GPS System Technical Document

Version No	Date	Contributors	Status
1.0	23-07-10	AnLe	Closed
1.1	23-10-18	AnLe	Processing

Table of contents

I. Missions and visions

This is a start-up product by the AnLe team.

The device can be a smartphone (Android/ iOS/OEM) which can be easy to install our application.

Customers can view the real-time GPS of the device via smartphone or website.

In the future, our product can be deployed as a Cloud Service Platform.

Any developer can use our SDKs to build their system.

II. List of features

No	Name	Developer	Estimate time (working hours)	Status
1	Send GPS from device to Back-end	AnLe	40	Idle
2	Design GPS packet structure	AnLe	40	Idle
3	Identity device on packet	AnLe	40	Idle
4	Security and checksum packet	AnLe	40	Idle
5	Storage GPS to database	AnLe	40	Idle
6	Visualize GPS history of device on Android	AnLe	40	Idle
7	Visualize GPS history of device on iOS	AnLe	40	Idle
8	Visualize GPS history of device on Website	AnLe	40	Idle
9	User profile	AnLe	40	Idle
10	User update information	AnLe	40	Idle
11	Authenticate the device with ID	AnLe	40	Idle
12	Interval tick packet	AnLe	40	Idle

III. Use cases

1. Send GPS from device to Back-end

Use Case ID	GPS.Device.1
Use Case Name	Send GPS from device to Back-end
Description	Device must send GPS to BE 3 times per minute.
Priority	High
Primary actor	Device
Secondary actor	NA
Preconditions	Authorized device
Basic flow	
1. The device sends a GPS packet to the Back-end.	
Alternative flows	
1. The device doesn't authorize can't send a GPS packet to the Back-end. 2. The back end denies storing GPS packets from the device to the database.	
Post condition	
1. The device receives a GPS response packet.	
Extension points	
No	
Exceptions	
1. The device receives a denied packet from the back end.	
Related use cases	
No	

2. Design GPS packet structure

3. Identity device on packet

4. Security and checksum packet

IV. Packet design

1. Design GPS Login packet structure

Starting: 2 bytes - ushort (using for define device model id - at the starting of packet - example: 0x79 0x79)

Packet id: 2 bytes - ushort (using for define packet type - example: 0x01 0x01)

Device uid: 6 bytes - string (example: 0x01 0x02 0x03 0x04 0x05 0x06)

Packet order index: 2 bytes - ushort (start from 1, plus 1 per each next packet - example: 0x00 0x01)

Additional Data (if need): N bytes - TBD (explain more at gps packet)

Checksum number: 2 bytes - ushort (using crc32 for checksum from start bytes to data bytes)

Ending: 2 bytes - ushort (using for define the end of packet - example: 0x00 0xFF)

2. Design GPS Login response packet structure

Starting: 2 bytes - ushort (using for define device model id - at the starting of packet - example: 0x79 0x79)

Packet id: 2 bytes - ushort (using for define packet type - example: 0x01 0x02)

Device uid: 6 bytes - string (example: 0x01 0x02 0x03 0x04 0x05 0x06)

Packet order index: 2 bytes - ushort (start from 1, plus 1 per each next packet - example: 0x00 0x02)

Received packet index: 2 bytes - ushort (the received index of last packet example: 0x00 0x01)

Checksum number: 2 bytes - ushort (using crc32 for checksum from start bytes to data bytes)

Ending: 2 bytes - ushort (using for define the end of packet - example: 0x00 0xFF)

3. Design GPS Information Packet structure

Starting: 2 bytes - ushort (using for define device model id - at the starting of packet - example: 0x79 0x79)

Packet id: 2 bytes - ushort (using for define packet type - example: 0x02 0x01)

Device uid: 6 bytes - string (example: 0x01 0x02 0x03 0x04 0x05 0x06)

Packet order index: 2 bytes - ushort (start from 1, plus 1 per each next packet - example: 0x00 0x01)

Latitude: 4 bytes - int32 (latitude record as heximal - 10.777624 * 10^6 = 10777624 - example: 0x00 0xA4 0x74 0x18 = 10777624d)

Longitude: 4 bytes - int32 (longitude record as heximal - 106.711670 * 10^6 = 106711670 - example: 0x06 0x5C 0x4A 0x76 = 106711670d

Checksum number: 2 bytes - ushort (using crc32 for checksum from start bytes to data bytes)

Ending: 2 bytes - ushort (using for define the end of packet - example: 0x00 0xFF)

4. Degis GPS Information Response Packet structure

Starting: 2 bytes - ushort (using for define device model id - at the starting of packet - example: 0x79 0x79)

Packet id: 2 bytes - ushort (using for define packet type - example: 0x02 0x02)

Device uid: 6 bytes - string (example: 0x01 0x02 0x03 0x04 0x05 0x06)

Packet order index: 2 bytes - ushort (start from 1, plus 1 per each next packet - example: 0x00 0x01)

Received packet index: 2 bytes - ushort (the received index of last packet example: 0x02 0x01)

Checksum number: 2 bytes - ushort (using crc32 for checksum from start bytes to data bytes)

Ending: 2 bytes - ushort (using for define the end of packet - example: 0x00 0xFF)

V. Software design

1. Technical requirements

No	Technical name	Description	References
1	Dependency injection (IoC)	Must have this technique Helping for clean code Easier for unit test Parallel test Mock test	DI for dotnet
2	Unit test	Must coverage at least 80%	
3	Mock test	Same with unit test	
4	ReactiveX	Good knowledge should have	
5	Concurrency & Parallelism	Good to have Safe threading	
6	Observable	Good to have Subscribe flows Avoid callback hell, good tips	
7	Material design	Good design concepts	material design

2. References