Basic Input Manual

V2G

```
Electrification rate (1.0= 100% of the agents have an EV or PHEV)
double electrification= 1.0;

Percentage of EVs from electric vehicle fleet in the system (1.0=100% EVs)
double ev=0.0;

Output folder
String outputPath="D:/Output/...";

Config path
final String configPath="test/scenarios/berlin/config.xml";

Battery size of EV and PHEV vehicle in kWh
double kWHEV =16;
double kWHPHEV =16;

// gas price, i.e. 1.70 CHF/liter
double gasHigh = 1.70;

Define the hubs and their input. for each hub create a HubInfo Object and
add it to the ArrayList<HubInfoDeterministic> myHubInfo. For multiple hubs,
add multiple entries to myHubInfo

Below is an example for one hub with specified parameters
   -  Maximum charging price at hub [CHF/kWh]
   -  Minimum  charging price at hub [CHF/kWh]
   -  Input file with 15 min bin data for free load curve [W]

double priceMaxPerkWh=0.11;
double priceMinPerkWh=0.07;
String freeLoadTxt= "test/input/playground/wrashid/sschieffer/load.txt";

ArrayList<HubInfoDeterministic> myHubInfo = new
ArrayList<HubInfoDeterministic>(0);
myHubInfo.add(new HubInfoDeterministic(1, freeLoadTxt, priceMaxPerkWh,
priceMinPerkWh));


Define the mapping class that shall be used to map the linkdIds to the hubs
in the DecentralizedSmartCharger. The object needs to extend the abstract
class MappingClass, currently StellasHubMapping is implemented which allows
you to specify the number of rectangular hubs you want in x and y direction
of the network

int numberOfHubsInX=1;
int numberOfHubsInY=1;
StellasHubMapping myMappingClass= new
StellasHubMapping(numberOfHubsInX,numberOfHubsInY);

Define the speed of the standard electricity outlet connection [W]

double standardConnectionWatt=3500;

LP Optimization parameters
- battery buffer for charging (e.g. 0.2=20%, agent will have charged 20%
more than what he needs before starting the next trip )
```

```
final double bufferBatteryCharge=0.0;

Charging Distribution
- standard charging length [s] = time resolution

final double standardChargingLength=15*60;

Create simulation object
DecentralizedChargingSimulation mySimulation= new
DecentralizedChargingSimulation(
      configPath,
      outputPath,
      electrification,
      ev,
      bufferBatteryCharge,
      standardChargingLength,
      myMappingClass,
      myHubInfo,
      false, // indicate if you want graph output for every agent to
visualize the SOC over the day
      kWHEV,kWHPHEV, gasHigh,
      standardConnectionWatt
      );
```

(Additional to Input for Decentralized Smart Charger)

```
Information about all stochastic loads at the hubs as an
ArrayList<HubInfoStochastic> Object
ArrayList<HubInfoStochastic> myStochasticHubInfo = new
      ArrayList<HubInfoStochastic>(0);


GENERAL STOCHASTIC LOAD (REQUIRED)
To add a general stochastic hub load at hub 1, specify the 96 bin data of
the stochastic load as an input .txt. file and add it to the new
HubInfoStochastic Object for hub 1
String stochasticGeneral= "stochastic.txt";
HubInfoStochastic hubInfo1= new HubInfoStochastic(1, stochasticGeneral);


HUBSOURCES (OPTIONAL)
To add a hub load, create a general source object and add it to the
ArrayList
ArrayList<GeneralSource> generalHubSource= new ArrayList<GeneralSource>(0);

To define the general source with discrete load intervals, create the new
General Source with an ArrayList of LoadDistribution Intervals
ArrayList<LoadDistributionInterval> generalHubLoad= new
      ArrayList<LoadDistributionInterval>(0);
generalHubLoad.add(new LoadDistributionInterval(3500, 7000, 5000));
generalHubSource.add(new GeneralSource(
                        generalHubLoad, //ArrayList of Loads at hub source
                        new IdImpl(1),  //LinkId
                        "discrete load", // name
                        0.005) ); // compensation for feed in

ArrayList<GeneralSource> generalHubSource= new ArrayList<GeneralSource>(0);
```

```
To define the general source with a continuous load curve, create the new
General Source with a 96 bin .txt file
String hubSourceLoad= "stochasticHubLoad.txt";
generalHubSource.add(new GeneralSource(
                        hubSourceLoad, // input .txt file
                        new IdImpl(2),
                        "continuous load",
                        0.005));

Add all hub loads to hubInfo
hubInfo1.setStochasticGeneralSources(generalHubSource);


STOCHASTIC VEHICLE LOAD (OPTIONAL)
For every vehicle specify the input load intervals for every vehicle as an
ArrayList of LoadDistribution intervals and save them to a HashMap with the
Agent Id as an identifier.
HashMap <Id, ArrayList<LoadDistributionInterval>> vehicleLoadHashMap = new
HashMap<Id, ArrayList<LoadDistributionInterval>>();
ArrayList<LoadDistributionInterval> vehicleLoad= new
ArrayList<LoadDistributionInterval>(0);
vehicleLoad.add(new LoadDistributionInterval(3500, 7000, 3500));
vehicleLoadHashMap.put(new IdImpl(1), vehicleLoad);

hubInfo1.setStochasticVehicleSourcesIntervals(vehicleLoadHashMap);

Add all stochastic loads corresponding to one hub:
myStochasticHubInfo.add(hubInfo1);


Create simulation object
DecentralizedChargingSimulation mySimulation= new
DecentralizedChargingSimulation(
      configPath,
      outputPath,
      electrification, ev,
      bufferBatteryCharge,
      standardChargingLength,
      myMappingClass,
      myHubInfo,
      false,kWHEV,kWHPHEV, gasHigh,
      standardConnectionWatt
      );



Specify percent of agent contracts providing only regulation down or
regulation up and down

final double xPercentDownUp=1.0;
final double xPercentDown=1.0- xPercentDownUp;

V2G compensation for regulation up, down, and feed in
double compensationPerKWHRegulationUp=0.1;
double compensationPerKWHRegulationDown=0.005;
double compensationPERKWHFeedInVehicle=0.005;

V2G set up, including events listener
mySimulation.setUpV2G(
      xPercentDown,
      xPercentDownUp,
```

```
        new StochasticLoadCollector(mySimulation, myStochasticHubInfo ),
        compensationPerKWHRegulationUp,
        compensationPerKWHRegulationDown,
        compensationPERKWHFeedInVehicle);

Run
mySimulation.controler.run();
```