OJP Middleware Documentation

Modules

all modules are single npm packages some are dependent on each other

- api-ojp
- api-otp
- ep-manager
- tests

learn more (modules.md)

Services

some of modules implement a Docker service running in individual container and associated with a specific port to an Api REST interface.

docker-compose.yml this sets up the infrastructure to make these services interact

- api-ojp
- api-otp
- ep-manager

learn more (services.md)

Config

each module of project contains a single config.yml file it define contains service configurations

learn more (config.md)

Structure

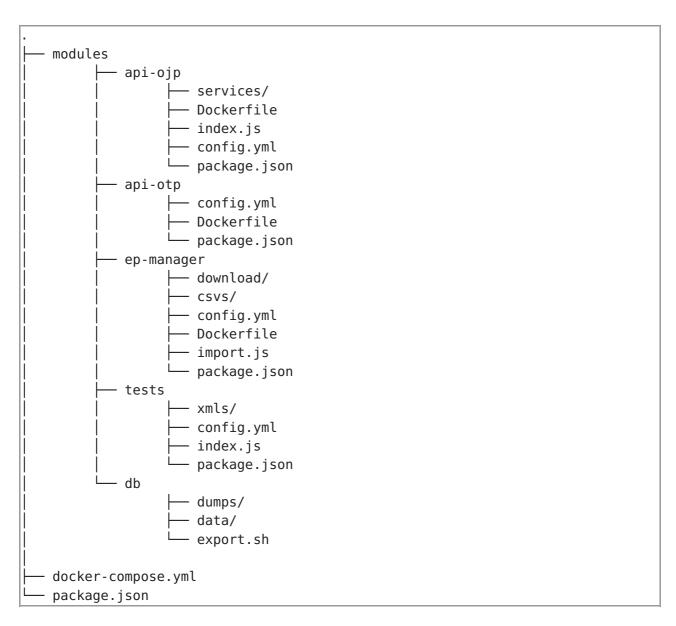
Common structure for modules and services is:

- config.yml
- index.js
- package.json

and for services is:

- Dockerfile
- env.example(renamed to .env in dev environment)

The basic structure of code:



References

OJP general api docs:

https://github.com/VDVde/OJP/tree/markdowns

api requests/response docs:

https://vdvde.github.io/OJP/generated/OJP.html

Modules

- api-ojp
- api-otp
- ep-manager
- tests

api-ojp (api-ojp.md)

OJP entrypoint, implements OJP responses

api-otp (api-otp.md)

maintain connection to OTP instance

ep-manager (ep-manager.md)

OJP exchangepoint mananger exchange point collect stops

tests

simple web front-end to test OJP requests

Services

base structure of any Docker service:

- a file <u>config.yml</u> (<u>config.md</u>) contains common service configurations(example PORT) or specific setting for service
- a .env contains specific environment variables, this file is based on env.example for debugging mode of single service

Ports

default ports in production environment by services

service	production	development
api-ojp	9091	8081

api-otp 9092 8082 ep-manager 9093 8083 db 27017/9095 -

27017/3033

tests 9096 8086

Config

each module of project contains a single **config.yml** file it define service configurations(example PORT).

dev and prod implement two different environments, development and production, prod also refers to docker-compose.yml in the project root.

Outside of *dev* and *prod* are common configurations to the two environments.

Below of common structure of a config.yml file:

```
environments:
  default: prod
dev:
  server:
   port: 8083
  db:
    uri: mongodb://${MONGO HOST}:${MONGO PORT}/
    name: oip
    collection: exchange points
prod:
  server:
    port: 9093
  db:
    uri: mongodb://db/
    name: ojp
    collection: ${dev.db.collection}
import:
 version: 0.16
  csvFile: 5T.csv
```

these config files may contain environment variables that are valued at runtime. In this example MONGO HOST, MONGO PORT

the same values defined within the yml file can be used to make substitutions at runtime In this example \${dev.db.collection}

defaults project ports configurations listed here: services.md#ports (services.md#ports)

API OJP

OJP entrypoint

implements this OJP entrypoints:

- OJPLocationInformation
- OJPTrip
- OJPStopEvent
- OJPTripInfo
- OJPExchangePoints
- OJPMultiPointTrip

default environments variables

OTP MAX PARALLEL REQUESTS maximum number of parallel request to OpenTripPlanner

default restrictions by config.yml

• include_intermediate_stops: value of ojp:IncludeIntermediateStops (default: false)

- include accessibility: value of ojp:IncludeAccessibility (default: false)
- ojptag in response: include namespace ':ojp' in all tags in results (default: true)
- include_precision: include ojp:Precision tag in reponses (default: false)
- location_digits: precision for all coordinates in reponses (default:5)
- transfer_limit: value of ojp:TransferLimit in reponses (default: 2)
- limit: limits of results (default: 10000)
- skip: results starting from (default: 0)

api-otp

maintain connection to OpenTripPlanner instance

environment

OTP HOST hostname instance of OpenTripPlanner

OTP PATH basepath of OpenTripPlanner graphql api example: /otp/routers/default/index/graphql

OTP PORT port instance of OpenTripPlanner

QUERY_DEBUG if set show graphql queries in output

default query parameters by config.yml

• caching: false

• default_limit: 10000

default_skip: 0

exchangepoint manager

mongodb models to store ojp exchangepoints

TODO download.sh script to download remote exchangepoint

TODO maybe include NETEXT IFOPT

https://github.com/NeTEx-CEN/NeTEx/blob/master/xsd/ifopt.xsd

import exchange points CSV data manually

locale data:

CSV VERSION=10 node import.js

or from remote resource:

CSV_URL=https://remote-resource.com/exchange-points.csv node import.js

environment

CSV VERSION is directory inside csvs default is version param inside config.yml

CSV_AUTOIMPORT is True enable auto import of exchange points csv data into database at startup

CSV_URL remote URI of exchange point in csv format

usage in docker

docker-compose up ep-manager

browse: http://localhost:8083/

browse: http://localhost:8083/geojson

development mode

docker-compose up db

npm run dev