Fing Mobile SDK Developers Guide

3.8 - May 2021

1. Introduction

Device recognition technology is at the core of Fing App - the network scanning and troubleshooting tool downloaded over 35 million times and scanning millions of networks around the world every day. Fing's Device Recognition technology identifies billions of connected devices quickly and accurately and provides valuable insights about networks and connected devices. Development teams can save substantial time and resources by implementing Fing technology in applications for the connected world.

Fing's Development Toolkit consists of a Software Development Kit (SDK) for mobile apps and embedded devices that allow developers access to Fing's proprietary machine learning expertise and unrivaled device identification. The SDK analyses a device's Wi-Fi network and emits details about the connected devices and Fing's cloud service identifies specific device properties including make, model, category, operating system and version. Access a snapshot of currently connected and previously connected devices for a given network.

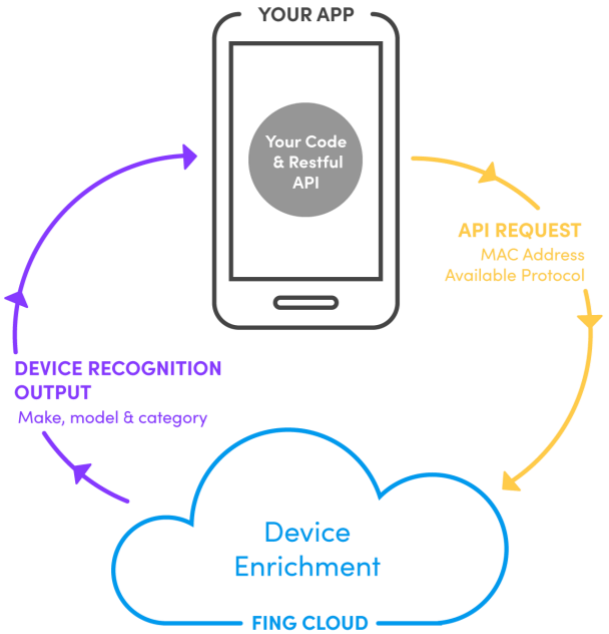
The key benefits for developers are:

- Improve existing user experience which relies on poorly implemented Device Identification, often developed internally.
- Accelerate time to market of products, apps and services that rely on accurate real-time connected device identification.
- Easily integrate non-invasive and light touch network discovery and device recognition capabilities without the need to create low-level networking technology and fingerprinting.

How it Works

Fing's proprietary technology leverages an ever-expanding crowdsourced knowledge to understand discovery protocols including Bonjour (Multicast-DNS) queries, UPnP queries, SNMP requests, DHCP monitoring and NetBIOS queries.

Fing's Device Recognition technology analyses and provides the best match data for each device. Fing's catalogue currently has more than one hundred types of device ranging from tablets to surveillance cameras. Device types are grouped into eight categories - mobile, entertainment, home & office, network, server, home automation, surveillance, engineering. In addition, Fing provides, in JSON text format, the full set of network details and analysis for each network protocol the devices comply with.



2. Integrate Fing Development Toolkit into your native app

Using Fing's developer tools is simple and straightforward on iOS and Android. Create an app, access your unique Fing License Key and simply add the relevant frameworks for Fing integration. You can then initialize Fing and start a network scan.

Integration with an iOS app

Fing SDK 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 applications written in Swift language. It is compatible with Apple iOS 9.x and greater.

Fing SDK requires the following items to be added in **"Linked Frameworks and Libraries"** in your XCode project.

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

Functionalities are accessed via the main singleton class FingScanner.

Requirements and Limitations

iOS 14.5+

Multicast Networking Entitlement

Beginning with iOS 14.5, an iOS app must obtain the Multicast Networking Entitlement to leverage multicast networking protocols. The entitlement can be requested by the project Account Holder. Instructions on how to configure your app to use the obtained entitlement can be found here. The entitlement must be declared in the app entitlements file (usually <appname>.entitlements) file using:

```
<key>com.apple.developer.networking.multicast</key>
<true/>
```

Without this entitlement, the FingKit cannot make use of multicast networking protocols to detect devices and network information such as Bonjour, UPnP. The lack of the Multicast Networking Entitlement also impacts the recognition of the device model and manufacturer.

iOS 14+

Local network access prompt

Beginning with iOS 14, the first time an iOS app attempts to access network information a modal prompt is presented, informing the user that local network data are going to be accessed. Accepting local network access is critical for Fing to detect connected devices information. The displayed message can be customized by adding the following in the App.plist:

```
<key>NSLocalNetworkUsageDescription</key>

<string>MyApp requires your consent to access to the local network and identify connected devices </string>
```

Because apps cannot intercept the user choice, it is best to prompt the user with the local network access dialog before initiating the network scan.

To present the popup is sufficient to initiate a local network request either using Objective-C or Swift.

iOS 13+

Location permissions

Beginning with iOS 13, an iOS app must obtain user Location Permissions to be able to detect Wi-Fi network name and address (SSID and BSSID).

iOS 12+

Access WiFi Information Entitlement

Beginning with iOS 12, an iOS app must obtain the Access WiFi Information Entitlement to be able to detect

Starting from iOS 13, the Apps that want to access Wi-Fi details (e.g Wi-Fi network name [SSID] and address [BSSID]) must include also the “Access WiFi Information”.

The entitlement must be declared in the app entitlements file (usually <appname>.entitlements) file using:

```
<key> com.apple.developer.networking.wifi-info </key>

<true/>
```

iOS 11+

MAC Addresses

Beginning with iOS 11, the possibility for third-party apps to read the MAC addresses of network connected devices has been removed. These MAC addresses, available in the ARP table of the device, have been changed with a fixed, locally administered MAC address (i.e. 02:00:00:00:00:00) which cannot be used to uniquely identify a device. The lack of MAC addresses also impacts the recognition of the device model and manufacturer.

However, even though MAC addresses are not available anymore in the ARP table of the device, the Fing discovery engine bundled in the FingKit is capable of detecting some of them by leveraging some network protocols in which the MAC addresses is published.

Integration within an Android app

The SDK 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. It is compatible with Android 5.0 and above. The following dependencies should be added in your Gradle-based or Maven-based project.

Group	Name	Version
androidx.appcompat	appcompat	1.1.0
com.google.android.gms	play-services-analytics	17.0.0
com.google.protobuf	protobuf-java	2.6.1
org.snmp4j	snmp4j	2.5.0
com.squareup.okhttp3	okhttp	4.8.0

The archive fing-kit.aar should 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>, place the library in <root/app/libs>) and it will be added as transitive compilation item in your build system.

Android Studio automatically includes the framework in the final package. Below is 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
    }

    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'com.google.android.gms:play-services-analytics:17.0.0'
    //noinspection GradleDependency
    implementation 'com.google.protobuf:protobuf-java:2.6.1'
    Implementation 'com.squareup.okhttp3:okhttp:4.8.0'
    implementation 'org.snmp4j:snmp4j:2.5.0'
}
```

The functionalities are accessed via the main singleton class FingScanner.

3. API Specification

Asynchronous design

Fing SDK 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);
}
```

Parameter	iOS/Android Type	Description
Result	NSString * / String	The result coming from Fing. It may be nil/null if there is no result or if 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.

The execution of the scanning request must be initiated from main thread in foreground execution on both platforms; executing the activity from a scheduled background task on Android or in the background mode on iOS may lead to inconsistent states and errors.

Error Handling

On Android, errors are represented as Exception objects passed as parameters. 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
-300	Scan operation failed

License Key validation

To enable the functionalities delivered by the Fing SDK, you must first obtain a License Key from sales@fing.com 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 Fing SDK.

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 API. The key is used to identify the 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)	A 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	Ok

	Suspended Revoked	
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	*Deprecated feature.* 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.

Network Info

Fing allows to conveniently retrieve network details from the Wi-Fi the device is connected to. The network details may be retrieved 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.

Starting from iOS 12.2 and Android 8.1, geolocation permissions are required to access Wi-Fi details (BSSID and SSID). The hosting App must acquire coarse location permissions before attempting to execute the scan or retrieve the network info.

If location permissions are not granted, the SDK may return a Locally Administered Address (usually 02:00:00:00:00:00). Hosting Apps can use the Boolean values “bssidIsLAA” and “bssidIsMasked” to respectively check

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
bssidIsLAA	A Boolean indicating if the BSSID has been generated by the current device and therefore not representing a real MAC Address. It could be anonymized, randomized or masking the real address	true
bssidIsMasked	A Boolean indicating if the real MAC address has been masked with a fixed value, usually 02:00:00:00:00:00	false
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

This functionality is accessed via a single method that performs the scan and enrichment of data, if enabled. The scan is integrated with the 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;
-(void) networkScan:(nullable FingScanOptions *) options
     withScanInput:(nullable FingScanInput *) input
        completion:(nullable FingResultCallback) completion;
-(void) networkScanStop;
```

Android (Java)

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

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	iOS/Android Type	Description
options	FingScanOptions * (Optional)	The set of options to tune the network scan procedure. See Scan Options for details

input	FingScanInput (Optional)	An external input that drives the scan procedure (iOS only). Hosting Apps may optionally provide the IP Address to MAC address mapping of devices in a network, using data from an external source such as private API from a router. The structure of the Input is explained in the sections below. If provided, the Fing Scanner engine will use the provided data as the source of network truth, ignoring any information the local ARP table may provide. IP Addresses must match the current network's netmask; non-matching IPs will be discarded.
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.

Stopping a Scan

The scan can be stopped at any time using the corresponding method networkScanStop; if the scan was not running at that time, nothing is performed. After the stop operation is requested, all pending updates and the enrichment update will be delivered to scan completion handler.

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.

Scan Input

You may provide an externally obtained table of data through a custom Scan Input object. The scan input is made of a list of devices FingScanDeviceEntry objects, each representing a mapping of additional data for a device, identified by its MAC address. The structure of the entry is as follows:

Option	iOS/Android Type	Description
ipAddress	NSString * / String	The IPv4 Address of the device, in dotted decimal format such as 192.168.1.1. IP Addresses not matching the network’s netmask will be discarded.
macAddress	NSString * / String	The MAC Address of the device, in hexadecimal format such as AB:00:12:34:00:FF. MAC addresses are used to identify devices uniquely in the network

		and to correctly represent the timeline of online/offline status
hostName	NSString * / String (Optional)	The device hostname, as administered by a local DNS, if any. This detail is optional.

Please note that scan input is supported on iOS only at the moment.

4. Data structure of a Fing scan

Regardless of the platform being used, the Fing 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 number of nodes found in the network	12
nodes_up_count	The number of nodes found online in the network. The state of network devices is preserved locally by the system and merged with the latest scan	10
nodes_down_count	The number of nodes found offline in the network.	2

	The state of network devices is preserved locally by the system and merged with the latest scan	
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 from 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 the last operation of the scan 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 latitude of the ISP point in decimal degrees	W10 5BN
latitude	The longitude of the ISP point in decimal degrees	20.23123
longitude	The time zone of the scanning device	-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	Example
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	“Apple”, “Belkin” (but not “Foxconn”)

	case the network interface (ETH, WIFI) is different.	
best_model	The human-readable name of the model	“iPhone 5S”, “P9”
is_family	Flag advising if the model is a generic family rather than a specific model.	true
best_os	The name of the Operating system, when applicable	“iOS 9.3.2”, “Android 5.0.1”, “Windows 7”.
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. Higher values indicate a more confident device recognition	95
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 that Fing 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 evolves.

Group	Description
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 Fing using Apache Cordova

Fing's Development Toolkit may also be used 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 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

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:

```
lipo -remove i386 FingKit.framework/FingKit -output FingKit.framework/FingKit
lipo -remove x86_64 FingKit.framework/FingKit -output FingKit.framework/FingKit
```

6. Open-Source software

Fing relies on third-party libraries and tools distributed in accordance with Open Source distribution licenses. Below the list with the corresponding license

Fing SDK for iOS

Library	License
CocoaAsyncSocket	Public Domain or BSD License
DTFoundation	2-clause BSD license
XMLDictionary	Custom
Google Protocol Buffers	Creative Commons Attribution 3.0

Fing SDK for Android

Library	License
Android Support AppCompat	Apache License 2.0
SNMP 4J	Apache License 2.0
Google Protocol Buffers	Creative Commons Attribution 3.0
Apache HTTP Client	Apache License 2.0

Apache HTTP Mime	Apache License 2.0