Linkages between the scenario Explorer and EMPS-W

1. Reading from the explorer

1.1 Electricity demand profile

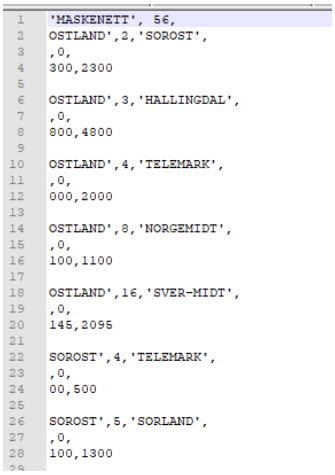
Here we read the electricity profile from the scenario explorer and put it into EMPS-W. This is done in one python script elDemand_fromSE2h5.py, which does:

1) download all data from GUSTO and the relevant scenario from the explorer.
2) extract only the electricity profile (the other data from GUSTO we don't need)
3) extract only the relevant regions (Norway and Spain – with subregions), for the test case provided we only use one region ES112
4) convert the "series" format of the values to a numpy array
5) write to .h5 file:
    a. open the *Here_goes_el_emand.h5* file
    b. if there are entries for electricity demand in the .h5 file, they are deleted
    c. create a new "group" (h5 subitem) for electricity demand
    d. create a dataset in the electricity demand group and fill it with the numpy array
    e. add unit

1.2. Transmission data

Here we read the electricity transmission data from the scenario explorer and put it into EMPS-W. This is done in one python script manipulateMaskenett.py, which does:

1) read in existing MASKENETT.data file (file contains transmission capacities and losses)
2) the file has "4 blocks" (each with a different format, i.e. amount of rows, amount of items per row, delimiter,…) of data, initiate a dataframe for each block
3) download data on transmission lines and losses from the scenario explorer and read that data from excel file
4) extract the data for relevant regions (Spain and Norway)
5) save "from" region and "to" region (transmission data has two regions associated with each data point)
6) convert the data into the format of the "4 blocks" – i.e. adjust order, delimiter, rownumber, etc.
7) combine the blocks into one long list
8) print the new list to the MASKENETT.data file



```
'MASKENETT', 56,
OSTLAND',2,'SOROST',
,0,
300,2300

OSTLAND',3,'HALLINGDAL',
,0,
800,4800

OSTLAND',4,'TELEMARK',
,0,
000,2000

OSTLAND',8,'NORGEMIDT',
,0,
100,1100

OSTLAND',16,'SVER-MIDT',
,0,
145,2095

SOROST',4,'TELEMARK',
,0,
00,500

SOROST',5,'SORLAND',
,0,
100,1300
```

2. Writing to the explorer

2.1. Electricity prices

Here we write the electricity prices produced by EMPS-W to the IAMC format and upload to the database. The script producing data in IAMC format is the python script read_h5_res4hero.py, which does:

1) Establish the IAMC format for the output, i.e. header row, fixed column content (version name, model name, unit, etc.)
2) Read in region names in nomenclature naming
3) Create timestamps for timesteps (in correct format)
4) Read price data from .h5 file (EMPS-W output file)
5) Extract the relevant regions
6) Add costs for grid use and taxes to prices
7) Convert from öre/kWh to US$2010/GJ
8) Create 3 final dataframes, mean/min/max prices
9) Prices are only available for every third hour, so we duplicate them for the hours missing
10) Print to excel

The excel files are then uploaded manually to the scenario explorer.

2.2 Pumped storage energy

Here we write the energy in pumped storage per region produced by EMPS-W to the IAMC format and upload them to the database. The script producing data in IAMC format is the python script read_h5_pump4tepes.py, which does:

1) Establish the IAMC format for the output, i.e. header row, fixed column content (version name, model name, unit, etc.)
2) Read in region names in nomenclature naming
3) Create timestamps for timesteps (in correct format)
4) Read pumped energy stored data from .h5 file (EMPS-W output file)
5) Extract the relevant regions
6) Create 3 final dataframes, mean/min/max energy level
7) Energy level is for the time being only available per week, but will be available for each hour soon. Until then I duplicate them for the hours missing – so as a tmp solution we have the same value for each hour of the week.
8) Print to excel

The excel files are then uploaded manually to the scenario explorer.