

**IoT Tracker Architecture**

08 June 2023

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes made |
| V1 | 8 June | Document Created |

|  |  |  |
| --- | --- | --- |
| **Prepared by:** | **Richard Davies** | **Electro Rent** |
|  |  |  |
| **Approved by:** |  |  |
|  |  |  |
| **Distributed to:** | **Internal Use** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[1. Introduction 3](#_Toc138062758)

[2. Overview 3](#_Toc138062759)

[3. Tracker 4](#_Toc138062760)

[3.1Configuration 4](#_Toc138062761)

[3.1.1 Operating Modes 4](#_Toc138062762)

[3.2Configuration page 4](#_Toc138062763)

[3.3Commands 4](#_Toc138062764)

[4. Server 5](#_Toc138062765)

[4.1Overview 5](#_Toc138062766)

[4.2Security 5](#_Toc138062767)

[4.2.1 Authentication 5](#_Toc138062768)

[4.2.2 Rate limiting 5](#_Toc138062769)

# 1. Introduction

This overview will delve into the architecture of our proposed IoT tracker system, highlighting its key functional and technical aspects. The tracker is a compact device, easily attached to any equipment and offering reliable, precise location data. The device has an expected battery life of one year and is highly configurable. It uses mobile networks to directly transmit location information to our dedicated server, avoiding any third-party involvement.

# 2. Overview

The system employs an IoT tracker that combines GPS and Wi-Fi and utilizes Narrowband-Internet of Things (NB-IoT) and Long-Term Evolution for Machines (LTE-M) networks for uploading current location to a server. This server will reside on a Virtual Machine (VM) hosted on Oracle Cloud Infrastructure (OCI). The tracker can be configured remotely by sending SMS messages and we can integrate this functionality by using the APIs offered by most SIM card providers.

Important to note is that the IoT tracker sends data over TCP to a specific IP address and port number. This means we require a server with a static IP address to be our cloud gateway for the trackers. Other common methods such as cloud services/ lambda functions can’t be used directly as the IP addresses are ephemeral (not fixed).

****

# 3. Tracker

### 3.1Configuration

The tracker can be configured either remotely via SMS or directly through a USB cable. The initial setup must be done either by us as the SIM APN needs to be set before SMS will work.

|  |  |  |
| --- | --- | --- |
| **Setting** | **Description** | **Example/ Notes** |
| APN | Access Point Name – Internet Gateway | pepper (Things Mobile) |
| IP & Port | IP address and port for uploading data | 10.32.101.9:8080 |
| Protocol | Transfer protocol – TCP or UDP | TCP preferred |
| AGPS | Assisted Wi-Fi location – ON or OFF | On |
| LBS | Location based services | On |
| NWM | Network Mode | NB-IoT or LTE-M |
| BAND | Frequency band to use for data connection | Default is all bands |
| MODE | 6 operating modes with various parameters | See below |
| HBC | Starts periodic heartbeat message | [5-60] minute interval |
| MSW | Set SOS power button – ON or OFF | On/off |
| Last location report | Check? |  |

#### 3.1.1 Operating Modes

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Notes** |
| MODE,1,T | Device will report data every Ts. | T: [60-600] seconds  **GPS and TCP always ON** |
| MODE,2,T, X, Y | Turn off GPS when not using | T: [10-60] minutes  X: GPS keep alive  Y: TCP keep alive |
| MODE,3,T | Deep sleep with periodic wakeup | T: [1-24] hours |
| MODE,4,T | Deep sleep until motion detected | T: [60-600] seconds |
| MODE,5,T, X, Y | Wi-Fi only mode. Fixed X = 0 | T: [1-60] minutes  Y: TCP keep alive |
| MODE,6 | SMS only mode | Text “WHERE0000” for current location |
| MODE,0,T1,T2 | Mode 3 until motion detected. Then mode 4 | T1: [60-600] seconds  T2: [1-24] hours |

### 3.2Configuration page

We will need to host a webpage that allows users to configure the trackers. At its simplest, this is a HTML form which can call the required HTTPS endpoints and trigger the required SMS messages to be sent. A login page should also be implemented if we intend of giving customers access to this.

### 3.3Commands

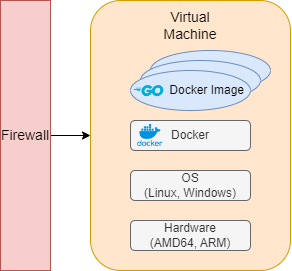
Also triggered via SMS.

|  |  |  |
| --- | --- | --- |
| RCONF, X | Read configuration | X range is [1,3] |
| REBOOT | Reboot device |  |

# 4. Server

### 4.1Overview

The IoT trackers can only send location data to a specific IP address. This necessitates the use of a gateway server with a static IP that can forward data to our main servers using HTTPS.

This gateway server will be housed on a Virtual Machine (VM) and run a Golang application deployed within a Docker container. Golang is an easy to understand general purpose language that was designed for servers. Features include built-in rate limiting and go routines, which offer safe concurrency and prevent data races from occurring. The use of Docker abstracts behavior from the underlying operating system and hardware on the server. This ensures consistent behavior across different environments, simplifies version control, and facilitates easy rollbacks, if necessary. For example, if a piece of software needs to be updated, we build a new docker image with the updated software and swap it with the current image on the server. If something goes wrong, we can easily swap back to the previous container. Docker also supports easy scaling of our service in response to increasing demand. As demand increases, we simply start up an extra instance of the container. If traffic ever increases to the point that the server is at max capacity, then we will have to implement a reverse proxy or some other type of load balancing server, which will forward data to multiple VMs. Oracle OCI offers a range of products for this, and it can be added to the current system without any changes to how the docker image runs.

### 4.2Security

The main weaknesses of this approach are:

1. Can’t authenticate trackers unless static IP addresses are assigned for each device.
2. Server is left open and vulnerable to Denial-of-Service (DoS) attacks.

#### 4.2.1 Authentication

This refers to validating the identity of a connection before accepting the data is important for maintaining data integrity and security. In our case, anyone who discovered the IMEI of a device could send fake to our server very easily. This can be solved by assigning a static IP address to each device. We can then maintain a whitelist of known IP addresses and reject any connections coming from other sources. Most SIM card providers offer this functionality. On the server side, it just requires a firewall to filter out connections from unknown IP addresses.

#### 4.2.2 Rate limiting

If a device or malicious actor were to start sending lots of traffic to our gateway server, it could overload the system and cause legitimate packets to be missed. This is called a Denial-of-Service (DoS) attack and is a very common attack. Rate limiting mitigates DoS threats by preventing any given traffic source from sending too many requests per second. Golang has features for rate limiting built-in and can efficiently block any devices that are sending too much traffic.