Intro to the goTenna SDK

goTenna is the world's first wireless platform for long-range, mobile, off-grid Mesh networking. The goTenna SDK allows you to develop applications that require direct networking using Android and iOS smartphones. Using the SDK & goTenna hardware, you can build applications that work even when traditional mobile network infrastructure is down (e.g., during an emergency) or where it doesn't exist at all (e.g., when hiking in the countryside). Alternate options that enable direct networking, such as Bluetooth and WiFi may offer higher data bandwidth, but are extremely limited in distance. With point-to-point range often measured in miles and the ability to easily double or triple your range by meshing, the goTenna platform may be a much more practical option for these applications.

Although our own app focuses on text messaging and GPS location sharing, we hope the developer community will come up with new ways of utilizing the goTenna platform to deliver unique applications that work anywhere. The goTenna SDK allows developers to transmit any kind of data over a goTenna network.

Primary Functional Features

- Support for Mesh and Pro X
- Open broadcast messaging
- 1-to-1 private messaging
- Group messaging
- Meshing for private messages and group messages (see table below

Message Type	Maximum number of hops
Private	3
Group	3
Shout	1
Emergency	1

- Frequency hopping
- Completely free-form agnostic data bucket for you to do anything you want, with 2 simple fair-use rules
    - 235 byte maximum payload size for all transmission types
    - 5 total transmissions allowed per end-user per minute (this applies to only for Mesh)

Other things the SDK will do for you

- Manage Bluetooth connectivity for iOS or Android
- Report on receipt or failure for 1-to-1 messaging
- Option to activate goTenna encryption within 1-to-1 and group messaging
- Report on goTenna battery life and firmware version
- Manage firmware upgrades
- Provide alerts and other diagnostic data as needed
- Maintains separation from other SDK users. To illustrate, App X can only speak to App X on goTenna via a tokening process, apps cannot be interoperable for security reasons. If you misplace your token, please contact goTenna support to retrieve it.

Why is the SDK the way it is? Many of you might wonder why the payload and bandwidth limits are in place. This is purely because spectrum is a finite resource, and the FCC tightly controls the amount of spectrum available for use in the unlicensed frequency bands we operate at. For this reason, we all have to be mindful of our network footprint and put in place some controls so that we don't ruin the airwaves for other SDK users. If we allowed non-stop transmissions, not only would it likely destroy the hardware, but instead of us being able to support many people in an area, we would only be able to support 1 or 2. So we ask that everyone please be considerate of how you decide to use goTenna. Building an app that just does HD video transfer across a city for fun may technically work if you're patient enough, but it would ruin the experience for anyone else in the area. So play nice please!

Development Design Tips

- Keep in mind the purpose of goTenna, which is enabling short-form burst data on shared channels at long distances. Although larger things can be done with the SDK, it is not what the system is designed for. Using the goTenna for applications which require large throughput will result in lower battery life, less bandwidth for others (the airwaves are shared), latency, and generally just not work well. Plus, it is not nice to others.
- Encode your data as often as you can for small size optimization. If you have certain pre-set types of information you want to send, you're much better off sending a small burst that represents that data as opposed to the raw data itself. For example, need to have an "Alert = Arriving" or "Alert = Emergency" in your app? Instead of spelling everything out, consider a system of flags and structured data so that you can send something like, "A:1" or "A:2" and then interpret it on the other end!
- Find any opportunity to cut down on your own payload's structural overhead.
- Pre-cache data within your applications whenever possible. The goTenna is best when used to send beacons of information like changes, alerts, or other small bursts that can then be contextualized richly using information that was previously available on the other user's app.

- Keep in mind that Bluetooth Low-Energy has a very low data throughput, so at times it may be a bottleneck in your communications back and forth from goTenna. Delays may sometimes be from this root cause.
- Try to avoid any communications architectures where you create a situation where many nodes are trying to communicate at the exact same time. For example, a poor architecture would be one where 1 unit sends out a polling message, which then results in every other unit in the area trying to respond simultaneously in some way, this can result in latency and lost packets. A better architecture would be one where units are temporally spread out when trying to communicate to a single unit.
- Important: We strongly encourage you to use the shout, 1-to-1, and group chat functions as they are natively designed. Shout broadcasts are NOT encrypted.

Hardware Support

- The goTenna SDK offers support for the goTenna Mesh and goTenna Pro X devices.
    - The goTenna Pro is not supported at this time.
- The goTenna Mesh device is a consumer-focused radio designed for civilian use when outdoors, traveling abroad, or any time that a traditional cellular or WiFi connection is unavailable.
- The Pro X is a commercial and military-grade device designed for off-grid communications and requires an FCC license to operate.
    - Users of the Pro X must operate their devices only on the frequency specified by their license.
    - SDK developers supporting the Pro X must ensure their app allows users to set the Frequency, Power Level, Bandwidth, and Channel of their goTenna device in accordance with their FCC license. Failure to do so will result in users not being able to operate their goTenna Pro legally.

Other Rules

- SDK Terms of Use are included in the repository, please read thoroughly.
- It is the responsibility of the developer to comply with all FCC regulations and ensure users can operate their goTenna in compliance with FCC regulations.
- goTenna reserves the right to revoke SDK access to any application that is found to be abusive and against the spirit of what this tool is designed to be used for, example of such violations include:
    - Patterns of continuous transmissions (large file transfers or any other operations which consistently max out the limits of the SDK on bandwidth use).
    - Excessive automated beaconing which noticeably degrades other goTenna users' quality of service.
    - Applications violating FCC rules.
    - Any evidence/attempts to circumvent the fair-use controls within the SDK protecting public bandwidth.

    - Additional reasons are outlined in the SDK Terms of Use.

- SDK developers must agree to abide by goTenna brand guidelines, specifically:
    - Your primary product identifier should not be the goTenna icon or name. or example:
        - Do not name your app "goTenna"
        - Do not make the public app store listing icon be the goTenna icon
        - Do not make the home-screen shortcut icon be the goTenna icon
    - That fact that your app/plugin is not created by goTenna Inc. must be made clear in any listing of your development product marketplace, as well as within the product's info/about screen. References to goTenna must explicitly use the phrase “Built on goTenna.”