Fing

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

phone : (+44) 203 598 4003
email : sales@fing.io

www.fing.io

FingKit

FIND AND IDENTIFY DEVICES, ON ANY NETWORK

FingKit API Specification
Last Update : January 2018
Document Version : 3.0

1. Introduction

Fing is a network analysis solution that provides insight on the user's network. The solution relies on a Software Development Kit (FingKit) for mobile apps and embedded devices that discovers the devices connected to the network, and a cloud service that identifies their models and properties through advanced Machine Learning.
The kit analyzes the network the device is connected to, emitting a number of details about the whole network and about each single device such as device name, make, model and type.

This system is at the heart of **Fing**^{TM} mobile apps, already downloaded by 15 million users and performing over 700,000 networks scans every day. FingKit uses a state-of-the-art discovery engine, running a collection of algorithms developed in years of experience on mobile and embedded devices, aimed at teams to save crucial development time and get unparalleled device recognition.

Fing discovery engine is blazing fast: in a few seconds FingKit provides a detailed snapshot of the devices currently connected and devices that joined the network in the past. The recognition of devices relies on Fing's distinctive algorithms to probe devices, Bonjour (Multicast-DNS) queries, UPnP queries, SNMP requests, DHCP monitoring and NetBIOS queries.

How does it work?

For each device, the FingKit Device Recognition System analyzes the data to provide the best match among its supported types – currently more than 100 types ranging from tablets to surveillance cameras, grouped in 8 categories (Mobile, Entertainment, Home & Office, Network, Server, Home Automation, Surveillance, Engineering).

In addition to the best-matching device type, FingKit provides in JSON text format the full set of network details and the analysis for each network protocol the devices comply with.

The table below reports the scan and protocols supported on each platform.

Feature	**Mobile Android**	**Mobile iOS**	**Computer Windows, Linux, macOS, OpenWRT**	**Notes**
Network discovery	✓	✓	✓✓	PC discovery 100% reliable on Ethernet and Wi-Fi networks performed at low level
Hostname	✓	✓	✓	
NetBIOS	✓	✓	✓	
UPnP	✓	✓	✓	
Bonjour	✓	✓	✓	
SNMP	✓	✓	✓	
DHCP		✓	✓	Best effort DHCP discovery on iOS

2. Integrate FingKit into a native app

Integrating the FingKit is easy and straightforward for iOS and Android developers. Once you create an app and get your Fing License Key, you just need to add the required frameworks for Fing integration, initialize Fing and start a netwrk scan.

Integration within an iOS app

The Kit is available as an Objective-C Framework library, suitable to be used with the standard development tools (Xcode) and to be published on the official Apple Store. As a framework, it may also be used by application written in Swift language.

FingKit for iOS is compatible with Apple iOS 9.x and greater. FingKit requires the following items to be added in the "**Linked Frameworks and Libraries**" in your Xcode project.

- `libresolv.9.tdb`
- `libsqllite3.tbd`
- `SystemConfiguration.framework`
- `Security.framework`
- `Foundation.framework`
- `CFNetwork.framework`
- `CoreTelephony.framework`

The FingKit framework itself shall be added as "**Embedded Binaries**" as well; Xcode automatically includes the framework in the final package. To import and use the functionalities of the FingKit modules, you shall simply import the module main header.

```
#import <FingKit/FingKit.h>
```

The FingKit functionalities are accessed via the main singleton class `FingScanner`.

Integration within an Android app

The Kit is available as an AAR (Android Archive) library, suitable to be used with the standard development tools (Android Studio) and to be published on the official Play Store. As a framework, it may also be used by applications written in Kotlin language.

FingKit for Android is compatible with Android 4.4 and greater. FingKit requires the following dependencies to be added in your Gradle-based or Maven-based project.

Group	Name	Version
com.android.support	appcompat-v7	25.4.0
com.google.android.gms	play-services-analytics	11.4.2

org.snmp4j	snmp4j	2.5.0

The FingKit archive `fing-kit.aar` shall be placed locally in a folder placed at the same level of the android app source code, (e.g. if your source code is in `<root/app/src>`, you shall place the lib in `<root/app/libs>`) and it shall be added as transitive compilation item in your build system. Android Studio automatically includes the framework in the final package. Below is reported an excerpt of a Gradle build module that includes the library in the build system.

Android (Gradle)

```
allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
        google()
    }
}

dependencies {
    compile(name:'fing-kit', ext:'aar') {
        transitive=true
    }

    compile 'com.android.support:appcompat-v7:25.4.0'
    compile 'com.google.android.gms:play-services-analytics:11.8.0'
    compile 'com.google.protobuf:protobuf-java:2.6.1'
    compile 'org.snmp4j:snmp4j:2.5.0'
}
```

The FingKit functionalities are accessed via the main singleton class `FingScanner`.

3.API Specification

Asynchronous design

FingKit operates asynchronously, to ensure your App is never blocked during each operation. A callback block is used to deliver the result of an operation, or the error object in case the operation could not be completed.
All callback methods are invoked in the main thread on iOS and Android.

iOS (Objective-C)

```
typedef void (^FingResultCallback)
             (NSString *_Nullable result, NSError *_Nullable error);
```

Android (Java)

```
public interface FingResultCallback {
    void handle(String result, Exception error);
}
```

The callback block accepts the following list of parameters:

Parameter	iOS/Android Type	Description
result	NSString * / String	The result coming from the FingKit. It may be nil/null if there is no result or an error occurred. The result is usually in JSON format, but in general it depends on the type of operation
Error	NSError * / Exception	An error descriptor, in case the operation may not be completed.

If successful, the completion callback result string contains a JSON-formatted result and a `nil/null` error.

Error handling

On Android, errors are represented as Exception objects passed as paramters. On iOS, the completion callback may return one of the following error codes in the `NSError` object if the attempt to validate the key failed.

Error Code	Description
-100	The provided key is not valid
-101	The service replied, but could not validate the key
-102	The operation timed out
-200	Account operation failed
-300	Scan operation failed

License Key validation

To enable the functionalities delivered by the FingKit, you must first obtain an API key and validate it. The validation requires access to the Internet, and it shall be executed at every application session in order to activate the features; a missing or failed validation disables the features of the FingKit.

iOS (Objective-C)

```
-(void) validateLicenseKey:(NSString *) key
                 withToken:(NSString *) token
                completion:(nullable FingResultCallback) completion;
```

Android (Java)

```
public void validateLicenseKey(String key,
                               String token,
                               FingResultCallback completion);
```

The method accepts the following list of parameters:

Parameter	iOS/Android Type	Description
Key	NSString * / String (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

token	`NSString * / String`
	`(Optional, max 512 characters)`
		An token generated by your App or by your backend services, that will be sent back to your remote services through a webhook.
		The purpose of the optional token is to allow you to recognize, if needed, the activation of a session using your license key.

Completion	`FingResultCallback`
	`(Optional)`
		A callback block that is invoked when the validation terminates.

If successful, the callback contains a JSON-formatted result as described in the following table, and a `nil/null` error.

Key	Value	Example
kitLicenseId	Your license key	Will be the same value passed as parameter
kitCustomerId	Your unique customer identifier, assigned on sign up. Usually, it's your company or App name	ACME
expiryDate	The time at which the provided key expires and a new key or new validation shall be performed	2016/11/23 02:00:07
state	The state of the license. It may be one of: • Ok • Suspended • Revoked	Ok
grantDiscovery	A Boolean value indicating if the network discovery feature is granted by your license	true
grantEnrichment	A Boolean value indicating if a Fing Service enrichment is enabled. Enrichment provides additional results on top the local scan, such as device type recognition.	true
grantAccount	A Boolean value indicating if the ability to attach the App to an account is granted by your license.	True
usageToken	A token assigned to the running device for the present month	ABC123
usageCounted	A Boolean value indicating if this validation was the first validation of the licensing period	true

If the validation could not be performed or fails, a description of the error is reported in the `NSError` object.
An example of the JSON result is reported below.

iOS and Android (JSON)

```
{
  "kitLicenseId":”ABC123”,
  "kitCustomerId":”ACME”,
  “expiryDate”:”2016/12/30 00:00:00”,
  “state”:”Ok”,
  “grantDiscovery”: “true”,
  “grantEnrichment”: “true”,
  “grantAccount”: “false”,
  “usageToken”: “ABC123”,
  “usageCounted”: “false”
}
```

A failure to validate the key is reported via an NSError. Every error in the validation process disables all functionalities.

Fing Account management

Hosting applications manage the Fing account to persist backup, synchronize, persist the scanned networks. The functionalities allow users to receive at every scan not only the current snapshot, but the full history of the device states and times.
An account is also required to perform our cloud enrichment to improve the identification of devices.

FingKit ensures a secure and encrypted communication between the App and Fing Servers. Each account is identified by an App-provided unique identifier. FingKit do not impose any authentication mechanism, relying on the hosting app to implement its own authentication system.

iOS (Objective-C)

```
-(void) accountAttach:(FingAccountProfile *) profile
            withToken:(nullable NSString *) token
           completion:(nullable FingResultCallback) completion;

-(void) accountInfo:(nullable FingResultCallback) completion;

-(void) accountDetach:(nullable FingResultCallback) completion;
```

Android (Java)

```
public void accountAttach(FingAccountProfile profile,
                          String token,
                          FingResultCallback completion);

public void accountInfo(FingResultCallback completion);

public void accountDetach(FingResultCallback completion);
```

The three methods (attach, verify, detach) manage the lifecycle of a Fing Account:

- **ATTACH**: attaches the current hosting app to an account. The operation automatically configures a new account if missing, or update it if already existing, with the given `FingAccountProfile` object

- **DETACH**: detaches disconnects the kit from the given account ID. The operation detaches also any account-related functionalities such as device enrichment
- **INFO**: returns the state of the currently connected account, if any.

The methods accept the following list of parameters:

Parameter	iOS / Android Type	Description

profile	`FingAccountProfile`
	`(Required)`
		The set of profile data to sign in or sign up an account. This parameter is mandatory, and must contain a valid user id as a string in the [azAZ_09] charset and max length of 256 characters.
		See Fing Account Profile for details

completion	`FingResultCallback`
	`(Optional)`
		A callback block that is invoked when the method completes.

If successful, the completion callback result string contains a JSON-formatted result as described in the following table.

Key	Value	Example
attached	A Boolean value indicating if the account is attached or detached at the end of the operation.	“false”
name	The user name	“John Appleseed”
user_id	The unique user id. It could be prefixed by the customer identifier	“domotz,abc”
network_count	The amount of networks in the account. An optional value provided only for network info operations	3

An example JSON is reported below.

iOS and Android (JSON)

```
{
  "attached":”true”,
  ”name”:”John Appleseed”,
  “user_id”:”domotz,abc”,
  “network_count”:3
}
```

Fing Account Profile

The following scan options may be specified through the appropriate `FingAccountProfile` object:

Option	iOS / Android Type	Description
`accountId`	`NSString * / String`	The unique account identifier that the hosting application has authenticated.

	(Required)	The identifier may be a user's e-mail address or any other internal code.
accountFullName	NSString * / String (Required)	The full name of the account's holder
accountEmail	NSString * / String (Required)	The email address of the account's holder

Network info

The FingKit allows to conveniently retrieve network details from the Wi-Fi the device is connected to. The network details may be retrieve through the following method.

iOS (Objective-C)

```
-(void) networkInfo:(nullable FingResultCallback) completion;
```

Android (Java)

```
public void networkInfo(FingResultCallback completion);
```

If successful, the callback contains a JSON-formatted result as described in the following table, and a `nil/null` error.

Key	Value	Example
address	The base IP address of the network	192.168.0.0
netmask	The netmask expressed as CIDR notation. It represents the number of bits that make up the subnet part, and consequently the remaining bits identify the host part	24
bssid	The BSSID, that is the MAC Address of the Access Point the device is connected to at the moment	AA:BB:CC:00:01:02
ssid	The name of the network, as assigned by the network administrator	My Network
gatewayAddress	The IP Address of the network gateway, if available	192.168.0.1
dnsAddress	The IP Address of the network DNS, if available	192.168.0.1
hasConnectivity	Discriminates if the current connection with the server has network connectivity	true

Network scan

The main functionality is accessed via a single method that performs the scan and enrichment of data, if enabled. The scan is integrated with the Fing Account and Fing Device Recognition Service, based on the features and services enabled on your API key.

iOS (Objective-C)

```
-(void) networkScan:(nullable FingScanOptions *) options
          completion:(nullable FingResultCallback) completion;
```

Android (Java)

```
public void networkScan(FingScanOptions options,
                        FingResultCallback completion);
```

If a Fing Account is enabled on your API key, the scan will report also the state of devices from previous executions, and will perform automatic merge of multiple BSSIDs for the same network. As last step of the scan process, the network will be added or updated in the account, enriched and send to the hosting App.

The scan progress is delivered asynchronously to a completion handler, so that hosting Apps can be informed and display the progress of the execution.

The method "scan" accepts the following list of parameters:

Parameter	Type	Description

options	`FingScanOptions *`
	`(Optional)`
		The set of options to tune the network scan procedure.
		See Scan Options for details

completion	`FingResultCallback`
	`(Optional)`
		A callback block that is invoked when the validation terminates.
		The validation may check both locally and remotely the given key, and report the result or an empty result with an error.
		See section 4 for details.

Scan Options

You may enable and tune the scan process through a set of Options. The following scan options may be specified through the appropriate `FingScanOptions` object:

Option	iOS / Android Type	Description
`reverseDnsEnabled`	`Bool / boolean`	Enables Reverse DNS
`upnpEnabled`	`Bool / boolean`	Enables UPnP scan
`bonjourEnabled`	`Bool / boolean`	Enables Bonjour scan
`netbiosEnabled`	`Bool / boolean`	Enables NetBIOS scan

`snmpEnabled`	`Bool / boolean`	Enables SNMP scan
`maxNetworkSize`	`NSInteger / integer`	Imposes a maximum network size
`resultLevelScanInProgress`	`FingScanResultLevel`	The level of results that shall be returned while the scan is in progress. One of FingScanResultNone, FingScanResultSummary, FingScanResultFull. The default is value is FingScanResultNone.
`resultLevelScanCompleted`	`FingScanResultLevel`	The level of results that shall be returned when the scan is complete. The default is value is FingScanResultSummary.
`resultLevelScanEnriched`	`FingScanResultLevel`	The level of results that shall be returned when the scan is enriched. The default value is FingScanResultFull
`outputFormat`	`NSString * / String`	The output format, expressed as MIME type. Currently only “application/json” is supported.

Please note that scan options are supported only on iOS at the moment.

4.The data structure of a Fing scan

Regardless of the platform being used, the FingKit returns the same set of results in the requested format. At the moment, JSON format is supported, which allow an easy integration with any kind of hosting app or process.
Since iOS 11, MAC addresses may not be retrieved for the local device and the scanned device, and are therefore not reported in the JSON result.

Summary dataset of the network

For the current network, Fing will provide a JSON data structure describing the network details and analyzed properties. This is the set of details returned at Summary level.

Key	Value	Example
nodes_count	The amount of nodes found in the network	12
nodes_up_count	The amount of nodes found online in the network. If the account is active, the state of network devices is preserved by the system and merged with the latest scan	10
nodes_down_count	The amount of nodes found offline in the network. If the account is active, the state of network devices is preserved by the system and merged with the latest scan	2
last_scan_timestamp	The time of the last scan	2016/11/23 02:00:07
network_short_address	The network address, in CIDR format	192.168.0.1/24
progress	The progress of the scan, in percentage fro 0 to 100	80
enriched	A boolean flag discriminating if this scan has been enriched by Fing Device Recognition service	true
completed	A boolean flag discriminating if this scan completes the scan progress. Depending on the license and enrichment, the last scan report may come from as an account update or as the last operation after the scan has completed	false

Extended dataset of the network

This is the set of details returned at Full level, in addition to all the details provided at Summary level. This structure is contained in the "network" JSON key.

Key	Value	Example
last_change_timestamp	The time of the last change	2016/11/23 02:00:07
gateway_ip_address	The IP address of the gateway	192.168.0.1
gateway_mac_address	The MAC address of the gateway	AB:00:DD:FF:01:CC
address	The network address	192.16.0.0

address_type	IPv4 or IPv6	IPv4
dns_address	The IP address of the DNS	192.168.0.1
mask_prefix_length	The netmask length applied by the scan engine, in bits	24
original_prefix_length	The netmask length as defined in the network, in bits	22
name	The network name from the Wi-Fi SSID, if any	My Network
bssid_list	A list of the access points BSSID	[“AB:00:DD:FF:01:CC”, “AB:00:DD:FF:01:CD”]
time_zone	The time zone of the scanning device	Europe/London

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
address	The public IP address	44.211.2.94
host_name	The public host name	host.viacom.com
country_code	The 2-letters country code	UK
country_code_3	The 3-letters country code	ITA
country_name	The name of the country	United States
country_region_code	The region code	LAZ
country_region	The region name	Tuscany
country_city	The city name	Washington
country_postal_code	The postal code of the address	W10 5BN
latitude	The latitude of the ISP point in decimal degrees	20.23123
longitude	The longitude of the ISP point in decimal degrees	-82.22938
isp_name	The name of the Internet Service Provider	AT&T
organization	The name of the organization providing Internet Access	Your Local Building
net_speed	The nominal network speed	40 Mbs

Network node base dataset

For each identified device, Fing will provide a data structure describing the network details and analyzed properties.

Key	Value	Description
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	“Laptop”, “Mobile”, “Photo Camera”.
best_category	A single major category the device falls in	“Entertainment” for a TV, “Personal” for a laptop, “IT” for a server

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”
best_os	The name of the Operating system, when applicable	“iOS 9.3.2”, “Android 5.0.1”, “Windows 7”.
mac_address	The MAC Address of the device that is currently using to connect to the network	“06:5c:89:c9:e7:d1”
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
ip_addresses	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”
host_name	The DNS name of the device	“mydevice.thissite.com”
state	Discriminates if the device is connected to the network or not. Can be “UP” or “DOWN”	“UP”
first_seen_timestamp	The timestamp the device was first discovered in this network	“2016-04-28 11:34:45”
last_change_timestamp	The timestamp the device changed the state (UP/DOWN)	“2016-04-28 11:34:45”

Network node extended dataset for NetBIOS

In addition to general-purpose properties, Fing exports for NetBIOS the following JSON structure, contained in the “`netbios`” JSON key.

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”

Network node extended dataset for Bonjour

In addition to general-purpose properties, Fing exports for Bonjour the following JSON structure, contained in the “bonjour” JSON key.

Property	Description	Example
name	The Bonjour name the device publishes	“My Macbook”
model	The Bonjour model the device publishes	“SCD8291221”, “Apple TV4,5”
os	The Bonjour Operating System name the device publishes	“Linux 12.4”
service_list	A list of bonjour services published by the device	“_afpovertcp._tcp.local.” “_smb._tcp.local.”

Network node extended dataset for UPnP

In addition to general-purpose properties, Fing exports for UPnP the following JSON structure, contained in the “upnp” JSON key.

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 extended dataset for SNMP

In addition to general-purpose properties, Fing exports for SNMP the following JSON structure, contained in the “snmp” JSON key.

Property	Description	Example
name	The SNMP name the device publishes	“HP
description	The SNMP description of the device	“Cisco IOS Software, C3750 Software (C3750-IPSERVICESK9-M), Version 12.2(46)SE”
location	The SNMP location of device	“North Corridor”
contact	The SNMP contact point	“admin@cisco.com”
sysoid	The unique identifier of the device type	“1.3.6.1.4.1.9.1.516”

Network node extended dataset for DHCP

In addition to general-purpose properties, Fing exports for DHCP the following JSON structure, contained in the "dhcp" JSON key.

Property	Description	Example
name	The DHCP name the device publishes	"My Macbook"
vendor	The DHCP vendor	"Samsung"

The type of devices the Kit recognizes

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

5. Integrate FingKit using Apache Cordova

FingKit may be used also through systems that rely on cross-platform mobile development toolkit based on Apache Cordova, such as Ionic 1 and 2, PhoneGap and similar. All such platforms rely on Javascript development and specific plugins that extend the core functionality to interact with additional frameworks.

In order for the plugin to work correctly, you shall use the default tools (npm) to install the system, the plugin and the dependencies mentioned in this document. A typical workflow includes the following steps:

- Install the basic system
  - npm install
  - bower install
- Install iOS or Android platforms
  - ionic platform add ios
  - ionic platform add android
- Install the FingKit plugin
  - ionic plugin add ./fingkit
- if not already present, **manually add Fingkit.framework in the embedded binaries of your Xcode project**
- Build for the target platforms
  - ionic build ios
  - ionic build android

The Apple App Store may require to strip the framework of the Intel-based architectures used for testing locally on the iPhone Simulator (i386 and X86_64). You may remove these architectures from the FingKit with the following commands

```
lipo -remove i386 FingKit.framework/FingKit -output
FingKit.framework/FingKit

lipo -remove x86_64 FingKit.framework/FingKit -output
FingKit.framework/FingKit
```

6. Integrate FingKit on embedded devices

FingKit embedded is as a command-line tool. The integration is designed to be flexible and loosely coupled with the hosting environment, leveraging standard mechanism of communications available on every platform.

To ensure decoupling of runtime and dynamic dependencies, FingKit Embedded runs as a separate command (or service, or daemon) in the embedding environment, thus isolating the execution of the command from the caller’s environment. The process is configured through a configuration file, and generates output to a destination folder.

How does it work?

FingKit can run continuously or for a configured number of rounds, generating output discovery files in the configured destination folder, one for each scan. The output discovery file format is JSON and follows the specs described in section 4. The File name is controlled by a configuration file, and by default it contains the date and time info.
The default configuration will generate files containing Year, Month, Day, Hour, Minutes, Seconds.

Default output file name example

```
discovery-20161104134534.json
```

Available platforms

FingKit is a cross-platform solution, supporting many of the most common platforms used in embedded and desktop devices. As the potential number of combinations is huge, the table groups the supported platforms in three categories: Kernel, Operating System and Package. Every combination of these option is supported.

Kernel	**Architecture**	**Operating System**	**Package Format**
Linux	Intel i686, amd64, arm, arm64, armhf, arm64hf, mips, mips64, mipsel, mipsel64	Debian, Ubuntu, Raspbian	.deb
Linux	Intel i686, amd64, arm, arm64, armhf, arm64hf, mips, mips64, mipsel, mipsel64	CentOS, Fedora	.rpm
Linux	Intel i686, amd64, arm, arm64, armhf, arm64hf, mips, mips64, mipsel, mipsel64	Other Linux flavors	.tar.gz
OpenWRT	See note (*)	OpenWRT	.ipk
Microsoft Windows	x86 (compatible with 64-bit processors)	Microsoft Windows	.exe
Darwin	X64	Apple macOS	.pkg

(*) Available for the standard target/subtarget of **Chaos Chalmer** version. Click here (https://wiki.openwrt.org/doc/howto/build#image_configuration) for further information.

Configuration of the FingKit

The kit is configured through a simple configuration file in the standard properties format, that is a list of `key=value` pairs, one for each line. Comments are supported by means of special character #. The backslash (\) char is sequence escaper.

Property	Description	Example
license	Your FingKit License key	ABCDEF123
network.interface	An optional interface name, to select a specific network interface; if not provided Fing will take the default one having a valid IPv4 address and network associated	eth0
rounds	An optional number of rounds Fing will execute; default is infinite	100
refresh.interval	The scan interval in milliseconds, default is 1 minute	60000
output.folder	The output folder where Fing will place output files	/users/home/fing-output
output.file.prefix	The file name prefix. The default is "discovery-"	discovery-
output.file.suffix	The file name suffix. The default is ".json"	.json
output.file.name	The output file name. It shall contain at least one of the following placeholder variables to generate a new file every time: • %Y = Year • %m = Month • %d = Day • %H = Hour • %M = Minute • %S = Second	%Y%m%d%H%M%S
output.file.name.timezone	The timezone to calculate timestamps with. The default is: UTC; use LOCAL for local device's timezone	UTC

Execution of FingKit

FingKit Embedded runs as a fing command and, requiring root/Admin privileges, because it leverages low-level network programming. You must make sure it runs as root user or with sudo.

Example of FingKit command-line execution on a linux-based device

```
> sudo fing --silent --kit /path/to/my/config
```

Linux/Rasp/Mac: Fing can also be started in background/daemon mode.
Windows: FingKit can be installed as a Windows service to automatically start when computer is booted.

Sample usage and Demo mode

FingKit supports a demo mode that doesn't require a license.

```
> sudo fing --kit demo
```

Please note that without a valid license key, FingKit works slower than normal, performing network activities at slower, random times. If a destination folder is not provided, FingKit write the output to the system temporary folder.

FingKit on Docker

FingKit can run also inside lightweight containers such as Docker, which is a self-contained, isolated images ideal for testing and for an easier and safer deployment.

Docker can generate an image on your system from a remote URL pointing to the tarball file on our website; please contact us for the latest URL and version number to ensure you will be using the latest version. On your hosting machine where docker is installed, you shall write the following command:

```
> docker import <url> registry.fing.io/docker/kit:<version>
```

You now have a ready-to-go docker image in your system. You just need to specify how the image will communicate with the hosting machine; being an isolated container, these inputs and ouputs must be "mounted", i.e. connected with the outside world.

File or Folder	Description	Folder inside Dockers
Configuration file	The input to configure the kit	/var/fing/kit.cfg
Scan output folder	The folder where scan results are written	/var/fing/log
Log folder	The folder where logs are written. Mounting this folder is optional	/var/fing/results

Your final Docker execution command shall specify how to bind these locations inside docker to locations into your hosting device, as in the example below.

```
docker run \
 --name fing-kit \
 --net=host \
 --mount type=bind,source=/location/on/host/kit/cfg,target=/var/fing/kit.cfg \
 --mount type=bind,source=/location/on/host/kit/out,target=/var/fing/results \
 --mount type=bind,source=/location/on/host/kit/log,target=/var/fing/log \
 registry.fing.io/docker/kit:<version>
```

You do not have to launch any command: you just need to run the docker mounting the proper directories on the host file system. Below the description of the command options:

- [--name] Specifying the container's name helps in identifying it inside your docker network. It's an *optional* parameter.
- [--net] The container requires the access to the host network to perform the scan. It's a *mandatory* parameter.
- [--mount] The container requires mounting points to bind inputs and outputs:
  - The kit file is expected to be in `/var/fing/kit.cfg` inside the docker file system. You shall mount this file from the host system (see first mount option). It's *mandatory*. The paths inside the `kit.cfg` file must refer to the docker file system. We suggest to keep `/var/fing/results` as output folder and `/var/fing` as kit folder and use the mount options to bind them with the proper folders in the host system.
  - The `output.folder` property in your `kit.cfg` should specify a mounted folder inside Docker file system, to export the results into the host system (see the second mount options). It's a *mandatory* parameter.
  - If you want to look at the fing-kit log on the host system, you may think to mount the log folder on the host system (see the third mount options). This parameter is *optional*.

Please refer to the official documentation of Docker for any further details.

www.fing.io