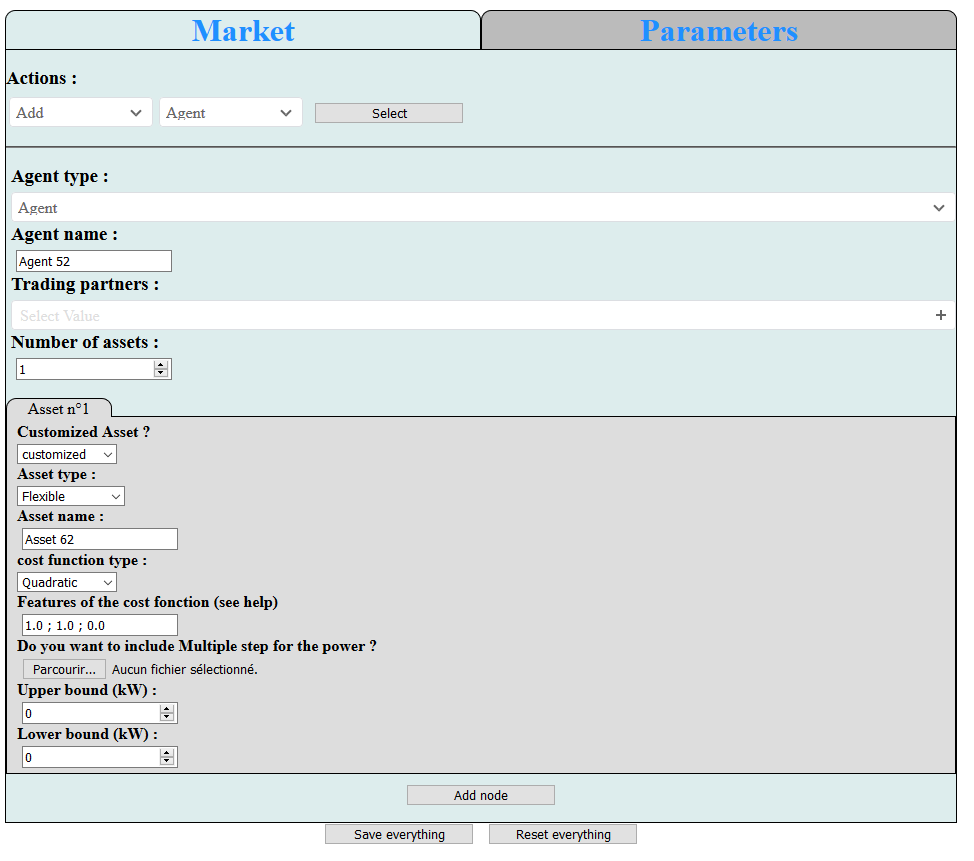
User guide

# General:

This application aimed at generating simulation case for smart-grid. We used the same notation and the variable than in our simulation. But this application just generates a JSON object, so feel free to use the different feature of this application for simulate other things, you can treat the JSON object as you want. This application is for now, not compatible with Internet Explorer (due to iterator on object).

# Tutorial:



1

2

3

4

There are two tabs for the application (1), the first one, called Market, is used to generate the study case (agent, community, link and asset), the second one, called Parameters is used to generate the parameters of the simulation to resolve the Market. The two tabs generate their own javascript object, and you can download in one JSON file, respectively reset, the two objects by using the button “Save everything”, respectively the button “Reset everything” on the footer of the page (4). The dowload is made by javsascript, so you can’t choose the name of th file (but you can rename it manualy), to chose where it will be download, it depend on web browser and its option.

In (3), there are all forms needed to complete the action chosen in (2). The form printed may depend on the action and the chooses in the previous forms. Every change (for the action or in the form) is considered only when the correspondant button is clicked. Whithout that every change is temporary, and only visual. Below, there is the list of the option available and their description.

## Add

### Agent:

In this choice, you can add an agent (which have asset) or a community manager (which have a community objective and can have community members, these members are NOT chosen in this part but by adding link). We can add trading partners and administrator to the node. A name is given by default (according to the type and the id of the node).

### Community:

In this choice, you can add a community (one community manager with trading partners and a community manger) and several agents which have no asset (you need to add asset with the choice Add asset). Names for the community and agent are set by default, but you can change them.

Link:

In this choice, you can add a link of community membership (between a community manager and a node) or a link of partnership between two nodes. It is possible to choose several nodes for the destination to create several links for a node. Preferences of the link can be set, numbers must be separated by a “;” and the decimals are after a “.” (for example 1.1;2.3). It is also possible to realise a full peer to peer (create link between all agents and delete all community managers) or a producer to consumer (create link between agent according to their assets and delete all community managers). It is worth noticing that it is not possible to create two links between the same couple of nodes (even if the links don’t have the same type).

### Asset:

In this choice, you can add one or several assets to a selected agent. The asset can be customized or an asset by default (\*not really implemented yet\*). A name is given by default (according to the id of the asset) but it can be changed. The type of the asset can be chosen between flexible (the asset can provide or consume every power between its power max and minimal), uncontrollable (the asset just consumes or provide one value of power) and uncertain (the power is defined by a stochastic law). For now, only one type of cost function is available: quadratic. To define the feature of cost or the stochastic function the number must be separated by a “;” and the decimals are after a “.” (for example 1.1;2.3), see the Class part to see what the coefficients are. It is also possible to add multiple step for the simulation. To do that you must add a file csv where the second line and the third line (if flexible) are numbers which represent the power. Even if a file is selected, the form for one power remain, because it will correspond to a default power (if you don’t want to have several steps for one simulation). This option is not available for uncertain asset.

### From example:

In this choice, you can add a set of nodes, links and asset from json files included in the application. As we include with JavaScript, it is possible to add your own examples. To do that you just have to put your json file in the same folder that the html file *index.htlm*, and add the name of your example (the name of the file) in the list *choices.examples* in the file *data\_gestion.js.* See next part to know how to have a compatible file.

### From file

In this choice, you can add a set of nodes, links and asset from json files in your computer. The object must be the same than in the application (see Variable part). It is possible to convert a csv file into a json file which is compatible thanks to the files in the transformation folder. You can change the function processing according to the field of the csv file (for example, the actual function converts a list of assets into the object data).

## Modify

### Link:

In this choice you can modify the preferences of a link by selecting the two nodes joined by the link. To change the type of the link (if it is possible), you must delete the link and create the link with the good type.

### Agent/Community:

In this choice you can modify a node. There are the same forms than for the creation of the node, but you can also modify the community members of a community manager and you can’t change the number of assets for the agent.

### Asset:

In this choice you can modify the assets of an agent. There are the same forms than for the creation of the asset. Beware even if it is not visible, if at a moment you had a file to the asset for a multiple step simulation, these values remain.

## Delete

### Link:

In this choice, you can delete a link between two nodes, it will also update the node concerned.

### Agent/Community

In this choice, you can choose between delete an agent or a community manager, but also delete only some assets of an agent or a community complete. The deletion of the community is not recursive, if another community is included on the deleted community, it will just delete the community manager (and the link) but not the agent of the community included.

## Save

In this choice, you just download the json file corresponding to the actual variable data (with the node, the link and the assets).

## Open

### New:

In this choice, you will reset the variable data, so it will delete all agents, community manager, links and assets. This action is irreversible so save before if you don’t want to lose the market case.

### From example:

In this choice, you can open a set of nodes, links and asset from json files included in the application. It will delete everything on data, so don’t forget to save before if you don’t want to lose the market case. As we include with JavaScript, it is possible to add your own examples. To do that you just have to put your json file in the same folder that the html file *index.htlm*, and add the name of your example (the name of the file) in the list *choices.examples* in the file *data\_gestion.js.* See next part to know how to have a compatible file.

### From file:

In this choice, you can open a set of nodes, links and asset from json files in your computer. It will delete everything on data, so don’t forget to save before if you don’t want to lose the market case. The object must be the same than in the application (see Variable part). It is possible to convert a csv file into a json file which is compatible thanks to the files in the transformation folder. You can change the function processing according to the field of the csv file (for example, the actual function converts a list of assets into the object data).

# Options added on the application online:

## Session:

A tab has bee added to allow users to connect themselves or subscribe with a pseudo (which must be unique) and a password. Be registered and connected, give to the user access to a personal folder on the server, where the user can save (or add, open) personal market case. As the files are on a server, that is no longer possible to add example. When the user saves a market case on the server (with the option SAVE -> Save As), he must choose a name for the file (which will also define how refer to this case to select it). If the name is already use by the user, it will delete the former file, be cautious.

# Modules:

Slim-select: for a visual change of the different select, allows also to search an option when we must choose, and finally change the way of selecting options in multiple select. The stylesheet css must be imported to use this module.

Cystoscape: module for the gestion of the graph (position of node, position of node, style of the graph). The js file cytoscape-cxtmenu allows us to add a menu for the node, the edge and the core of the graph.

Nouislider: module for visual change on the slider and allows to have two pointers. As for slim-select, the stylesheet css must be include.

Papaparse: module which convert csv file into a JS object and vice versa.

# Class:

## Node

constructor (id, type, name, partners, administrator, asset, assetActive, objective ='', communityMember = [], typeAgent='')

* Id: identifier of the node, unique for all nodes
* Type: type of the node, can be Agent (which have Asset) or Manager (can be the administrator of a community)
* Name: name of the node, can be chosen by the user, not necessarily unique
* Partners: list of id of node which have a partnership link with the node
* Administrator: list of id of Manager which are the administrator of the communities where the node is.
* Asset: list of the id of Asset that have this node (only used for Agent)
* assetActive: list of the id of active Asset that have this node (only used for Agent) *\* not implemented yet\**
* objective: the objective of the community which is administrate by this node (only for Manager)
* communityMember: list of id of nodes which are in the community administrate by this node (only for Manager)
* typeAgent: type of the agent according to its assets (can be producer, consumer or prosumer)

## Link

Constructor (id, type, source, destination, name, weightSrc=0, weightDest =0)

* Id: identifier of the link, unique for all links
* Type: type of the link, can be Partnership (between two nodes) or Community membership (between a node and a Manager)
* source: id of the node which is the source (only for convention for the sign for power exchange)
* destination: id of the node which is the destination (only for convention for the sign for power exchange)
* Name: name of the link, can be chosen by the user, not necessarily unique
* weightSrc: preference of the source ($/kW)
* weightDest: preference of the destination

## Asset

Constructor (id, name, type, functionType, functionCharac, Pmincap, Pmaxcap, Pmint, Pmaxt, uncertaintyType, uncertainCharac, GPS)

* Id: identifier of the asset, unique for all assets
* Name: name of the link, can be chosen by the user, not necessarily unique
* Type: type of the asset, can be Flexible (the power can be chosen between two values), Uncontrollable (the power can only take one value), Uncertain (can add stochastic effect)
* functionType: type of the cost function, for now, can only be Quadratic
* functionCharac: list of coefficients of the function cost, if quadratic the three coefficients are a; b; c for the function ax²+bx+c; the number must be separated by a “;” and the decimals are after a “.” (for example 1.1).
* Pmincap: lower limit of the power (kW)
* Pmaxcap: Upper limit of the power (kW) (equal to Pmincap if uncontrollable)
* Pmint: list of Lower limits of the power (kW) the user wants to run a multiple step simulation
* Pmaxt: list of Upper limits of the power (kW) the user wants to run a multiple step simulation (equal to Pmint if uncontrollable)
* uncertaintyType: type of the stochastic function if the asset is uncertain, can be Gaussian or Gamma
* uncertainCharac: list of coefficients who can be used to describe the function; for gaussian: expectation and standard deviation; for gamma shape parameter k and scale parameter θ.
* GPS: list of GPS coordinates of the Asset *\* not implemented yet \**

Function:

Most of the functions are made to generate the different forms, and have a function associate clic*FunctionName()* which take the inputs of the forms et stock it on data (or simopt for the parameters) when we clicked on the button. There are also function if select changed, but if they have a name, their name have the form of change*FunctionName*()

ResetGraph() : use the module Cystoscape to remove all nodes (so also all edge) of the graph.

UpdateGaph(): call ResetGraph() and then to add node and edge to correspond with the actual object data.

Processing\_data(results, indice): retrieve the result of the conversion from csv to js object made in the function clicChoiceStep(indice) by the module Papaparse, and add the second and third line to a list. This data is used for the simulation of multiple step. The indice corresponds to the asset which have these multiple time step.

checkname(choice): this function looks at if the name is already used for node (return true) or not (return false) choice can be the name or the input where the user writes the name.

change\_tab(name): this function changes the visible tab for the asset, (the visible tab become content\_tab\_name).

function change\_tab2(name): this function changes the visible tab between parameters and Market

function clicSaveEverything() : this function create an object data\_export which contain data and simopt and download it into a json file.

clicResetEverything(): this function offers to save everything and after confirmation set everything to their default value

hDelLinks(id\_node): this function deletes all links of the node which have this id (but don't change the nodes). So, this function is not made to be used alone (because even if the link disappears, the nodes remind partners or in the community)

hDelAgent(Id\_node, flag\_asset =true ): this function deletes the node who have this id; this will called the function hDelLinks; this will remove this id of all other partner's, admin's, member's nodes. If flag\_asset is set to false, the assets of the Agent will not be deleted. This feature is used when we modify an agent to don’t have to re-create the asset.

# Variables:

Data (for the market)

* link: list of Link (object) on the market
* asset: list of Asset (object) on the market
* node: list of Node (object) on the market
* idLinkUnused: list of link id not used (because we delete the link)
* idAssetUnused: list of asset id not used (because we delete the link)
* idNodeUnused: list of node id not used (because we delete the link)
* old\_onglet: global variable used for the visualisation of the asset tabs
* oldNbAsset: global variable used for the visualisation of the asset tabs
* old\_onglet2: global variable used for the visualisation of the presentation tabs
* multipleStep\_1: global variable to set Pmaxt (case flexible) or Pmaxt and Pmint (uncontrollable)
* multipleStep\_2: global variable to set Pmint (case flexible)

## simopt (for the simulation)

See the annexe part to have more descriptions of the different options.

### market:

* Stochastic: Boolean which indicate if the simulation must consider the uncertainty of the assets. False by default.
* Reserves: String of the counter measure used by the market against the uncertainty, “Balancing” by default.
* Reserve\_type: String of the type of reserve considered, P2P by default.
* confidence\_level: Number which indicate the confidence level they should cover, 0.95 by default.
* confidence\_repartition: how reserve confidence levels should be spread, if the reserve type is P2P, fairly by default.
* multiple\_time\_steps: Boolean which indicate if the market must consider multimer time step for the simulation, false by default.
* treat\_multiple\_times: String which indicate how the market treats the multiple time steps, sequentially by default.

### network:

* present: Boolean which indicate if the market considers the limitation of the network, false by default.
* integration: String which indicate how the market include the power system’s limitation, Exogenous by default.
* charge: String which indicate how the network charges are estimated if the integration is exogenous, Uniform by default
* unit\_fee: number which indicate the unit fee level (in $/MWh) if the integration is exogeneous, 1 by default.
* model: String which indicate the power line model used if the integration is endogenous, DC by default.

optimizer:

* centralized\_optimization: Boolean which indicate if the optimization must be resolved by a centralized manner of not, false by default.
* overwrite: Boolean which indicate if the optimizer can overwrite already exiting results, false by default.
* save\_in\_multiple\_files: Boolean which indicates if the computed results of each time step (if existing) must be saved in separate files, false by default.

## testcase:

* Imp\_fee, Exp\_fee, P2P\_fee: respectively the import, export, peer to peer fee that should be applied (in $/MWh), 0 by default.
* selected\_time: 1 or 2 Numbers that indicate what time steps (a step or an interval of steps) should be tested (If the test case has multiple ones). If the value in null, all times steps should be tested. By default, all times steps should be tested.

## SOparams:

* maxit: Number that indicate the maximum number of iterations of the coordination with the system operator, 200 by default.
* penalty: Number that indicate the penalty factor of the coordination with the system operator, 1 by default.
* espPrimR, espDualR: Numbers that respectively represent primal and dual residuals' tolerances of the coordination with the system operator?

Annexe

## General characteristics – Market

### Is the market stochastic?

true - The market knows that not all agents are fully controllable and offer counter measures.

false - The market supposes all agents to be certain or controllable. In this case uncertain agents such as renewable generators must trade carefully, e.g. by bidding the expectation or value at risk of their probability distribution function.

### If the market is stochastic, what counter measures are proposed?

'Stochastic' - The market is fully stochastic, hence agents bid probability distribution functions.

'Reserves' - Agents can buy/offer capacity reserves.

'Balancing' - Agents can compensate their uncertainty in a balancing market.

### What type of reserve trading do you want to consider?

'Pool' - Agents participate to a centralized pool reserve market.

'Sup' - Each agent trades reserves bilaterally with a set of partners in a decentralized manner under supervision.

'P2P' - Each agent trades reserves bilaterally with a set of partners in a decentralized manner.

### If P2P, how reserve confidence levels should be spread?

'Globally' - Confidence level of uncertain agents is all equal to the overall confidence levels.

'Homogenous' - Confidence level of uncertain agents is all equal to the overall confidence level divided by the number of uncertain agents.

'Heterogeneously' - Confidence level of uncertain agents is all equal and depend on the overall heterogeneity of uncertainties.

'Fairly' - Confidence level of uncertain agents is fairly spread with respect to their part in the overall uncertainty.

### If they are present, can the market treat multiple time steps?

true - The market can directly run on multiple time step test cases.

false - The market cannot handle multiple time steps and treats them

sequentially and independently.

### If able, how does the market treat multiple time steps?

'Completely' - The market is cleared directly on all time steps.

'Sequentially' - The market considers time steps sequentially with time binding constraints such as ramping constraints.

## General characteristics - Network

### Do you want the market to consider power system's presence?

true - The market considers power system's limitations.

false - The market assumes no physical limitations on between peers.

### If considered, how do you want the market to include power system's limitations?

'Nested' - The system operator endogenously steers trades and network charges directly after each market clearing.

'Integrated' - The system operator endogenously steers trades and network charges directly at each trade proposals.

'Exogenous’ - The system operator exogenously defines network charges before the market clearing.

### If exogenous, how do you want network charges to be estimated?

'Distance' - Network charges are individually applied per trade based on the equivalent electric distance.

'NumZone' - Network charges are individually applied per trade based on the number of zones which are crossed.

'Uniform' - Network charges are equal for all trades indifferently to grid usage.

### If endogenous, what power line model do you want to use?

'AC' - Power lines are model by their full nonconvex complex model.

'SDP’ - Power lines' AC model is convexified using Semi Definite Programming.

'SOCP' - Power lines' AC model is convexified using Second Order Cone Programming, which is tighter than SDP but slower.

'DC' - Power lines' AC model is linearized by solely considering the active power. Note that this model is not valid on distribution networks.

General characteristics - Optimizer

### Should the optimization be solved using a unique centralized global solver?

true - The overall problem is solved in a central manner.

false - The overall problem is split in smaller local problems based on ADMM decomposition, hence resulting in a distributed or decentralized algorithm.

### Can the optimizer overwrite already exiting results?

true - The optimizer will run and save all simulations whether result file already exists or not.

false - The optimizer only runs and saves simulations which have not been computed already.

### Should the computed results of each time step (if existing) be saved in separate files?

true - Each time step result is saved in a specific file.

false - Every time step results are saved in a single file.