemons/com.fing.fingkit.FingKit.plist
```

Further documentation about Daemon and Services on Apple operating systems are available here.

- Linux - OpenWRT

The widest, simplest and most stable daemon manager for GNU/Linux operating systems is System V. The new *systemd* of Red-Hat family and *systemctl* of Ubuntu/Debian family are the replacement for SysVinit and Upstart.

Because of they are compatible with SysVinit, and lots of embedded device runs Operating System (like OpenWRT) that still work with SysVinit, we do provide the instruction for that daemon manager.

Copy or link the init.d script from to System V folder:

```
sudo ln -s /usr/local/lib/fing/init.d/fing-kit /etc/init.d/
```

Then add to your services with:

```
sudo chkconfig --add fing-kit
```

or:

```
sudo update-rc.d fing-kit defaults
```

To manually control the service:

```
sudo service fing-kit start
sudo service fing-kit stop
```

- Docker

Before starting the docker, you have to prepare the following directory tree:

```
.
├──── data/
├──── kit.cfg
└──── log/
```

where data/ and log/ are empty folder and kit.cfg is the configuration file of kit tool as described here.

From the root directory of the tree, tap:

```
docker run --network host --detach \
       --mount type=bind,source=$(pwd)/kit.cfg,target=/var/fing/kit.cfg \
       --mount type=bind,source=$(pwd)/log,target=/var/log \
       --mount type=bind,source=$(pwd)/data,target=/var/data \
       registry.fing.io/docker/kit:<version>
```

N.B. The mount options is required to have the log and the output folder on your system.

Output

The Fing CLI returns the set of results format in according to configuration.

At the moment, JSON format is supported, which allow an easy integration with any kind of hosting app or process. The output is available into the output folder specified in the `'kit.properties'` file

Discovery dataset of the network

For the current network, FingKit will provide a JSON data structure describing the network details and analysed properties. This is the set of details returned.

Key	Value	Example
result_state	Flag discriminating if this scan has been enriched by Fing Device Recognition service	"enriched"
last_scan_timestamp	The time of the last scan	"2016/11/23 02:00:07"
time_zone	The time zone of the scanning device	"CEST"
nodes_count	The amount of nodes found in the network	"12"
nodes_up_count	The amount of nodes found online in the network.	"10"
network	Network dataset	
isp	Service Provider dataset	
nodes	List of Network node base dataset	

Network dataset

This is the set of details returned to network interface monitored.

Key	Value	Example
address_type	IPv4 or IPv6	IPv4
name	The network name from the interface	eth0
address	The network address	192.16.0.0
mask_prefix_length	The netmask length applied by the scan engine, in bits	24
gateway_ip_address	The IP address of the gateway	192.168.0.1
dns_address	The IP address of the DNS	192.168.0.1

Internet Service Provider dataset

If internet connection is available, the scan reports also additional details on the ISP connection and location. Some of these details may not be available, depending on the user's connection.

Key	Value	Example
country_city	The city name	Rome
address	The public IP address	62.23.136.134
host_name	The public host name	acces.134.136.23.62.rev.coltfrance.com
latitude	The latitude of the ISP point in decimal degrees	12.4833
longitude	The longitude of the ISP point in decimal degrees	41.8999
timezone	The time zone of the ISP	Europe
organization	The name of the organization providing Internet Access	COLT Technology Services Group Limited
country_code	The 2-letters country code	UK
country_region_code	The region code	LAZ
continent_code	The 2-letters country code	EU
country_postal_code	The postal code of the address	W10 5BN

Network Node base dataset

For each identified device, Fing provides a data structure describing the network details and recognition result and also analysed network protocols properties.

Key	Value	Description
mac_address	The MAC Address of the device that is currently using to connect to the network	"06:5c:89:c9:e7:d1"
addresses_list	The list of IP address assigned to the device in the current network. It may be multiple if the element is a network bridge or if it's temporarily being assigned multiple addresses	"172.28.0.14"
state	Discriminates if the device is connected to the network or not. Can be "UP" or "DOWN"	"UP"
best_name	The best name of the device, evaluated from the names returned from the various protocols it replies to	"HP 2832", "Marco's iPhone"
best_type	A single type identifying its major role. It's intended to be as brandless as possible. **See Appendix 1 for further details.**	"Laptop", "Mobile", "Photo Camera", "Desktop".
best_make	The name of the makers/vendor of the device. It may overlap with the manufacturer, but it may be also different in case the network interface (ETH, WIFI) is different.	"Apple", "Huawei" (but not "Foxconn")
best_model	The human-readable name of the model	"iPhone 5S", "P9"
is_family	Flag advicing if the model is a generic family and not a specific model.	true
best_os	The name of the Operating system, when applicable	"iOS", "Android", "Windows", "macOS".

best_osver	The version of the Operating system, when applicable	"7 Ultimate", "10 Pro", "Mojave"
best_osbuild	The build number of the Operating system, when applicable	"19D88", "30.3454"
recog_rank	Rank value of the device recognition	95
host_name	The DNS name of the device	"mydevice.thissite.com"
mac_vendor	The name of the company that is officially manufacturing the network interface (ETH or WIFI). Names are reviewed and optimized to be consistent	"Samsung", "Apple", "Lenovo" for major brands, but also "Foxconn" for manufacturers that registered their components directly
netbios	Network node detail dataset for NetBIOS	
bonjour	Network node detail dataset for Bonjour	
upnp	Network node detail dataset for UPnP	
dhcp	Network node detail dataset for Dhcp	
dhcp6	Network node detail dataset for Dhcpv6	
http	Network node detail dataset for Http	
snmp	Network node detail dataset for Snmp	

Network node detail dataset for NetBIOS

FingKit exports for NetBIOS the following JSON structure, contained in the "netbios" JSON key, if the full protocol detail is configured.

Property	**Description**	**Example**
name	The NetBIOS name is used to uniquely identify the NetBIOS services listening on the first IP address that is bound to an adapter. The NetBIOS name is also known as a NetBIOS computer name.	"MACBOOKPRO"
domain	A type of Fully-qualified Domain Name.	"mypc.locallan"
user	An optional user name. Due to security concerns, this is rarely available in the standard implementation	"MARCO"
is_file_server	An optional flag to detect if available file server is running.	"1" or "0"
is_domain_controller	An optional flag to detect if available domain controller is enabled.	"1" or "0"

Network node detail dataset for Bonjour

FingKit exports for Bonjour the following JSON structure, contained in the "bonjour" JSON key.
If the full protocol detail is configured.

Property	**Description**	**Example**
name	The Bonjour name the device publishes	"name": "Giuseppes-MacBook-Pro"

model	The Bonjour model the device publishes	"model": "MacBookPro11,4"
os	The Bonjour Operating System name the device publishes	"os": "OSX:17"
serviceinfo_list	A list of bonjour additional services published by the device	{"name": "Giuseppe\u0019s MacBook Pro._device-info._tcp.local." , "addinfos": { "model":"MacBookPro11,4", "osxvers": "17" } }

Network node detail dataset for UPnP

FingKit exports for UPnP the following JSON structure, contained in the "upnp" JSON key. If the full protocol detail is configured.

Property	Description	Example
name	The UPnP name the device publishes	"My Macbook"
make	The UPnP Make name the device publishes	"Samsung"
model	The UPnP Model the device publishes	"SCD8291221"
type_list	A list of UPnP device types published by the device	"urn:Belkin:device:controllee:1"
service_list	A list of UPnP services published by the device	"urn:Belkin:service:manufacture:1" "urn:Belkin:service:smartsetup:1"

Network node detail dataset for Http User Agent

FingKit exports for Http User Agent the following JSON structure, contained in the "http" JSON key. If the full protocol detail is configured and http user agent if available and it is enabled as option. Please note HTTP user agent can be got only if FingKit is running on a gateway device, like e.g. the network router.

Property	Description	Example
useragent	The Http user agent list	"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET4.0C; .NET4.0E; Media Center PC 6.0; InfoPath.3; BRI/2)"

Network node detail dataset for SNMP

FingKit exports for SNMP the following JSON structure, contained in the "snmp" JSON key. If the full protocol detail is configured.

Property	Description	Example
sysoid	The unique identifier of the device type	"1.3.6.1.4.1.9.1.516"

name	The SNMP name the device publishes	“HP
services	The SNMP list services the device publishes	
description	The SNMP description of the device	“Cisco IOS Software, C3750 Software (C3750-IPSERVICESK9-M), Version 12.2(46)SE”
contact	The SNMP contact point	“admin@cisco.com”
location	The SNMP location of device	“North Corridor”

Network node detail dataset for DHCP

FingKit exports for DHCP the following JSON structure, contained in the “dhcp” JSON key. If the full protocol detail is configured.

Property	Description	Example
name	The DHCP name the device publishes	“My Macbook”
vendor	The DHCP vendor	“Samsung”
params	The DHCP params	“1,33,3,6,15,28,51,58,59”

Network node detail dataset for DHCP6

FingKit exports for DHCPv6 the following JSON structure, contained in the “dhcp6” JSON key. If the full protocol detail is configured.

Property	Description	Example
name	The DHCPv6 name the device publishes	“DESKTOP-TR18HAM”
vendor	The DHCPv6 vendor	“Samsung”
options	The DHCPv6 options	“1:8,1,3,39,16,6”
params	The DHCPv6 option params	“17,23,24,39”
enterpriseid	The DHCPv6 enterprise id	311

Full Sample

An example of the JSON file is reported below:

```
{
    "result_state": "enriched",
    "last_scan_timestamp": "2020-03-10 09:55:05",
    "time_zone": "CET",
    "nodes_count": "2",
    "nodes_up_count": "2",
    "network": {
        "address_type": "IPv4",
        "name": "en0",
        "address": "192.168.43.0\/24",
        "mask_prefix_length": "24",
        "gateway_ip_address": "192.168.43.1",
        "dns_address": "192.168.43.1"
    },
    "isp": {
        "country_city": "Treviso",
        "address": "5.170.193.79",
        "host_name": "5.170.193.79",
        "longitude": "12.242800000000001",
        "latitude": "45.661799999999999",
        "timezone": "Europe\/Rome",
        "organization": "Telecom Italia Mobile",
        "country_name": "Italy",
        "country_code": "IT",
        "continent_code": "EU"
    },
    "nodes": [
        {
            "mac_addresses": "0C:2C:54:0A:CB:75",
            "address_list": [
                "192.168.43.1"
            ],
            "state": "up",
            "best_type": "MOBILE",
            "best_make": "Huawei",
            "best_model": "P20 lite",
            "best_os": "Android",
            "recog_rank": "95",
            "mac_vendor": "Huawei"
        },
        {
            "mac_addresses": "8C:85:90:5E:32:F2",
            "address_list": [
                "192.168.43.68"
            ],
            "state": "up",
            "best_name": "Gargantua",
            "best_type": "LAPTOP",
            "best_make": "Apple",
            "best_model": "MacBook Pro",
            "best_os": "macOS",
            "best_osver": "19.0.0",
            "best_osbuild": "19B88",
            "is_family": "true",
            "recog_rank": "99",
            "host_name": "Gargantua",
            "mac_vendor": "Apple",
            "bonjour": {
                "name": "Gargantua",
                "model": "MacBookPro14,1",
                "os": "OSX:19",
                "serviceinfo_list": [
                    {
                        "name": "Gargantua._device-info._tcp.local.",
                        "addinfos": {
                            "ecolor": "157,157,160",
                            "model": "MacBookPro14,1",
                            "osxvers": "19"
                        }
                    },
                    {
                        "name": "Gargantua._eppc._tcp.local.",
                        "addinfos": ""
                    },
                    {
                        "name": "Gargantua._sftp-ssh._tcp.local.",
                        "addinfos": ""
                    },
                    {
                        "name": "Gargantua._smb._tcp.local.",
                        "addinfos": ""
                    },
                    {
                        "name": "Gargantua._ssh._tcp.local.",
                        "addinfos": ""
                    }
                ]
            }
        }
    ]
}
```

Appendix 1 - Fing Categorization - Groups and Types

For each device, Fing will analyze all the details and provide the best match among its supported types and categories. The list is reviewed and grows constantly as our Machine Learning system evolves.

Group	Device types
Mobile	Generic, Mobile, Tablet, MP3 Player, eBook Reader, Smart Watch, Wearable, Car
Audio & Video	Media Player, Television, Game Console, Streaming Dongle, Speaker/Amp, AV Receiver, Cable Box, Disc Player, Satellite, Audio Player, Remote Control, Radio, Photo Camera, Photo Display, Mic, Projector
Home & Office	Computer, Laptop, Desktop, Printer, Fax, IP Phone, Scanner, Point of Sale, Clock, Barcode Scanner
Home Automation	IP Camera, Smart Device, Smart Plug, Light, Voice Control, Thermostat, Power System, Solar Panel, Smart Meter, HVAC, Smart Appliance, Smart Washer, Smart Fridge, Smart Cleaner, Sleep Tech, Garage Door, Sprinkler, Electric, Doorbell, Smart Lock, Touch Panel, Controller, Scale, Toy, Robot, Weather Station, Health Monitor, Baby Monitor, Pet Monitor, Alarm, Motion Detector, Smoke Detector, Water Sensor, Sensor, Fingbox, Domotz Box
Network	Router, Wi-Fi, Wi-Fi Extender, NAS, Modem, Switch, Gateway, Firewall, VPN, PoE Switch, USB, Small Cell, Cloud, UPS, Network Appliance
Server	Virtual Machine, Server, Terminal, Mail Server, File Server, Proxy Server, Web Server, Domain Server, Communication, Database
Engineering	Raspberry, Arduino, Processing, Circuit Board, RFID Tag

Appendix 2 - Fing Recognition Rank - Computation

Recognition rank is a measure from 0 to 100 of the quality of the recognition.

Its scale is not a standard one like you could expect (e.g. where above 60 is sufficient and below 60 is not good) The higher and nearer to 100 is the better, but even low values are good. In case no decent recog is available, no result is provided. It can always be used in comparisons, meaning that if r1 > r2, r1 is always better

Some references:

- 90+
  Sure information, gathered directly from the devices or by processing sure info with ML
- 40+
  Very stable protocol info (e.g. from SNMP, UPnP, Bonjour, HTTP User Agent fingerprints)
- 20+
  DHCPv4/v6 info, which in most cases is best effort but in some others is as strong as protocols above
- 1-20
  NetBIOS, empiric rules (e.g. rules based on hostnames and mac vendor)

Our machine learning algorithms, when processing the fingerprints and generating prediction models for recognition, interpolate the above reference scores by weighting confidence levels.
It's recommended to always keep Fing results, never discard even low values, because the drop is performed by our engine itself, avoiding the most possible to send misrecognitions.
The more protocols are provided the better recognition works.