openlr

Release 1.0.0

Can Güney Aksakalli

Feb 21, 2020

CONTENTS:

OpenLR is an open standard for "procedures and formats for the encoding, transmission, and decoding of local data irrespective of the map" developed by TomTom.

This project implements XML and Binary format conversions of OpenLR standard according to the white paper and the reference implementation. (It does not include map-matching feature.)

CHAPTER

ONE

GETTING STARTED

You need a Python version >= 2.7. Install the package via pip:

```
pip install openlr
```

The package comes with a simple CLI to convert base64 encoded binary data into OpenLR XML format.

```
python -m openlr CwRbWyNG9RpsCQCb/jsbtAT/6/+jK1lE
```

Output:

```
<?xml version="1.0" ?>
<OpenLR xmlns="http://www.openlr.org/openlr">
  <LocationID></LocationID>
  <XMLLocationReference>
    <LineLocationReference>
      <LocationReferencePoint>
        <Coordinates>
          <Longitude>6.126819849014282</Longitude>
          <Latitude>49.60851788520813</Latitude>
        </Coordinates>
        <LineAttributes>
          <FRC>FRC3</FRC>
          <FOW>MULTIPLE_CARRIAGEWAY</FOW>
          <BEAR>141</BEAR>
        </LineAttributes>
        <PathAttributes>
          <LFRCNP>FRC3</LFRCNP>
          <DNP>557</DNP>
        </PathAttributes>
      </LocationReferencePoint>
      <LocationReferencePoint>
        <Coordinates>
          <Longitude>6.128369849014282</Longitude>
          <Latitude>49.60398788520813</Latitude>
        </Coordinates>
        <LineAttributes>
          <FRC>FRC3</FRC>
          <FOW>SINGLE_CARRIAGEWAY</FOW>
          <BEAR>231</BEAR>
        </LineAttributes>
        <PathAttributes>
          <LFRCNP>FRC5</LFRCNP>
          <DNP>264</DNP>
        </PathAttributes>
```

(continues on next page)

(continued from previous page)

```
      </LocationReferencePoint>
      <LastLocationReferencePoint>
        <Coordinates>
          <Longitude>6.128159849014282</Longitude>
          <Latitude>49.60305788520813</Latitude>
        </Coordinates>
        <LineAttributes>
          <FRC>FRC5</FRC>
          <FOW>SINGLE_CARRIAGEWAY</FOW>
          <BEAR>287</BEAR>
        </LineAttributes>
      </LastLocationReferencePoint>
      <Offsets>
        <PosOff>149</PosOff>
        <NegOff>0</NegOff>
      </Offsets>
    </LineLocationReference>
  </XMLLocationReference>
</OpenLR>
```

The same example programmatically:

```
import openlr

location = openlr.binary_decode("CwRbWyNG9RpsCQCb/jsbtAT/6/+jK1lE")
print(location.points[0].lon)   # 6.126819849014282
print(location.points[0].lat)   # 49.60851788520813
```

Defining a location object and converting it to XML and binary physical formats

```
import openlr

location = openlr.GeoCoordinateLocation(
    openlr.Coordinates(52.495218, 13.461668)
)
print(openlr.binary_encode(location))   # IyVUdwmSoA==
print(openlr.xml_encode_to_string(location))   # <?xml version="1.0" ?><OpenLR ...
```

Defining a LineLocationReference

```
from openlr import LineLocationReference, LocationReferencePoint, FRC, FOW

location = LineLocationReference(
    [
        LocationReferencePoint(
            6.1268198, 49.6085178, FRC.FRC3, FOW.MULTIPLE_CARRIAGEWAY, 141, FRC.FRC3,␣
↪557,
        ),
        LocationReferencePoint(
            6.1283698, 49.6039878, FRC.FRC3, FOW.SINGLE_CARRIAGEWAY, 231, FRC.FRC5,␣
↪264
        ),
        LocationReferencePoint(
            6.1281598, 49.6030578, FRC.FRC5, FOW.SINGLE_CARRIAGEWAY, 287, FRC.FRC7, 0
        ),
    ],
    0.26757812,
```

(continues on next page)

(continued from previous page)

```
    0,
)
```

CHAPTER

TWO

REFERENCE

2.1 Locations

2.1.1 Enums

class openlr.**FRC**

The functional road class is a road classification based on the importance of a road.

FRC0 = 0

Main road, highest importance

FRC1 = 1

First class road

FRC2 = 2

Second class road

FRC3 = 3

Third class road

FRC4 = 4

Fourth class road

FRC5 = 5

Fifth class road

FRC6 = 6

Sixth class road

FRC7 = 7

Other class road, lowest importance

class openlr.**FOW**

The form of way describes the physical road type.

UNDEFINED = 0

The physical road type is unknown

MOTORWAY = 1

A Motorway is defined as a road permitted for motorized vehicles only in combination with a prescribed minimum speed. It has two or more physically separated carriageways and no single level-crossings.

MULTIPLE_CARRIAGEWAY = 2

A multiple carriageway is defined as a road with physically separated carriageways regardless of the number of lanes. If a road is also a motorway, it should be coded as such and not as a multiple carriageway.

SINGLE_CARRIAGEWAY = 3
All roads without separate carriageways are considered as roads with a single carriageway.

ROUNDABOUT = 4
A Roundabout is a road which forms a ring on which traffic traveling in only one direction is allowed.

TRAFFICSQUARE = 5
A Traffic Square is an open area (partly) enclosed by roads which is used for non-traffic purposes and which is not a Roundabout.

SLIPROAD = 6
A Slip Road is a road especially designed to enter or leave a line.

OTHER = 7
The physical road type is known but does not fit into one of the other categories.

class openlr.**SideOfRoad**
The side of the road is valid only for point locations and indicates on which side of the referenced road the point location is located.

ON_ROAD_OR_UNKNOWN = 0
Point is directly on (or above) the road, or determination of right/left side is not applicable (default)

RIGHT = 1
Point is on right side of the road

LEFT = 2
Point is on the left side of the road

BOTH = 3
Point is on both sides of the road

class openlr.**Orientation**
The orientation is only valid for point locations and indicates for which direction the point information is relevant.

NO_ORIENTATION_OR_UNKNOWN = 0
Point has no sense of orientation, or determination of orientation is not applicable (default)

WITH_LINE_DIRECTION = 1
Point has orientation from first LRP towards second LRP

AGAINST_LINE_DIRECTION = 2
Point has orientation from second LRP towards first LRP

BOTH = 3
Point has orientation in both directions

2.1.2 Location Properties

openlr.**Coordinates**(*lon*, *lat*)
A coordinate pair *longitude* (lon) and *latitude* (lat) represented in WGS84 coordinates.

class openlr.**LineAttributes**(*frc*, *fow*, *bear*)
Line attributes consist of functional road class, form of way, and bearing.

bear multiplied by 11.25 is the bearing angle in degrees.

class openlr.**PathAttributes**(*lfrcnp*, *dnp*)
Path attributes consist of lowest FRC to next point, and distance in meters to next point.

class openlr.**LocationReferencePoint**(*lon*, *lat*, *frc*, *fow*, *bear*, *lfrcnp*, *dnp*)
A location reference point consists of coordinate, line attribute and path attribute data.

Refer to *Coordinates*, *LineAttributes*, and *PathAttributes* for more information.

2.1.3 Reference Locations

class openlr.**LineLocationReference**(*points*, *poffs*, *noffs*)
A line location reference describes a path within a map and consists of location reference point(s), a last location reference point and offset data.

class openlr.**GeoCoordinateLocationReference**(*point*)
GeoCoordinate is a point location which consists of exactly one Coordinate pair.

class openlr.**PointAlongLineLocationReference**(*points*, *poffs*, *orientation*, *sideOfRoad*)
Point along line is a point location which is defined by a line and an offset value. The line will be referenced by two location reference points and the concrete position on that line is referenced using the positive offset. Additionally information about the side of the road where the point is located and the orientation with respect to the direction of the line can be added.

class openlr.**PoiWithAccessPointLocationReference**(*points*, *poffs*, *lon*, *lat*, *orientation*, *sideOfRoad*)
Point along line with access is a point location which is defined by a line, an offset value and a coordinate. The line will be referenced by two location reference points and the concrete position of the access point on that line is referenced using the positive offset. The point of interest is identified by the coordinate pair. Additionally information about the side of the road where the point is located and the orientation with respect to the direction of the line can be added.

class openlr.**CircleLocationReference**(*point*, *radius*)
A CircleLocationReference represents a circle area location.

The radius is given in meters.

class openlr.**RectangleLocationReference**(*lowerLeft*, *upperRight*)
A RectangleLocationReference represents a rectangular area location.

class openlr.**GridLocationReference**(*lowerLeft*, *upperRight*, *n_cols*, *n_rows*)
A GridLocationReference represents a rectangular area location with a grid of n_cols columns and n_rows rows.

class openlr.**PolygonLocationReference**(*corners*)
A PolygonLocationReference represents a polygonal area location.

openlr.**ClosedLineLocationReference**
A ClosedLineLocationReference is defined by an ordered sequence of location reference points and a terminating last location reference point.

alias of openlr.locations.ClosedLineLocation

2.2 XML Format

XML OpenLR physical format conversion methods based on the white paper.

openlr.**xml_decode_document**(*doc*)
Decodes OpenLR xml minidom Document into a location

openlr.**xml_decode_file**(*filename_or_file*)
Decodes an OpenLR XML from a filename or file object

openlr.**xml_decode_string**(*string*)
Decodes an OpenLR XML from string

openlr.**xml_encode_to_document**(*location*)
Encodes location object to OpenLR xml minidom document

openlr.**xml_encode_to_string**(*location*, *is_pretty=True*)
Encodes location object into an OpenLR XML string

2.3 Binary Format

Binary OpenLR physical format conversion methods based on the white paper.

openlr.**binary_decode**(*data*, *is_base64=True*)
Decodes binary data into Location

Parameters

- **data** (*str, bytearray, bytes*) – A bytes-like object that contains the binary data
- **is_base64** (*bool*) – Boolean flag for base64 encoded string data

Returns **location** – Location object

Return type NamedTuple

openlr.**binary_encode**(*location*, *is_base64=True*)
Encodes Location object into binary data

Parameters

- **location** (*NamedTuple*) – Location object
- **is_base64** (*bool*) – Boolean flag for base64 encoded string data

Returns **data** – A bytes-like object that contains the binary data

Return type str, bytearray, bytes

2.4 Binary Internal APIs

Internal API for binary format conversion. It provides an extended io.BytesIO stream class to read/write OpenLR binary data.

openlr.openlr_bytes_io.**deg_to_int**(*deg*, *resolution=24*)
converts degree coordinate into integer

$$\text{int} = 0.5 \times \text{sgn}(\text{deg}) + \frac{\text{deg} \times 2^{\text{resolution}}}{360°}$$

Parameters

- **deg** (*float*) – Coordinate Degree
- **resolution** (*int*) – Resolution bits of this integer, default: 3 bytes = 24

Returns **val** – Coordinate value in integer

Return type int

openlr.openlr_bytes_io.**int_to_deg**(*val*, *resolution=24*)
converts degree coordinate into integer

$$sgn\deg = \frac{(\text{int} - 0.5 \times (\text{int})) \times 360°}{2^{\text{resolution}}}$$

Parameters

- **val** (*int*) – Coordinate value in integer
- **resolution** (*int*) – Resolution bits of the returned integer, default: 3 bytes = 24

Returns **deg** – Coordinate value in degree

Return type float

openlr.openlr_bytes_io.**bytes_to_int**(*b*, *signed=True*)
converts big endian bytes to signed/unsigned int

openlr.openlr_bytes_io.**int_to_bytes**(*val*, *size=3*, *signed=True*)
positive/negative int values to big endian

class openlr.openlr_bytes_io.**OpenLRBytesIO**
In-memory binary stream for reading/writing OpenLR data

read(*size=-1*)
Python 2 compatibility for str -> bytes

read_status()
Reads status from the first byte: version & location type

Returns

- **version** (*int*) – OpenLR version: 3 bit
- **location_type** (*int*) – Combination of Point flag, Area flag & Attributes flag: 4 bit

read_offset()
Reads offset rate from the stream, 1 byte

Returns **offset** – offset rate in [0,1] range

Return type float

read_dnp()
Reads distance to next point from the buffer, 1 byte

Returns **dnp** – Distance to next point

Return type int

read_point_attributes()
Reads point attributes from the buffer, 2 bytes

Returns

- **fow** (*int*) – form of way (3 bits)
- **frc** (*int*) – functional road class (3 bits)
- **bear** (*int*) – Bearing (5 bits)
- **lfrcnp** (*int*) – Lowest FRC to next point or offset flags (3 bits)
- **reserved** (*int*) – mostly reserved for future use or Orientation/SideOfRoad (2 bits)

read_coords()
Reads absolute coordinates from the buffer, 6 bytes

Returns

- **lon** (*float*) – Longitude
- **lat** (*float*) – Latitude

read_coords_relative(*prev_lon*, *prev_lat*)
Reads coordinates from the buffer relative to the previous ones, 4 bytes

Parameters

- **prev_lon** (*float*) – Previous Longitude
- **prev_lat** (*float*) – Previous Latitude

Returns

- **lon** (*float*) – Longitude
- **lat** (*float*) – Latitude

read_radius()
Reads an integer from the remaining bytes till the end

Returns **radius** – integer value

Return type int

read_cols_rows()
Reads the number of columns and rows, 4 bytes

Returns

- **cols** (*int*)
- **rows** (*int*)

write_status(*version*, *location_type*)
Writes status byte to the stream

Parameters

- **version** (*int*) – OpenLR version (3 bit)
- **location_type** (*int*) – Combination of Point, Area and Attribute flags (4 bit)

write_offset(*offset*)
Writes offset rate to the stream, 1 byte

Parameters **offset** (*float*) – offset rate in [0,1] range

write_dnp(*dnp*)
Writes distance to next point to the stream, 1 byte

Parameters **dnp** (*int*) – distance to next point

write_point_attributes(*fow*, *frc*, *bear*, *lfrcnp*, *reserved*)
Writes 2 bytes point attributes to the stream

Parameters

- **fow** – form of way (3 bits)
- **frc** (*int*) – functional road class (3 bits)
- **bear** (*int*) – Bearing (5 bits)
- **lfrcnp** (*int*) – Lowest FRC to next point or offset flags (3 bits)

- **reserved** (*int*) – mostly reserved for future use or Orientation/SideOfRoad (2 bits)

write_coords(*lon*, *lat*)
Writes the absolute coordinates to the stream, 6 bytes

Parameters

- **lon** (*float*) – Longitude to be written
- **lat** (*float*) – Latitude to be written

write_coords_relative(*lon*, *lat*, *prev_lon*, *prev_lat*)
Writes the relative coordinates to the stream, 4 bytes

Parameters

- **lon** (*float*) – Longitude to be written
- **lat** (*float*) – Latitude to be written
- **prev_lon** (*float*) – Previous Longitude
- **prev_lat** (*float*) – Previous Latitude

write_radius(*radius*)
Writes the radius, 1-4 bytes depending on the value size

Parameters **radius** (*int*) – Radius value in [0, 4294967295] range

write_cols_rows(*cols*, *rows*)
Writes the number of columns and rows, 4 bytes

Parameters

- **cols** (*int*) –
- **rows** (*int*) –

2.5 Helper Functions

openlr.**get_dict**(*location*)
Helper to convert location object to dict

openlr.**get_lonlat_list**(*location*)
Helper to return a list of lonlat tuples of coordinates in a location

CHAPTER

THREE

DEVELOPMENT

This project uses tox for running tests and other things (style check, coverage reports, sphinx docs).

It is recommended to install conda along with tox-conda for using tox. Afterwards, `tox` command will run all the steps for you.

This project strictly uses black as an opinionated code style. Any written line should comply with that (don't forget to run `black .`).

Always add tests for bug fixes and feature developments.

3.1 Binary Location Types

Location type is determined based on the flags in the first byte and the data size.

Location type	Data size	B6	B5	B4	B3
Line	16 + (n-2) * 7 + [0/1/2] bytes	0	0	0	1
Geo-coordinate	7 bytes	0	1	0	0
Point along line	16/17 bytes	0	1	0	1
POI with access point	20/21 bytes	0	1	0	1
Circle	7 + [1/2/3/4] bytes	0	0	0	0
Rectangle	11/13 bytes	1	0	0	0
Grid	15/17 bytes	1	0	0	0
Polygon	15 + (n-3) * 4 bytes	0	0	1	0
ClosedLine	19 + (n-3) * 7 bytes	1	0	1	1

Flags in the first byte:

- Bit 6 (ArF1) - Area Flag 1
- Bit 5 (no point)
- Bit 4 (ArF0) - Area Flag 0
- Bit 3 (has attributes)

CHAPTER

FOUR

OTHER RELATED PROJECTS

- PyLR OpenLR implementation in Python2
- openlr-js OpenLR implementation in JavaScript
- OpenLR C# OpenLR library for .NET.
- openlr-decoder Web API for the reference implementation

PYTHON MODULE INDEX

O

INDEX

S

T

U

W

X