|  |
| --- |
| **CONFIDENTIAL** |
| **Incorrect Proximity Beacons Detection Tool**  **Integration and deployment guide**  **Revision: 1.0** |

|  |
| --- |
| **Document Code** |

|  |
| --- |
|  |

|  |
| --- |
| Trusted Positioning Inc.  A TDK Group Company  #405, 1000 Veterans Place NW  Calgary, AB Canada T3B 4M1 |

|  |  |
| --- | --- |
| REVISION: 1.0 - ISSUED FOR USE | |
|  |  |
| **PERMIT STAMP** | **ENGINEER’S STAMP** |
|  | |

A picture containing shape

Description automatically generated

# Table of Contents

[Table of Contents 3](#_Toc74666939)

[Version Log 4](#_Toc74666940)

[Figures 5](#_Toc74666941)

[Tables 5](#_Toc74666942)

[Equations 5](#_Toc74666943)

[Acronyms 6](#_Toc74666944)

[1 General tool description 7](#_Toc74666945)

[2 General architecture 8](#_Toc74666946)

[3 Data processing steps 9](#_Toc74666947)

[3.1 Beacon data preparation 9](#_Toc74666948)

[3.2 Create model data from positioning logs 9](#_Toc74666949)

[3.3 Create dat-files from model data 9](#_Toc74666950)

[3.4 Prepare xblp-venue.json with JA-tool 10](#_Toc74666951)

[3.5 Run FpBuilder in xBLP mode 10](#_Toc74666952)

[3.6 Run Proximity Beacon Placement Error script 11](#_Toc74666953)

[3.7 Ignore list json structure. 12](#_Toc74666954)

[4 Usage examples 13](#_Toc74666955)

[4.1 Test1 13](#_Toc74666956)

[4.2 Test2 13](#_Toc74666957)

[4.3 Test3 14](#_Toc74666958)

[4.4 Test4 14](#_Toc74666959)

[5 Appendices 15](#_Toc74666960)

[5.1 Appendix A 15](#_Toc74666961)

## Version Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date  (D/M/Y/ | Author | Checker | Update Description |
| 0.0 | 10/06/2021 | V. Pentiukhov |  | Draft |
| 1.0 | 15/06/2021 | D. Churikov |  | Initial version |
|  |  |  |  |  |

## Figures

**No table of figures entries found.**

## Tables

**No table of figures entries found.**

## Equations

**No table of figures entries found.**

## Acronyms

|  |  |
| --- | --- |
| Abbreviation | Definition |
| BLE | Bluetooth Low Energy |
| BLP | BLE-proximity |
| JA-tool | Json Automation tool |
| FPBL | Fingerprint Builder Library |

# General tool description

The Incorrect beacon detection tool assesses location and parameters of proximity beacons installed in venue and detests improper beacon installation or configuration parameters which can cause to positioning inaccuracy.

Current version of script detects the following improper situation:

1. Installation beacon to incorrect position or specifying incorrect position in BLP-database.
2. Twin beacons detection.
3. Mismatching of Tx power in beacon settings and in database.

The tool consists a few steps. A script or an application is running in each step.

The tool utilizes positioning logs collected during positioning sessions in the venue and produces ignore list json file which contains an information about improper proximity beacons found during the assessment. Currently this information should be displayed via Venue Dashboard.

The ignore list json file can be also used in following fingerprint buildings for disabling incorrect beacons in proximity database. This step will be realized after end of evaluation period of the tool.

# General architecture



# Data processing steps

All scripts and application described below are contained in Gift repo, branch master\_incorrect\_beacon\_detection if another is not specified.

## Beacon data preparation

Select a time interval for datasets analysis and select datasets which match to the interval

Copy selected datasets in a folder for further processing. This data is the input data for next steps.

## Create model data from positioning logs

For generating model data from positioning logs use script Gift\Tools\create\_model\_files\_from\_positioning\_logs\main\_create.py.

The script process positioning data in specified folder and creates model data files in each dataset folder.

Input data for this script are proximity database venue.blp4 (venue.blp3) and real time logs:

* Mixed positions log - \*fpp\_mixed.txt
* Blp log - \* proximity\_log\_c.txt

Output model data of the script are:

* irl\_data\_mt\_converter.txt
* blp\_debug\_converter.log

These files are placed in the same folder as input data located.

Example:

python <working folder> main\_create.py INPUT\_FOLDER venue.blp3

Note: Take venue.blp3 file from the fingerprint which was used in positioning during the selected time interval.

## Create dat-files from model data

For creating tpn-dat files from model data use the script main\_converter.py located in Gift\Applications\ModelDataToTPNConverter\scripts\.

This script uses Gift\Applications\ModelDataToTPNConverter application for converting the model data from previous step to TPN-like formatted data.

Output data for this script is tpn.dat file within the same folder as input data located.

Example:

python <working folder> main\_converter.py -i INPUT\_FOLDER – c MODELDATATOTPNCONVERTER\_FILE

where MODELDATATOTPNCONVERTER\_FILE is full name of ModelDataToTPNConverter application.

The ModelDataToTPNConverter application can work under Windows or Linux. Use makefile in the application project folder build with Linux.

## Prepare xblp-venue.json with JA-tool

Generate venue\_xblp.json file required in the next step with JA-tool.

* Take venue.json file from the latest fingerprint batch built for selected interval.
* Run JA-tool with specifics-xblp.json as specific file and the venue.json as template. Use the same coordinate and floors files as for regular JA tool run for the venue.
* Rename venue.json file, which is output of JA-tool, to venue\_xblp.json.

JA tool location:

* + Git repo: <https://github.com/InvenSenseInc/Coursa-Venue-AutoMap>
  + Branch: master\_ja
  + Path: Coursa-Venue-AutoMap\json\_automation\

Use specifics-xblp.json from Coursa-Venue-AutoMap\json\_automation\specific\_files\.

Example:

python ja\_main.py -t venue.json -v specifics-xblp.json -f floors.json -c coordinates.txt

As an alternative it is possible to merge specifics-xblp.json in to venue.json.

Result of this step is venue\_xblp.json file to use below.

## Run FpBuilder in xBLP mode

Use tpn.dat files as an input data for FP\_Builder to build ble4 fingerprint database with proximity beacons data.

Do the following actions:

In venue\_xblp.json:

Set “folder\_in” to folder where selected datasets with tpn.dat files located.

Set “folder\_out”, “folder\_grid” and “folder\_fp” to output folder.

Run FpBuilder with venue\_xblp.json as settings file and with --xblp\_detection key.

Example:

FP\_builder.exe --settings venue\_xblp.json --xblp\_detection on

Output is BLE-fingerprint (venue.ble4) build using proximity beacons data blp\_scaning\_log.csv file. These files will be used as input for the next step.

Use FpBuilder from master\_b branch of Gift.

## Run Proximity Beacon Placement Error script

Current version of script detects the following improper situation:

1. Installation beacon to incorrect position or specifying incorrect position in blp-database.
2. Twin beacons detection.
3. Mismatching of Tx power in beacon settings and in database.

In result ignore list json-file is generated. This file contains a list of incorrect beacons found.

Script location: Gift\Tools\proximity\_beacons\_placement\_error\main\_manager.py

The script receives the following parameters in command line:

-sv or --settings\_venue specify json file with venue settings (venue.json)

-blp or --blp\_db specify proximity data base in blp4 format

-ble or --ble\_db specify ble4 fingerprint built for proximity beacons

-blpl or --blp\_log specify proximity beacons log file

-blpi or --output\_ignore\_list set json file to output proximity beacons ignored list

-se or --settings\_extra specify json file with script settings (optional); this can be default settings\_extra.json from the script folder or local settings\_extra.json file from venue folder if provided

Example:

python <working folder>main\_manager.py

-sv venue\_xblp.json

-se extra\_setting.json

-blp venue.blp4

-ble venue.ble4

-blpl blp\_scaning\_log.csv

-blpi blp\_ignored\_list.json

## Ignore list json structure.

The tool creates/updates ignore list json with blp\_ignore\_list\_autogenerated section which contains a list of incorrect beacons detected and has the following structure:

{

"blp\_ignore\_list\_autogenerated": [

{

"uuid": "f7826da6-4fa2-4e98-8024-bc5b71e0893e",

"major": 2091187128,

"minor": 29,

"hash": 2091187128,

"rejection\_reason": "placement"

},

{

"uuid": "f7826da6-4fa2-4e98-8024-bc5b71e0893e",

"major": 2091187128,

"minor": 30,

"hash": 2091187128,

"rejection\_reason": "twin"

},

. . .

]

}

“Uuid”, “major” and “minor” fields identify incorrect beacon, “hash” means beacon hash number used for beacon identification in ble4 file and “rejection\_reason” denotes the reason of rejection.

“rejection\_reason” can have one of the following values:

* “placement” means beacon’s position in blp-database does not match to real beacon location
* “twin” means that at least two different beacons with the same UUID, major and minor where found in the venue
* tx\_setting denotes that beacon’s Tx power specified in BLP-database does not match to Tx power transmitted by the beacon

Currently this information should be displayed via Venue Dashboard.

The ignore list json file can be also used in following fingerprint buildings for disabling incorrect beacons in proximity database. This step will be realized after end of evaluation period of the tool.

# Usage examples

There are four scripts which imitates of Incorrect beacon detection process according to the diagram provided in section2. These examples/tests are realized as bat-files for Windows. They are located in [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_scripts\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_scripts\).

## Test1

This test realizes Create model data from positioning logs and Create dat-files from model data steps.

Input data: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test1\input\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test1\input\)

Reference results: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test1\ref\_data\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test1\ref_data\)

## Test2

This test realizes Run FpBuilder in xBLP mode step.

Input data: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test2\input\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test2\input\)

Reference results: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test2\ref\_data\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test2\ref_data\)

## Test3

This test realizes Run Proximity Beacon Placement Error script step.

Input data: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test3\input\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test3\input\)

Reference results: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test3\ref\_data\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test3\ref_data\)

## Test4

This test imitates whole process of incorrect beacon detection.

Input data: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test4\input\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test4\input\)

Reference results: [\\cayyc-proj01\compute02\FPL\_DATA\test\_data\xBlpDetection\test\_data.src\Test4\ref\_data\](file:///\\cayyc-proj01\compute02\FPL_DATA\test_data\xBlpDetection\test_data.src\Test4\ref_data\)

# Appendices

## Appendix A