FP Builder console interface description

FpBuilder version 1.6.0

July 21, 2021

The document contains FP builder interface description.

# Fingerprint builder input and output data flows



# Command line keys of FP\_Builder

Command line format: FP\_builder.exe -- settings venue.json [--key key\_value]

The following command line keys are supported

#### settings

The command specifies json file which contains FP\_builde settings.

Usage example: FP\_builder -- settings venue.json

venue.json – json file with setting and fingerprints parameters.

#### ignore\_list

The command specifies json file which contains ignore and/or white lists for WiFi and/or BLE fingerprints and proximity database.

Usage example: FP\_builder -- settings venue.json --ignore\_list ignore\_list.json

ignore\_list.json – json file with ignore/white lists. See description below.

#### xblp\_detection

The command activates special FP\_builder mode for procedure of incorrect proximity beacon placement detection.

Usage example: FP\_builder -- settings venue.json -- xblp\_detection on

#### input\_list

#### < to be supported>

The command disables automatic search for datasets in location provided with setting “folder\_in” of venue.json. FP\_builder utilizes datasets provided in dataset\_list.json only.

Usage example: FP\_builder --input\_list dataset\_list.json

dataset\_list.json – json file with list of datasets to be processed. See description below.

#### self\_healing

#### <under debugging>

The command activates special FP\_builder mode for wifi self-healing procedure.

Usage example: FP\_builder -- settings venue.json -- self\_healing on

1. Input Files for FP Builder

# Settings json-file

Settings json-file contains all necessarily fingerprints, grids and fp\_builder settings applied in fingerprint building and ASCA processing.

See description of the supported settings in “description\_of\_venue.json”.

See some examples of typical setting json-files in section 4 together with related build configuration description.

# WiFi and BLE Ignore lists

As ignore as white lists of WiFi access points are defined in json file specified with --ignoge\_list command line key.

A structure of each white or ignore lists is the following:

"<list\_name>":

[

"<mac\_mask\_1>",

. . .

"<mac\_mask\_i>",

. . .

"<mac\_mask\_n>",

]

Where “<list\_name>” defines the list and can be one of the following:

“wifi\_white\_list” or “wifi\_ap\_white\_list“ – defines wifi white list,

“wifi\_ignore\_list” or “wifi\_ap\_black\_list “ – defines wifi ignore list,

“<mac\_mask\_1>” defines MAC address or MAC address maskin the format "XX:XX:XX:XX:XX:XX:XX:XX".

The mask can contain 6 to 8 hexadecimal pairs of digits or ‘X’ symbols, which means any hexadecimal digit. ‘X’ symbols can be placed just in the end of mask. If higher MAC address bytes are absent it means zero in these bytes as for the mask as for the MAC address.

Example of ignore list json for WiFi white and black lists:

{

"wifi\_white\_list": [

"00:00:d0:1d:xx:xx:xx"

],

"wifi\_ignore\_list": [

"00:00:D0:1D:24:95:BC",

"00:00:D0:1D:24:95:AC"

]

}

BLE ignore list: <to be described>

# Datasets input list

Datasets list can be provided in json file specified with --input\_list key as list of dataset folders with attributes.

The structure of datasets input list is the following:

"datasets\_list":

[

["<dataset\_path\_1>", "<attrib1>"],

. . .

["<dataset\_path\_i>"],

. . .

["<dataset\_path\_n>”, "<attrib\_1>", "<attrib\_2>",…"<attrib\_n>"]

]

Where every dataset\_path specifies path to the dataset folder.

attrib\_1 - attrib\_n – list of attributes and keys which are applied to the dataset processing.

The following attributes are supported:

|  |  |  |
| --- | --- | --- |
| **Dataset attribute** | **Data Source** | **Comments** |
| “dataset\_type” | Coursa Venue + IVL | Specifies type of dataset |
| “dataset\_score” | Dataset survey quality score | Provides dataset survey quality assessment score “dataset\_score=*NN*”, NN is a score rating [0;100] |
| “route\_type” | Route settings | Specifies type of route related with the dataset. |

Attribute “data\_type” can modify settings defined with settings json-file. The modified setting is applied only to a dataset accompanied with the attribute.

|  |  |  |  |
| --- | --- | --- | --- |
| **dataset\_type** | **Modified setting** | | **Comment** |
| **in\_file\_mask** | **default\_mag\_validators** |
| “venue\_data” | “ivl\*.dat” | “for\_ivl” | Specifies dataset as crowdsourcing data |
| “survey\_data” | “irl\*.dat” | “for\_coursa\_survey\_tool” | Specifies dataset as survey data collected with Coursa Survey |
| “maper\_data” | “tpn.dat” | “for\_mapper” | Specifies dataset as Mapper data |
| “retail\_data” | “irl\*.dat” | “for\_irl” | Specifies dataset as retail data |
| “robotic\_survey\_data” | “irl\*.dat” | “for\_robo\_survey” | Specifies dataset as robotic survey data |

Attribute “route\_type” applies special dataset processing and validation depending on specified route type. The following route types are supported:

|  |  |
| --- | --- |
| **route\_type** | **Description** |
| “normal” | Processing and validation settings as defined by settings json file and “dataset\_type” attribute |
| “escalator” | Applies escalator portal processing and validation for the route. |
| “elevator” | Applies elevator portal processing and validation for the route. |
| “conveyor-belt” | Applies conveyor-belt portal processing and validation for the route. |
| “stairs” | Applies conveyor-belt portal processing and validation for the route. |
| “single-cell” | Applies restricted validation set for single cell routes. |
| “straight-line” | Applies restricted validation set for straight-line routes. |

Default value of the route\_type attribute is “normal” if the one is not specified. There is no processing for the dataset if unknown the route\_type attribute specified.

Note: there is no “crowdsourcing” route type defined. Use “dataset\_type= venue\_data” attribute to request crowdsourcing processing for the dataset.

Example of dataset list:

{

"datasets\_list":

[

[

["//server/dataset\_path\_1"],

["//server/dataset\_path\_1", "dataset\_type=survey\_data", “dataset\_score=78”],

["//server/dataset\_path\_3", "dataset\_type=venue\_data", “route\_type=single-cell”]

]

]

}

There are the following related settings in venue.json file.

#### “folder\_in”

This setting normally defines the path where FP\_builder performs search for datasets to processing according to the mask specified by “in\_file\_mask”. The search is performed in all subfolders.

If input list is provided then dataset search is disabled, and datasets specified into the input list file are processed. ~~Path, defined by setting “folder\_in” of venue.json, is joined with each dataset\_path\_i. So, setting “folder\_in” defines absolute path of datasets group and dataset\_path\_i defines local path for datasets to processing.~~

#### *"in\_file\_mask"*

The “in\_file\_mask” setting, specified in settings json-file, defines default mask used in the dataset search.

The mask can be modified individually for each dataset by dataset attributes in dataset input list.

The default mask is applied to search data file for each dataset folder specified in input list if “dataset\_type” attribute is not defined for the dataset.

Default value of "in\_file\_mask" is “irl\*.dat”.

#### *"default\_mag\_validators" and "mag\_validators"*

"default\_mag\_validators" setting defines a default validation set for data validation.

Dataset attributes specified in dataset input list can modify “default\_mag\_validators” setting for the accompanied dataset.

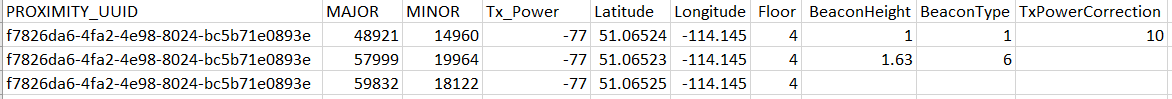
The default validation set is applied for each item specified in input list if “dataset\_type” attribute is not defined for the dataset.

"mag\_validators" settings defines additional validator list which is applied for every dataset whatever attribute specified.

# Proximity beacons list

A proximity beacons list for BLP-database build is provided to FPBL console in csv-format. The proximity beacons list file location is specified in settings-json file (venue.json) by a setting “in\_ble\_fprox\_file”.

Each line of the csv-file contains beacons UUID, major, minor, location, transmitted power level and other beacons parameters. An example of the file is given below.



The csv-columns description.

|  |  |  |
| --- | --- | --- |
| **Column number** | **Parameter name** | **Description** |
| 1 | BLE beacon UUID | UUID number in format “f7826da6-4fa2-4e98-8024-bc5b71e0893e” |
| 2 | Major | Major number |
| 3 | Minor | Minor number |
| 4 | Tx power | Transmitted power level at 1m distance in dBm. This parameter is required for iOS, because Tx power level of beacon is not available in iOS BLE interface. For Android this parameter is used for integrity check: the beacon data is rejected if Tx power specified in BLP-database does match to transmitted in BLE message Tx power. |
| 5 | Latitude | Beacon geodetic latitude in degrees. Positive values mean “N”, negative values mean “S”. |
| 6 | Longitude | Beacon geodetic longitude in degrees. Positive values mean “E”, negative values mean “W”. |
| 7 | Floor | Physical floor number where beacon installed. |
| 8 | Beacon installation height | Beacon installation height in meters from floor. Default value is 2m. |
| 9 | Beacon type | Beacon type. Default value is bbt\_unknown (0). See a type description below. |
| 10 | Tx power correction | Transmitted power level correction in dBm. Default value is 0 dBm. |

Columns 1-7 are mandatory and columns 8-10 are optional.

If an optional value is specified for a beacon, then all values in previous columns must be also provided for this beacon. For example, if a Tx power correction is provided for a bacon in 10th column, then values in all first 9 columns must be specified for this beacon – see example above.

Supported beacon types:

|  |  |  |
| --- | --- | --- |
| **Integer type number** | **Beacon type** | **Description** |
| 0 | bbt\_unknown | Type is unknown. The beacon is processed the same way as bbt\_proximity. |
| 1 | bbt\_proximity | General proximity beacon |
| 2 | bbt\_crowdsoursing | Crowdsourcing beacon |
| 3 | bbt\_crowdsoursing\_assistance | Crowdsourcing assistance beacon |
| 4 | bbt\_height\_assistance | Height assistance beacon |
| 5 | bbt\_bfp | BFP beacon |
| 6 | bbt\_restricted\_area | Beacon for restricted areas designation |
| 7 | bbt\_restricted\_assistance | Assistance beacon for restricted areas designation |

# Typical use-cases of fingerprint building

## Easy survey datasets processing

Usage: FP\_builder -- settings venue.json

Specific settings of venue.json:

{

…

“in\_file\_mask”: “irl\*.dat”,

"default\_mag\_validators": "for\_coursa\_survey\_tool",

"folder\_in": "//server/path\_to\_survey\_datasets",

…

}

Result: fingerprints and grids built with survey datasets from specified with “folder\_in” folder.

## Crowdsourcing datasets processing

#### < to be supported>

Usage: FP\_builder -- settings venue.json

Specific settings of venue.json:

{

…

“in\_file\_mask”: “ivl\*.dat”,

"default\_mag\_validators": "for\_ivl",

"folder\_in": "//server/in\_path\_to\_ivl\_datasets",

…

}

Result: fingerprints and grids built with crowdsourcing datasets from specified with “folder\_in” folder.

## Simultaneous processing of survey and crowdsourcing datasets



Usage: FP\_builder -- settings venue.json --input\_list dataset\_list.json

Specific settings of venue.json:

{

…

“in\_file\_mask”: “irl\*.dat”,

"default\_mag\_validators": "for\_coursa\_survey\_tool",

"folder\_in": "",

…

}

dataset\_list.json example:

{

"datasets\_list":

[

[

["dataset\_path\_1"], // survey data by default

["dataset\_path\_2", "survey\_data"],

["dataset\_path\_3", "venue\_data"],

…

]

]

}

Result: fingerprints and grids built with survey and crowdsourcing datasets provided in dataset\_list.json file.

## Iterative build



Early generated grids can be optionally provided to FP builder to support iterative build functionality. In this case only ivl datasets, which have not been processed before, must be provided for processing.

Usage: FP\_builder -- settings venue.json --input\_list dataset\_list.json

Specific settings of venue.json:

{

…

“in\_file\_mask”: “irl\*.dat”,

"default\_mag\_validators": "for\_coursa\_survey\_tool",

"folder\_in": "",

"input\_data" :

{

“mag\_grid\_file” : "//server/in\_path\_to\_grids/venue\_name.maggrid",

"wifi\_grid\_file" : "//server/in\_path\_to\_grids/venue\_name.wifigrid",

"ble\_grid\_file" : "//server/in\_path\_to\_grids/venue\_name.blegrid",

"portals\_ grid\_file" : "//server/in\_path\_to\_grids/venue\_name.portalsgrid",

}

…

}

dataset\_list.json example:

{

"datasets\_list":

[

[

["dataset\_path\_1"], // survey data by default

["dataset\_path\_2", "survey\_data"],

["dataset\_path\_3", "venue\_data"],

]

]

}

Result: fingerprints and grids based on specified grid files joined with data of survey and crowdsourcing datasets provided in dataset\_list.json file.

## Processing of survey and crowdsourcing datasets with grids separation



This case utilizes two step scheme and use separated grids for fingerprint generation. IVL grids built in first step are mixed with survey data in second step.

Early generated IVL grids can be optionally provided for first step build to support iterative build functionality. In this case only IVL datasets, which have not been processed before, must be provided for processing.

It is not necessarily to call ASCA or any other coverage tools after first step. Grids only are required for the second step.

Note, ivl\_venue.json and survey\_venue.json are different files and it is impossible to use one of them instead the other.

Usage:

Step 1: FP\_builder -- settings ivl\_venue.json

Step 2: FP\_builder -- settings survey\_venue.json

Specific settings of ivl-venue.json:

{

…

“in\_file\_mask”: “irv\*.dat”,

"default\_mag\_validators": "for\_ivl",

"folder\_in": "//server/in\_path\_to\_ivl\_datasets/",

"input\_data" :

{

“mag\_grid\_file” : "//server/in\_path\_to\_grids/venue\_name.maggrid",

"wifi\_grid\_file" : "//server/in\_path\_to\_grids/venue\_name.wifigrid",

"ble\_grid\_file" : "//server/in\_path\_to\_grids/venue\_name.blegrid",

"portals\_ grid\_file" : "//server/in\_path\_to\_grids/venue\_name.portalsgrid",

}

{

"max\_position\_uncertainty" : 2,

…

},

"wifi\_grid":

{

"max\_position\_uncertainty" : 4,

…

}

"ble\_grid":

{

"max\_position\_uncertainty" : 4,

…

}

…

}

Specific settings of survey\_venue.json:

{

…

“in\_file\_mask”: “irl\*.dat”,

"default\_mag\_validators": "for\_coursa\_survey\_tool",

"folder\_in": "//server/in\_path\_to\_ivl\_datasets/",

…

}