Structure

Object type specification
NOTE! Contains information on

(a)	*existing elements*	***plc***	*→ in place, no investment cost have to be considered*
(b)	*design options*	***opt***	*→ maximum scope, prior to optimization run*
(c)	*optimum setting*	***set***	*→ of system design, after optimization run*

Table 1 Object types

#	Scope	Abbr[1]	Remarks
1	Project	***prj***	General project (design+operation) related general info (+ data)
2	System(s)	***sys***	System alternatives / scenarios → contain actual instances of networks, DER + loads.
3	Node(s)	***nde***	Nodes are locations, where technologies link to (sub-)domain buses of networks.
4	Domain(s)	***dmn***	*prs* + *opt* + *set* for network ***ntw*** nodes Domain refers to types of energy carriers (can include various sub-types defined by different characteristics like potential or AC/DC). Domain layers are characterized by network infrastructure (buses + connections)[2]. Pure resources without infrastructure (e.g. onsite PE or SE without system internal infrastructure) are just virtual domains with virtual buses connecting to inputs.
5	Network(s)	***ntw***	[= Grid(s) | Structure] defining network instance(s) as graph with nodes (= domain buses) + connections
6	Resource(s)	***rsc***	*prs* + *opt* + *set* for network ***ntw*** nodes • Physical: PE | SE (= energy draft) • Economic: Income, Financing
7	Component(s)	***cmp***	*prs* + *opt* + *set* for instances of networks ***ntw*** | components ***cmp*** | loads ***loa*** • All DER (without spatial extension); y-x-plane • Connections (with spatial extension); z-x-plane
8	Load(s) / final demand	***loa***	*prs* + *opt* + *set* for network ***sys*** nodes • Fixed loads (fixed value for every time-step) Given as time-series or to be built from parameters Flexible loads (shiftable | curtailable) Final demand is given as load curve (can include range min/max) with an upstream component (converter/storage)
9	Stakeholder(s)	***stk***	*prs* + *opt* + *set* for instances of networks ***ntw*** | components ***cmp*** | loads ***loa*** Owners, operators

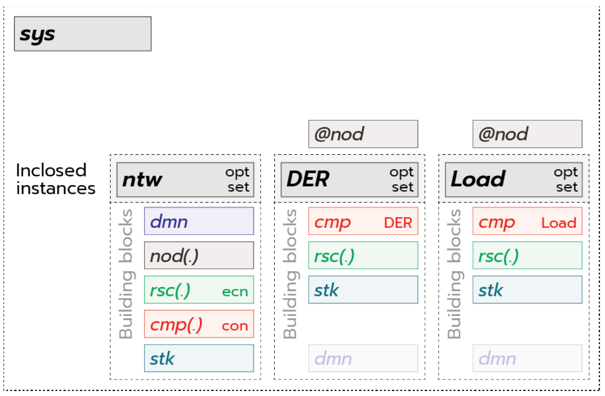

Figure 1 Assembling instances from building blocks

[1] 3-letter abbreviation, used to keep things small.
[2] It follows: if no system internal infrastructure needs to be considered (e.g. natural gas provided as fuel to just one device from an external network) → no domain needs to be defined explicitly (just virtually)

Table 2 General properties | attributes of objects

#	obj	Spec	Content	Fields relevant for[3]		
				1 GUI	2 MW	3 SE
1		***idx***	*Index of instance* *(necessary for reference)*	?	?	x
2		***nme***	*Name of instance* *(optional)*	?	?	?
3		***des***	*Description of instance* *(optional)*	?	?	?
4		***lnk***	*Link to element in library*	?	?	x
			or			
4		***typ.***[4]	*Modeling type*	?	?	x
		'.idx	[<# level_1> <# level_2> … <# level_n>]	x	?	–
		'.nme	*A full + short name*[5]*, related to type* {'<Full_level_1>',…,'<Full_level_n>'; '<Short_level_1>',…,'Short_level_n'}	x	?	–
		'.des	*A short verbal description*[6]*:* {'Lorem ipsum …'}	x	?	–
5		***mnf.***	*Manufacturer related info (for **cmp** only)* [7]	x	?	–
		'.idx	*???*	x	?	–
		'.nme	{'<manufacturer/provider XY>', '<model YZ>'}	x	?	–
		'.des	{'<model YZ specific info>'}	x	?	–

Each object has a header containing the fields given in Table 2.-The scope (***phs/tcn***, ***ecn/mng***, ***env***) will be indicated by an extra flag.

header
instance_index = …
type_index = …
type_name = …
type_description = …
further attributes

Figure 2 Attributes and classification for objects in general

For each attribute additional information is provided as shown in Table 3. Note that not all fields need to be defined in each case.

Table 3 Common fields for each specific attribute of objects, distinguishing numerical + non-numerical data / info

#	obj	Spec	Content	Example	Numeric?		Relevant for		
					yes	no	GUI	MW	SE
1		***nme***	Full + short name	{'Latitude', 'Longitude', 'Time zone'; 'lat', 'lon', 'time'}	x	x	x	?	–
2		***des***	Short verbal description	{'Geographic coordinate specifying the north-south position of the point on the Earth's surface.', 'Geographic coordinate specifying the east-west position of the point on the Earth's surface.', 'Time zone indicated as offset to Coordinated Universal Time (UTC), e.g. GMT = 0, CET = +1 and EST = -5.'}	x	x	x	?	–

[3] Content of field might of relevance for (1) GUI - user oriented information, (2) MW – middleware with data management oriented information, (3) SE – simulation engine (modelling + solving oriented info)
[4] This is modeling type info, ***'.nme*** and ***'.des*** are internally linked to ***'.idx***.
[5] A short name / abbreviation is always given in the second row
[6] This could be shown in the GUI behind a help button or in a dedicated field.
[7] In case of compound components → main component.

3	***scp***	Scope[8]	*First #: 1 phs/tcn, 2 ecn/mng, 3 env* *Second: 1 set-up, 2 operation* *e.g.* [1 1]	x	x	x	?	x
4	***val***	Value(s)[9]	[52.567, 13.345, 1]	x	x	x	?	x
5	***dtp***	Data type	{'float', 'float', 'int'}	x	–	–	?	–
6	***unt***	Unit[10]	{'degree', 'degree', ''} *NOTE! degree = °*	x	–	x	?	x
7	***upp***	Upper boundary[11]	[90, 180, 12]	x	–	–/x	?	x/–
8	***low***	Lower boundary	[-90, -180, -12]	x	–	–/x	?	x/–

General remarks:

- Fields can be empty. Those not relevant can be ignored. If important info is missing, either a best guess or a (pre-defined) default value is (to be) applied.

General rules:

- Specific values have higher priority than general ones, e.g. ***rsc@cmp*** > ***rsc@nde***
- If an object usually links to another object (type), e.g. a ***rsc*** has a ***stk*** and is available for specific ***cmp***, and no link is given, then this object is not linked (to any type of the other object). If the object should be linked to all types of the other object, then a link must be set with value 0.

[8] Indicates the most fitting scope. NOTE! For orientation only, might be used to structure info
[9] Can be a single value, a row of values, and even a time-series (with leading time-stamp).
NOTE! If scope [x 1] (= configuration): first row = ***set***; second row = ***opt***, giving a range between min and max either continuous [*min max*] or discrete [*min val2 val3 … max*].
[10] Also: a format explaining units, e.g. „yyyy-mm-ddTHH:MM:SS" for time stamp, according to ISO 8601
[11] Boundaries used for error checking: Related to nature of quantity/entity (NOT a modeling-specific value), further ranges of validity would be indicated in additional rows.

1 Project

General project (design+operation) related info (+ data). Input data only.

Table 4 Object project ***prj*** – attributes and grouping of attributes

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# in (user-specific) prj lib*	–			
2	***nme***		{'Berlin City District Microgrid'; 'BCDMG'}				
3	***des***		{'The BCDMG project is developed in the heart of the City …'}				
	typ.		*If the user would like to apply a classification of own projects.*				

prj > sys(s) | "libs":

- Totally specific to project / location
  - Nodes,
  - Networks,
- Broader scope:
  - Resources,
  - Loads,
- General:
  - (Standard-) domains,
  - Components (technologies).

2 System

The ***sys*** object contains all the

They can be several ***sys*** instances representing different scenarios that can differ all aspects with respect to system design and operation.

How to deal with modeller/solver settings with respect to different scenarios?

Input + output data → all system related information, the ***sys*** object defines:

- Instances of nodes (***nde***, from ***nde*** lib, or directly) with aligned
  - Instances of resources (***rsc***, from ***rsc*** lib)
  - Instances of components (***cmp***, from ***cmp*** lib) with aligned
    - Instances of resources (***rsc***, from ***rsc*** lib)
    - Instances of stakeholders (***stk***, from ***stk*** lib)
  - *Anything else?*
- Instances of networks ***ntw*** (from ***ntw*** lib, or directly?) with aligned

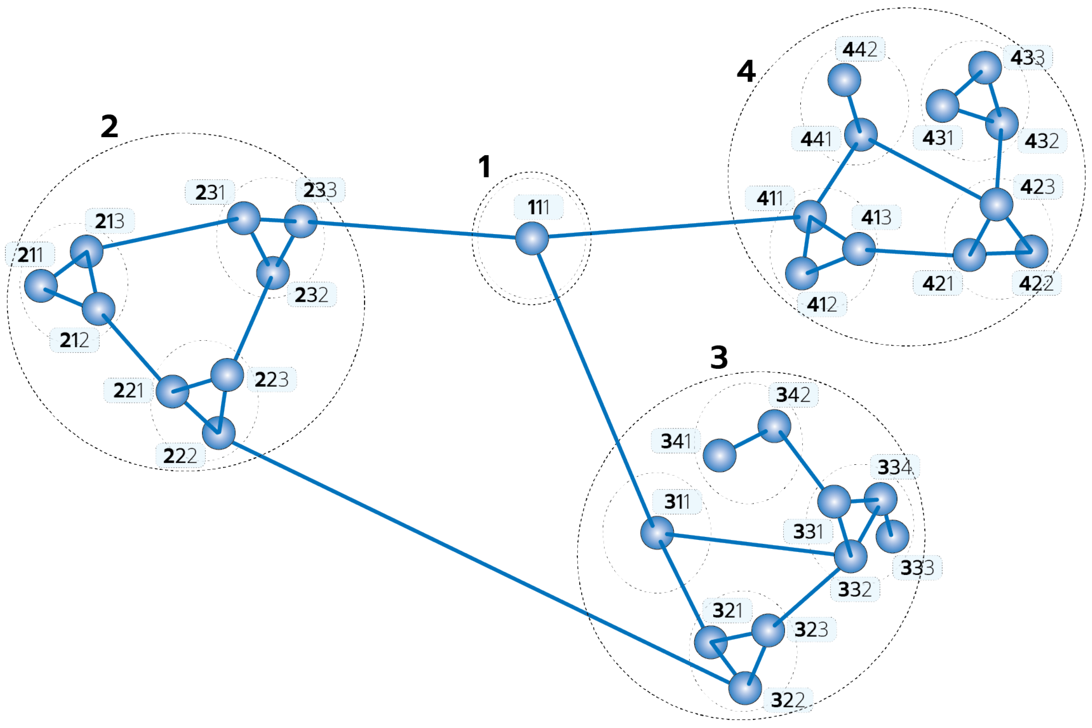

Figure 3 Nodes with indices (***idx***) in a network with (sub-) clusters

Table 5 Objects system ***sys****(s)* – attributes and grouping of attributes

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# of system (alternative)*	–			
2	***nme***		{'BCDMG – Version 1'; 'BCDMG1'}				
3	***des***		{'The first alternative of the BCDMG contains …'}				
	typ.		*If the user would like to apply a classification of systems.*				
	nod(n).		*Instances of nodes*[12]				
	'.lnk		*# of* **nde** *in* **nde** *lib*				
	'.<xxx>		*Additional info*				
	'.opt.[13]		*Options → shall it be considered for optimum system?*				
	''.prs	***nme***	{'Min presence', 'Max presence'; ' ', ' '}				
		des	{'Defines, if node is present in system.'}				
		scp	[1 1]				
		val	[0 1] → *to be considered*				
		dtp	{'bin'}				
		upp	[1]				
		low	[0]				
	'.set.		*Result → is it set in the optimum system?*				
	''.prs	***nme***	{'Presence; ' '}				
		des	{'Defines, if node is present in system.'}				

[12] Instead of linking to the ***nde*** "lib", also additional nodes can be defined with ***psn***, ***idx*** and ***plc***.
[13] For ***opt***, always a range is given for both ***prs*** and ***cap***. For ***prs*** this means that the user either sets the option for the solver to decide to whether include the element or not → [0 1] (default), or determines prior to the optimization run if the element will be included in the system → [1 1] or not → [0 0]. (only applicable for elements where it does make sense).

	scp	[1 1]				
	val	[1] → *is set*				
	dtp	{‘bin’}				
	upp	[1]				
	low	[0]				
‘.rsc(r).		*Instances of resources @node*[14]				
“.idx		*# of **rsc** @**nde***	–			
“.lnk		*# of **rsc** in **rsc** “library”*				
“.<xxx>		*Additional info*				
“.plc.						
“‘.prs	*des*	{‘Defines, if resource is present at node.'}				
“.opt.						
“.set.		*just for info: **rsc** used by “someone” @**nde**?*				
‘.cmp(c).		*Instances of components @**nde***[15]				
“.idx		*# of **cmp** @**nde***	–			
“.lnk		*# of **cmp** in **cmp** “library”*				
“.cpf	*nme*	{‘Capacity factor’}				
	des	{‘Factor, the actual component capacity should be greater as, to ensure availability of supply, taking into account uncertainty of forecasted demand. Does only effect the economic figures.'}				
	scp	[1 2]				
	val	[1]				
	dtp	{‘float’}				
	unt	{‘–’}				
	upp	[Inf]				
	low	[0]				
“.<xxx>		*Additional info*				
“.stk(s).		***stk @cmp***				
“‘.idx		*# of **stk** @**cmp***				
“‘.lnk		*# of **stk** in **stk** lib*				
“‘.<xxx>		*Additional info*				
“.rsc(s).		***rsc @cmp***				
“‘.idx		*# of **rsc** @**cmp***				
“‘.lnk		*# of **rsc** in **rsc** “library”*				
“‘.<xxx>		*Additional info*				
“.plc.		*of **cmp** instance @**nde***				
“‘.prs[16]	*des*	{‘Defines, if component is present at node.'}				
	val	[1] → *already installed*				
“‘.cap	*nme*	{'Capacity’; ' '}				

[14] This mainly addresses PE resources (weather data).
[15] In case of discrete technologies the instance ***cmp***(c) wraps all singular units.
[16] NOTE! At single nodes, for elements for which also a ***cap*** value is given the ***prs*** info is already included in the ***cap*** value (and could therefore can be omitted (since already given with ***cap***).

		des	{'Aggregated capacity of this component (type) at node.'}
		scp	[1 1]
		val	[50] → *already installed: 50 kW*
		dtp	{‘float’}
		unt	{‘kW’}
		upp	[Inf]
		low	[0]
	“.opt.		*of **cmp** instance @**nde***
	“’.prs	*val*	[1 1] → *to be set*
	“‘.cap	*nme*	{'Min capacity’, 'Max capacity’; ' ', ‘ ‘}
		des	{'Minimum aggregated capacity of this component (type) at node, if installed.', 'Maximum aggregated capacity of this component (type) at node, if installed.'}
		scp	[1 1]
		val	[100 500] → *if installed, betw. 100 – 500 kW*
		dtp	{‘float’}
		unt	{‘kW’}
		upp	[Inf]
		low	[0]
	“.set.		*of **cmp** instance @**nde***
	“’.prs	*val*	[1] → *is set*
	“‘.cap	*val*	[500] → *installed: 500 kW*
	“‘.cpx.		*CAPEX*
	“‘.opx.		*OPEX*
	“‘.mod(m).		*Modes*
	“”’.inp(i)	*nme*	{'Time stamp', 'Power'; ' ', ' '}
		val	{'2017-01-01T00:00:00', '23.5'; ... '2017-12-31T23:45:00', '9.2'}
		unt	{'yyyy-mm-ddTHH:MM:SS', 'kW'}
	“”.out(o)		*Schedules of outputs*
	“”.soc		*Schedule of SOC*
	“”.tme	*nme*	{'Time stamp', 'On/off'; ' ', ' '}
		val	{'2017-01-01T00:00:00', '1'; ... '2017-12-31T23:45:00', '1'}
		unt	{'yyyy-mm-ddTHH:MM:SS', '–'}
	ntw(n).		*Instances of networks in system*[17]
15	***‘.idx***		*# of **ntw** in **sys***
16	***‘.lnk***		*# of **ntw** in **ntw** “library”*
	‘.plc.		
	“.prs	*des*	{‘Defines, if network is present at system.'}

[17] NOTE! This refers to a network as a whole. For single elements (buses + connections), ***plc***, ***opt*** and ***set*** are defined in the networks “library”.

		val	[0] → *not yet placed*				
	‘.opt.						
	‘’.prs	*val*	[0 1] → *to be considered*				
	‘.set.						
	‘’.prs	*val*	[1] → *is set*				
	‘.<xxx>		*Additional location-specific info not covered in **ntw** lib*				

In addition to defining *actual* nodes, *virtual* nodes can be defined (aggregating info shared among low-level (sub-)nodes at higher level), as shown in Table 6. This feature allows addressing whole sub-systems with constraints and assignments, e.g. restricting installation capacities, applying resources and so on. Thus, it is mainly of interest for options ***opt***, but might also cover existing systems (***plc***) and summarized results (***set***).

Table 6 Objects system ***sys****(s)* – examples of virtual nodes, containing info for all sub-ordinated actual nodes

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
	nod(n).		→				
	‘.idx		[2 3] → *related to all nodes [2 3 …]*				
	‘.typ		*Modeling type*				
	‘’.idx		[3]				
	“.nme		{‘Virtual node’}				
	“.des		{‘Virtual aggregation of sub-nodes. For the definition of sub-node overarching characteristics.’}				
	‘.rsc(r).		*Instances of resources @node*				
11	*“.idx*		*# of resource @node*	–			
12	*“.lnk*		*# of resource in rsc “library”*				
	‘.cmp(c).		*Instances of components @node*				
	“.idx		*aggregated info → no # of component @node available*				
	“.lnk		*# of **cmp** in **cmp** lib*				
	“.typ.		*Modeling type*				
	“’.idx		[1 2] → *related to all components [1 2 …] ([A B] → [EL ND])*				
	‘.plc.		*optional*				
	‘.opt.						
	“.prs	*val*	[0 2] → *considered for installation @ up to 2 sub-nodes*				
		dtp	{‘int’}				
		upp	[Inf]				
		low	[0]				
	“.cap	*val*	[100 500] → *if installed, betw. 100 – 500 kW*				
		dtp	{‘float’}				
		unt	{‘kW’}				
		upp	[Inf]				
		low	[0]				
	‘.set.		*optional*				

3 Nodes

Nodes are the spatial locations where network buses and technologies are installed. They are abstract objects with no effort, cost, ownership and so on. Nodes are very useful to apply certain concepts with spatial relevance such as the allocation of local PE resources, virtual contraction / aggregation of system elements and even characterizing a vertically hierarchical structure within systems.

Since the ***sys*** object (section 2) can potentially have several instances (= scenarios) that may apply the same nodes, these cannot only be defined within ***sys*** itself. Instead, scenario-overarching nodes are "stored" in the ***nde*** "library". The lib contains all already existing nodes (in case of system upgrading) and base nodes applied in all the scenarios. Within each ***sys*** object, additional nodes can be integrated/defined. Nodes are structured hierarchically, as shown in Figure 3. However, the structure is treated as flat, the hierarchy is only indicated in the nodes' indices. In the ***nde*** "library", this hierarchy of nodes, is already implemented/given[18],

Following these guidelines, nodes to be "stored" in the ***nde*** "library" should be given at least the following attributes:

- Hierarchical position (***idx***),
- Spatial position (***psn***),
- Presence in existing system (***plc***), if applicable.

Other attributes can be defined/stored in here, if applicable to all scenarios, or in the ***sys*** object[19]:

- Linked resources (***rsc***),
- Linked components (***cmp***).
- Presence in optimization options (***opt***) and optimized system (***set***),

Table 7 Objects nodes ***nde****(n)* – attributes and grouping of attributes

#	Group / attribute	field	Content / example	Num ?	Relevant for GUI	Relevant for MW	Relevant for SE
1	***idx***		[2 3 1][20]				
2	***typ.***		*Modeling type*				
	'.idx		[1] → *1 – default node, 2 – construction node, 3 – virtual node, 4 – system boundary node*				
	'.nme		{'Default node'}				
	'.des		{'System-internal node components can be installed at. '}				
	psn.		*Geographical / spatial position*				
	'.abs[21]	***nme***	{'Latitude', 'Longitude', 'Time zone'; 'lat', 'lon', 'time'}				
		des	{'Geographic coordinate specifying the north-south position of the point on the Earth's surface.', 'Geographic coordinate specifying the east-west position of the point on the Earth's surface.', 'Time zone indicated as offset to Coordinated Universal Time (UTC), e.g. GMT = 0, CET = +1 and EST = -5.'}				
		scp	[1 1]				

[18] It is assumed, that the user-defined nodes hierarchy considers real local physical conditions equally important for the structure of different ***ntw*** of various ***dmn***. Thus, the hierarchy can already defined here in the ***nde*** "lib", which the ***ntw*** "lib" links to.
[19] All attributes can be stored in lib, and then overwritten individually in ***sys*** object. (General rule)
[20] 1 = default node @level; e.g. [2 3 1] → this is the first node in the third sub-cluster of the second main cluster.
[21] The ***abs*** value for a node can be calculated from ***rel*** and vice versa, except for default node.

	val	[52.5141932,13.3268113,1]	x			x
	dtp	{‘float’, ’float’, ‘int’}				x
	unt	{‘degree’, ’degree’, ‘’}				x
	upp	[90, 180, 12]				
	low	[-90, -180, -12]				
‘.rel[22]	*nme*	{'Relative distance N-S', ''Relative distance E-W'; 'relN-S', relE-W'}				
	des	{‘Relative distance to default node in north-south direction (y-axis)’, ‘Relative distance to default node in east-west direction (x-axis)’}				
	scp	[1 1]				
	val	[200,50]	x			x
	dtp	{‘float’, ’float’}				x
	unt	{‘m’, ‘m’}				x
	upp	[ ]				
	low	[ ]				
.plc.						
‘.prs	*nme*	{‘Presence; ' '}				
	des	{‘Defines, if node is present in system.'}				
	scp	[1 1]				
	val	[0] → *not yet present*				
	dtp	{‘bin’}				
	upp	[1]				
	low	[0]				

[22] Relative to default node with ***idx*** [1 1 1 …], meaning that for the default node the absolute value is needed (= location of project).

4 Domains

This is the user-determined project-specific “library” of available domains to be considered. It can just comprises main domains such as electricity or thermal heat but can also include more specific sub-domains characterized by e.g. potential and, in case of the electric domain, AC/DC. Classification of domains, resulting in type # ***typ***, is listed in the appendix, Tab. A 2.

Table 8 Objects domain ***dmn****(d)* – attributes and grouping of attributes

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# of available (sub-) domain (option)*	–			
2	***typ***		*Modeling type*				
	‘.idx		[1 1 1] → *[EL|LV|AC]*				
	‘.nme		{‘Electricity’, ‘Low Voltage’, ‘Alternate Current’; ‘EL’, ‘LV’, ‘AC’}				
	‘.des		{‘Low voltage AC network with ...’}				
3	***rpt***[23]	***nme***	{'Rated voltage'; 'ratV'}				
		des	{'Rated voltage level of system.'}				
		scp	[1 1]				
		val	[0.4]				
		dtp	{‘float’}				
		unt	{‘kV’}				
		upp	[Inf]				
		low	[0]				
4	***<xxx>***		*additional*	–			

[23] Rated potential. The available capacity depends on the rated capacity of applied line components.

5 Networks[24]

This is the user-determined project-specific “library” of available networks to be considered in the project. A network is defined by a domain type (Tab. A 1 + Tab. A 2) and its design, a graph based on buses and connections between. The network buses are located at nodes. Connections are physically built with available line components (Tab. A 4). Furthermore, networks are owned/operated by certain stakeholders (Tab. A 6), and can be linked with available resources (Tab. A 3).

If a network is to be use in one specific system alternative only, it could be defined in the ***sys*** object directly, instead of the ***ntw*** lib.

Following these guidelines, for networks to be “stored” in the ***ntw*** “library” at least the following attributes should be given:

- Type, which links to the ***dmn*** lib (***typ***),
- Available buses, which link to the ***nde*** lib (***bus***),
- Available connections between buses (***con***) and the applied physical lines (***cmp***),
- Presence in existing system (***plc***), if applicable.

Other attributes can be defined/stored here, if applicable to all instances in all scenarios. Otherwise, in the ***sys*** object[25]:

- Linked stakeholders as network owners/operators (***stk***),
- Linked resources (***rsc***),
- Presence in optimization options (***opt***) and optimized system (***set***),

Table 9 Objects network ***ntw****(n)* – attributes and grouping of attributes

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# of **ntw** in **ntw** lib*	–			
2	***nme***		{‘Network’}				
3	***des***		{‘ ‘}				
4	***lnk***		*# of **dmn** in **dmn** lib*				
	plc.		***ntw** already placed in potentially existing **sys**?*				
	bus(b).		*Available buses (@nodes)*				
	‘.idx		*# of **bus** in **ntw***				
	‘.lnk		*# of **nde** in **nde** lib*				
	con(c)		*Available connections*				
	‘.idx		*# of **con** in **ntw***				
	‘.lnk		[3 5] *#s of start + end **bus***				
	‘.lgt	***nme***	{‘Length’; ‘L’}				
		des	{‘One way length of connection (default value: linear distance). Can be user-defined, in case the real distance is larger due to topological reasons.'}				
		scp	[1 1]				
		val	[250]				
		dtp	{‘float’}				
		unt	{‘m’}				

[24] Alternative: Grids, Topologies, Structures
[25] All attributes can be stored in lib, and then overwritten individually in ***sys*** object. (General rule)

	upp	[Inf]	
	low	[0]	
	‘.plc.		*con already placed in potentially existing ntw?*
	“.prs	*des*	{'Defines, if connection is applied in the existing network.'}
	‘.cmp(c)		*Instances of included component (physical line element)*
	“.idx		*# of **cmp** (LN) in **con**) → very likely only <u>one</u>*
	“.lnk		*# of **cmp** in **cmp** lib*
	“.plc.		*cmp (LN) already placed in potentially existing ntw?*
	‘“.prs	*des*	{'Defines, if this line element is used for this connection in the existing network.'}
	“.<xxx>		*Certain extrinsic info, if necessary*
	rsc(r).		*Aligned resources*[26]
	‘.idx		#
	‘.lnk		*# of **rsc** in **rsc** lib*
	‘.plc.		
	“.prs	*des*	{'Defines, if resource is applied in the existing network.'}
	‘.<xxx>		*Certain extrinsic info, if necessary*
	stk(s).		*Stakeholder*[27]
	‘.idx		*# of **stk** in **ntw***
	‘.lnk		*# of **stk** in **stk** lib*
	‘.<xxx>		*Certain extrinsic info, if necessary*

Must define grid fees (*Netzentgelte*) for the usage by third parties (energy transmission) → coupled to ***ntw*** directly or ***stk***?

[26] This mainly addresses economic resources, e.g. financing.
[27] This is a fix assignment, no optimization option.

6 Resources

This is the user-determined project-specific “library” of available resources to be considered in the project. The addressable types of resources are classified in the appendix, Tab. A 3.

Resource options are **applicable to** (available for):

- Physical resources:
  - Single components ***cmp*** and loads ***loa***,
  - Nodes, meaning: all (fitting) ***cmp***+ ***loa*** connected at that node (and related sub-nodes)[28]
- Economic resources:
  - Single components ***cmp*** and loads ***loa***,
  - Networks.

<u>Who</u> provides ***rsc*** to <u>whom</u>?	→ ***stk***	economic, cash flow (if applicable)	←
<u>What</u> provides ***rsc*** to <u>what</u>?	→ ***cmp***	physical, energy flow (if applicable)	→

Following these guidelines, for resources to be “stored” in the ***src*** “library” at least the following attributes <u>should</u> be given:

- Resource type (***typ***),
- For time-series resources (similar to loads, cf. section 8):
  - Duration of time-series (***drt***) + the assemblage of single periods within (***asm***),
  - These single periods (***prd***) with
    - Default values of power (time-series) (***pwr***) and energy demand (***ene***),
    - Ranges defining cost (***rng***)[29].
    - Type-specific info (***spc***) to build the time-series by a separate simulation[30].
- For non time-series resources (economic|financing):
  - *t.b.d.*

Other attributes <u>can</u> be defined/stored here, if applicable to all instances in all scenarios. Otherwise, in the ***sys*** object[31]:

- Presence in existing system (***plc***), optimization options (***opt***), and optimized system (***set***),
- Linked stakeholders (***stk***) as resource providers,
- Linked components (***cmp***) as potential resource recipients,
- Linked loads (***loa***) as potential resource recipients,

Table 10 Objects resource ***rsc****(r)* – “Library” of available resources – Example 1: **Electricity (tariff)**

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# of resource (option)*	–			
2	***typ.***		*Modeling type*				
	‘.idx		[1 1] *→ [PH|EL] → Provision of electricity (with tariff)*				
	‘.nme		{‘Physical resource’, ‘Electricity’; ‘PH’, ‘EL’}				
	‘.des		{'Electricity tariff.'}				
	stk(s).		*Links to aligned stakeholder*[32]				

[28] NOTE! Only for ***rsc*** w/o ownership / free of charge (= on-site PE) → nodes don’t have ***own***/***opr***

[29] At least <u>two</u> ranges should be given: one stating the default cost, the other defining VoLL.

[30] Of interest for PE resources.

[31] All attributes can be stored in lib, and then overwritten individually in ***sys*** object. (General rule)

[32] **Provider** of ***rsc***. In case of physical resource: provides energy, receives payment; economic resource: provides payment, receives return service.

‘.idx		*# of **stk** in **rsc** → very likely only <u>one</u>*
‘.lnk		*# of **stk** in **stk** lib*
‘.<xxx>		*Additional info*
cmp(c).		*Links to allowed components*[33]
‘.idx		*#*
‘.lnk		*# of **cmp** in **cmp** lib*[34]
‘.typ.		*Modeling type*
“.idx		[2 1 1 2] *→ [HT|DP|EL|HP] → for compression heat pumps*
‘.<xxx>		*Additional info*
loa(l).		*Links to allowed loads*[35]
‘.idx		*#*
‘.lnk		*# of **loa** in **loa** lib*
‘.<xxx>		*Additional info*
drt	***nme***	{‘From’, ’To’, ’Duration’; ’From’, ‘To’, ‘Dur’}
	des	{‘Starting time (beginning point), absolute time’, ‘Ending time (ending point), absolute time’, ‘Duration in hours’}
	scp	[1 2]
	val	{‘2017-01-01T00:00:00’, ‘2018-01-01T00:00:00’, ‘8760’}
	dtp	{‘char’, ‘char‘, ‘float’}
	unt	{‘yyyy-mm-ddTHH:MM:SS’, ‘yyyy-mm-ddTHH:MM:SS’, ‘hours’}
	upp	[Inf Inf Inf]
	low	[0 0 0]
asm[36]	***nme***	{‘From’, ‘Resource period element’; ‘time’, ‘RE’}
	des	{‘Starting time (beginning point), absolute time’, ‘’}
	scp	[1 2]
	val	{‘2017-01-01T00:00:00’, ‘1’; … ‘2017-12-22T00:00:00’, ‘1’} *→ idx # of element*
	dtp	{‘char’, float‘}
	unt	{‘yyyy-mm-ddTHH:MM:SS’, ‘–’}
	upp	[Inf Inf]
	low	[0 0]
prd(p).		*Single resource period elements (can be applied repeatedly)*
‘.idx		[1]
‘.typ.		*Modeling type*
’’.idx		[0] *→ t.b.d., e.g. 0 – indifferent, 1 – WD, 2 – WE, 3 – spc.*
“.nme		{‘ ‘} *→ {‘Heat pump tariff.’, ‘–’}*
“.des		{‘ ‘} *→ {‘Electricity tariff for residential heat pumps.’}*
‘.drt[37]	***nme***	{‘From’, ’To’, ’Duration’;’ From’, ‘To’, ‘Dur’}

[33] **Allowed recipients** of ***rsc***. In case of physical resource: receives energy, provides payment; economic resource: receives payment, provides return service.
[34] Not relevant in this case since a ***cmp*** type is addressed.
[35] Like for ***cmp***: **Allowed recipients** of ***rsc***.
[36] Total resource profile assembled form single period elements (linking to idx #s)
[37] Same structure as ***drt*** of total load, start and end are given in plain (relative) hours.

	des	{‘Starting time (beginning point), relative time’, ‘Ending time (ending point), relative time’, ‘Duration in hours’}
	scp	[1 2]
	val	[0 24 24]
	dtp	{‘float’, float‘, ‘float’}
	unt	{‘hour’, ‘hour’, ‘hours’}
	upp	[Inf Inf Inf]
	low	[0 0 0]
‘.pwr[38]	***nme***	{‘Time (step)’, ‘Max power’; ‘time’, ‘P’}
	des	{‘Time stamp (beginning of time-step).’, ‘Upper peak power boundary per time-step.’}
	scp	[1 2]
	val	[0 1000][39]
	dtp	{‘float’, float‘}
	unt	{‘hour’, ‘kW’}
	upp	[Inf Inf]
	low	[0 0]
‘.ene	***nme***	{‘Max energy’; ‘E’}
	des	{'Upper boundary of cumulated energy.'}
	scp	[1 2]
	val	[10000]
	dtp	{‘float‘}
	unt	{‘kWh’}[40]
	upp	[Inf]
	low	[0]
‘.bsc[41]	***nme***	{'Base cost – band', 'Base cost – period'; 'BCb', 'BCp'}
	des	{‘Base cost, if applied in any instance of this potentially repeated period (EUR/band).’, ‘Base cost, if applied in this instance of the period (EUR/period).’}
	scp	[2 2]
	val	[60 0]
	dtp	{‘float’, ‘float‘}
	unt	{‘EUR’, ‘EUR’}
	upp	[Inf Inf]
	low	[0 0]
‘.opc[42]	***nme***	{‘Time stamp’, ‘Energy cost'; ‘Time’, 'EC'}
	des	{‘Time stamp (beginning of time-step).’, ‘Energy-specific cost.’}
	scp	[2 2]

[38] (Peak) power, which resource (tariff) is valid for (related to one single addressee ***cmp***/***loa***). Can be a constant value or time-series (→ hours of non-availability can be defined, e.g. for HP tariffs)
[39] Here: One constant single upper boundary value (1,000 kW) valid for each time step in the period. The lower boundary value is given by upper boundary of adjacent range (= 0, if no range is defined).
[40] Energy demand (in period).
[41] Base cost (in period).
[42] Energy cost (in period).

#	Group / attribute	field	Content / example	Num ?	GUI	MW	SE
		val	[0 0.25] → *flat tariff (one constant value), otherwise time-series*				
		dtp	{'float', float'}				
		unt	{'h', 'EUR/kWh'}				
		upp	[Inf Inf]				
		low	[0 0]				
	'.rng(r).		*Separate ranges*[43] *→ not applied here*				

Table 11 Objects resource ***rsc****(r)* – "Library" of available resources – Example 2: **PE resource (GHI)**

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# of resource (option)*	–			
2	***typ***[44]		*Modeling type*				
	'.idx		[1 5 1] → *[PH|PE|SL] → Time-series of solar data*				
	'.nme		{'Physical resource', 'Primary Energy', 'Solar'; 'PH', 'PE', 'SL'}				
	'.des		{'Time-series data of solar energy.'}				
	stk(s).		*Stakeholder*[45]				
	'.idx		#				
	'.lnk		*# of stakeholder (owner + operator) in stk "library"*				
	'.<xxx>		*Additional info*				
	cmp(c).		*Component*				
	'.idx		#				
	'.lnk		*# of **cmp** in **cmp** lib*				
	'.typ.		*Modeling type*				
	''.idx		[0] → *not restricted*				
	'.<xxx>		*Additional info*				
	loa(l).		*Load*				
	'.idx		#				
	'.lnk		*# of **loa** in **loa** lib*				
	'.typ.		*Modeling type*				
	''.idx		[0] → *not restricted*				
	'.<xxx>		*Additional info*				
	drt		*Same as in Example 1*				
	asm		*Same as in Example 1*				
	prd(p).		*Single resource period elements*				
	'.idx		[1]				
	'.typ.		*Modeling type*				
	''.idx		[3 5]				
	''.nme		{'Global Horizontal Irradiance', 'Diffuse Horizontal Irradiance'; 'GHI', 'FHI'}				
	''.des		{'Global Horizontal Irradiance, given in W/m2.', 'Diffuse Horizontal Irradiance, given in W/m2.'}				

[43] Defining additional ranges with related costs. See example in loads, section 8, Table 19
[44] In case only 3 digits are given in ***typ*** (PE-resource main-type like SL or WD), the ***prd.typ*** lists all the elements of the forth position digits included in the data (sequence taken over in ***prd.pwr***.
[45] **Provider** of ***rsc***. PE resources (weather data): just provider of data, e.g. NASA, DWD (?)

	‘.drt[46]	*nme*	{‘From’, ’To’, ’Duration’;’ From’, ‘To’, ‘Dur’}				
		des	{‘Starting time (beginning point), absolute time’, ‘Ending time (ending point), absolute time’, ‘Duration in hours’}				
		scp	[1 1]				
		val	{‘xxx0-01-01T00:00:00’, ‘xxx1-01-01T00:00:00’, ‘8760’}				
		dtp	{‘char’, ‘char‘, ‘float’}				
		unt	{‘yyyy-mm-ddTHH:MM:SS’, ‘yyyy-mm-ddTHH:MM:SS’, ‘hours’}				
		upp	[Inf Inf Inf]				
		low	[0 0 0]				
	‘.pwr[47]	*nme*	{‘Time stamp’, ‘Global Horizontal Irradiance’, ‘Diffuse Horizontal Irradiance’; ‘time’, ‘GHI’, ‘FHI’}				
		des	{‘Time stamp (beginning of time-step)’, 'Global Horizontal Irradiance, given in W/m2.', 'Diffuse Horizontal Irradiance, given in W/m2.'}				
		scp	[1 1]				
		val	{'xxx0-01-01T00:00:00', '0', '0'; … 'xxx1-01-01T00:00:00', '0', '0'}				
		dtp	{'float', float', 'float'}				
		unt	{'hour', 'W/m2', 'W/m2'}				
		upp	[Inf Inf Inf]				
		low	[0 0 0]				
	‘.ene		*If capacity restrictions → not needed here*				
	‘.bsc		*If base cost → not needed here*				
	‘.opc		*If base cost → not needed here*				
	‘.rng(r).		*→ not needed here*				

Table 12 Objects resource ***rsc****(r)* – “Library” of available resources – Example 3: **Income (PV FiT)**

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# of resource (option)*	–			
2	***typ.***		*Modeling type*				
	‘.idx		[2 1 2 2] → *[EC|IN|OP|EX]* → *Provision of electricity FiT*				
	‘.nme		{‘Economic resource’, ‘Income’; ‘Operation’, ‘External’; ‘EC’, ‘IN’, ‘OP’, ‘EX’}				
	‘.des		{‘Feed-in tariff for on-site generated energy.’}				
	stk(s).		*Stakeholder*[48]				
	‘.idx		#				
	‘.lnk		*# of stakeholder (owner + operator) in stk “library”*				
	‘.<xxx>		*Additional info*				
	cmp(c).		*Component*[49]				
	‘.idx		#				

[46] Due to TMY data, no specific year is given (“yyyy” = “xxxx”). Could be also given in relative time.
[47] PE resources not necessarily power values, but e.g. temperature or wind speed. However, here still covered by ***pwr*** and ***ene***. To be reconsidered!
[48] **Provider** of ***rsc***.
[49] **Allowed recipients** of ***rsc***.

‘.lnk		# of **cmp** in **cmp** lib
‘.typ.		*Modeling type*
“.idx		[1 2 5 1] → *[EL|DP|RD|PV] PV systems*
‘.<xxx>		*Additional info*
drt		*Same as in Example 1*
asm		*Same as in Example 1*
prd(p).[50]		*Single resource period elements*
‘.idx		[1]
‘.typ.		*Modeling type*
“.idx		[0] *→ t.b.d., e.g. 0 – indifferent, 1 – WD, 2 – WE, 3 – spc.*
“.nme		{‘ ‘} *→ {‘Fixed Tariff’, ‘–’}*
“.des		{‘ ‘} *→ {‘Time-indifferent compensation for PV feed-in.’}*
‘.drt	***nme***	{‘From’, ’To’, ’Duration’;’ From’, ‘To’, ‘Dur’}
	des	{‘Starting time (beginning point), absolute time’, ‘Ending time (ending point), absolute time’, ‘Duration in hours’}
	scp	[2 2]
	val	{‘2017-01-01T00:00:00’, ‘2018-01-01T00:00:00’, ‘8760’}
	dtp	{‘char’, ‘char‘, ‘float’}
	unt	{‘yyyy-mm-ddTHH:MM:SS’, ‘yyyy-mm-ddTHH:MM:SS’, ‘hours’}
	upp	[Inf Inf Inf]
	low	[0 0 0]
‘.pwr		*If peak power restrictions → not needed here*
‘.ene		*If capacity restrictions → not needed here*
‘.bsi[51]	***nme***	{'Base income – band', 'Base income – period'; 'BIb', 'BIp'}
	des	{‘Base income, if applied in any instance of this potentially repeated period (EUR/band).’, ‘Base income, if applied in this instance of the period (EUR/period).’}
	scp	[2 2]
	val	[0 0]
	dtp	{‘float’, float‘}
	unt	{‘EUR’, ‘EUR’}
	upp	[Inf Inf]
	low	[0 0]
‘.opi[52]	***nme***	{‘Time stamp’, 'Operational income'; ‘Time’, 'OI’}
	des	{‘Time stamp (beginning of time-step).’, ‘Energy-specific income.’}
	scp	[2 2]
	val	[0 0.11; 0.25 0.11; <…> ; 47.75 0.11][53]
	dtp	{‘float’, float‘}
	unt	{‘h’, ‘EUR/kWh’}

[50] Only one (default) range in this case
[51] Base income (in period), if any (unlikely).
[52] Energy income (in period) → operational income, kWh-rate.
[53] By default, all time-steps all modeled with equal size. Deviating time-series will be adapted.

	upp	[Inf Inf]	
	low	[0 0]	
'.rng(r).			*→ not needed here*

7 Components[54]

This is the user-determined project-specific “library” of available components to be considered. Component objects include energy distribution, conversion, storage, and facilities for demand provision (effective loads itself are defined in section 8). All *vertical* components (= all ***cmp*** except connection lines) are not part of networks and will be directly allocated to nodes (in the ***sys*** object). These components either convert energy between different sub-domain networks (either intra-domain or inter-domain), store energy at (sub-)domain level[55], or facilitate “consumption” of energy (energy sinks). In contrast, *horizontal* components (= connection lines for spatial distribution between nodes) are by definition part of (sub-)domain networks. Inter-domain vertical components are associated with a certain domain according to “common sense” (= main output in most cases), e.g. CHP plants are linked to the electric ***dmn*** and heat pumps to the thermal heat ***dmn***[56]. A list of ***cmp*** types is given in the appendix, Tab. A 4.

Each component has both **general** (defined for any/most) and **specific** modeling parameters. For **general operation constraints** (each dispatchable component, several inputs + outputs possible) the following rules apply:

In terms of input and output power (ports), there is

- One **main input** and one **main output** (the first one specified for each),
- One **controllable** entity (matching with the type of rated capacity), which is used to specify all other entities (directly or indirectly).

To characterize the operational value of each power element, the following rules apply:

1. Controllable entity	- no dependency; an operational range (min-max), if applicable,
2. Main entities	- depend on controllable entity,
3. Other entities	- depend on controllable entity,
4. Other entities	- depend on main entities,
5. Outputs	- depend on inputs.

Depending on the kind of entity, the operational dependences and efficiencies can be specified in the order of these rules. Three examples are given in Table 13 to Table 15. Note that the operational efficiencies can be defined by also including capacity-dependent and absolute offsets. For the controllable entity, an operational range with a lower and upper boundary relative to the capacity is defined, including the possibility to set a size-independent absolute offset.

Table 13 Dependences of operational power – example 1: controllable entity is output 1 (**default**)

		Entities applicable for defining dependences							
		Absolute offset	Capacity-dependent	Input 1	Input 2	Input 3	Output 1	Output 2	Output 3
Entities to be defined	Input 1	+	+	▨	–	–	+ (2)	–	–
	Input 2	+	+	+ (4)	▨	–	+ (3,4)	–	–
	Input 3	+	+	+ (4)	–	▨	+ (3,4)	–	–
	Output 1	+	+	– (1)	– (1)	– (1)	▨	– (1)	– (1)
	Output 2	+	+	+ (4,5)	+ (5)	+ (5)	+ (3,4)	▨	–
	Output 3	+	+	+ (4,5)	+ (5)	+ (5)	+ (3,4)	–	▨

[54] Alternative: Elements, Equipment, Technologies, Facilities, … Currently only actual energy generation, distribution, storage, load provision elements are considered. Need to be complemented by further additional / auxiliary components with relevant impact (cost, operation) like automation, control, communication a.s.o.
[55] Input energy type usually equals the output one, even if a conversion takes place within the storage.
[56] Heat pumps are part of the thermal domain if regarded as generation device. From the electric system PoV they are electric load devices.

Table 14 Dependences of operational power – example 1: controllable entity is input 1

		Entities applicable for defining dependences							
		Absolute offset	Capacity-dependent	Input 1	Input 2	Input 3	Output 1	Output 2	Output 3
Entities to be defined	**Input 1**	+	+	▨	– (1)	– (1)	– (1)	– (1)	– (1)
	Input 2	+	+	+ (3)	▨	–	–	–	–
	Input 3	+	+	+ (3)	–	▨	–	–	–
	Output 1	+	+	+ (2,5)	+ (5)	+ (5)	▨	–	–
	Output 2	+	+	+ (3,5)	+ (5)	+ (5)	+ (4)	▨	–
	Output 3	+	+	+ (3,5)	+ (5)	+ (5)	+ (4)	–	▨

Table 15 Dependences of operational power – example 2: controllable entity is output 2

		Entities applicable for defining dependences							
		Absolute offset	Capacity-dependent	Input 1	Input 2	Input 3	Output 1	Output 2	Output 3
Entities to be defined	Input 1	+	+	▨	–	–	–	+ (2)	–
	Input 2	+	+	+ (4)	▨	–	–	+ (3)	–
	Input 3	+	+	+ (4)	–	▨	–	+ (3)	–
	Output 1	+	+	+ (5)	+ (5)	+ (5)	▨	+ (2)	–
	Output 2	+	+	– (1)	– (1)	– (1)	– (1)	▨	– (1)
	Output 3	+	+	+ (5)	+ (5)	+ (5)	+ (4)	+ (3)	▨

The dependence of cost functions on the installed capacity (to consider the effect of scale) can be defined as summarized in Table 16.

Table 16 Options to calculate size-dependent investment cost

#	Function type		Equation
1	**Linear function** (offset + linear factor)	Ln	$C_{\text{Inv}} = a_{\text{lin}} \cdot P_{\text{el}} + b_{\text{lin}}$
2	**Polynomial function** (like Ln + quadr. fact.)	Pl	$C_{\text{Inv}} = a_{\text{pol}} \cdot P_{\text{el}}^2 + b_{\text{pol}} \cdot P_{\text{el}} + c_{\text{pol}}$
3	**Power function**	Pw	$C_{\text{Inv}} = a_{\text{pot}} \cdot P_{\text{el}}^{b_{\text{pot}}} + c_{\text{pot}}$
4	**Exponential function**	Ex	$C_{\text{Inv}} = a_{\exp} \cdot \exp\left(b_{\exp} \cdot P_{\text{el}}^{c_{\exp}}\right) + d_{\exp}$
5	**Logarithmic function**	Lg	$C_{\text{Inv}} = a_{\ln} \cdot \ln\left(P_{\text{el}}^{b_{\ln}} + c_{\ln}\right) + d_{\ln}$

Following these guidelines, for resources to be "stored" in the ***cmp*** "library" at least the following attributes need to be given:

- Component type (***typ***), also if discrete/continuous (***dcr***),
- Lifetime (***lft***), CAPEX (***cpx***), OPEX (***opx***),
- For each mode (***mde***, if several possible):
  - Rated capacity (***rat***),
  - For each input (***inp***) + output (***out***): constraints/bounds for power (***pwr***) + gradient (***grd***)[57],
  - *Optional:* - boundaries for run time (***run***) + stand-still time (***std***),
    - start-up penalties (***stu***).
- Component-specific modelling values (***spc***).

[57] For storage elements also boundaries for energy (***ene***) and ***grd*** of ***soc***.

Other attributes <u>can</u> be defined/stored here, if applicable to all instances in all scenarios. Otherwise, in the ***sys*** object[58]:

- Presence in existing system (***plc***), optimization options (***opt***), and optimized system (***set***),
- Linked stakeholders (***stk***) as owners/operators,
- Linked resources (***rsc***),
- Node-specific installation characteristics (***spc***).

Table 17 Objects component ***cmp****(c)* – attributes and grouping of attributes → "Library" of available components

#	Group / attribute	field	Content / example	Num ?	Relevant for GUI	Relevant for MW	Relevant for SE
1	***idx***		*# of component (option)*	–			
2	***typ***		*Modeling type*				
	'.idx		[1 1 2 1 1 1] → *[EL|DP|HE|CE|LV|AC]*				
	'.nme		{'Electricity', 'Dispatchable Generator', 'Heat driven', 'Combustion engine', 'Low voltage', 'Alternate current'; 'EL', 'DP', 'HE', 'CE', 'LV', 'AC'}				
	'.des		{'Combustion engine driven AC generator for low voltage applications with heat recovery (Combined heat and power).'}				
3	***mnf.***		*Manufacturer info*				
	'.idx		#				
	'.nme		{'Generic manufacturer', 'Generic model'; 'GN', 'GN'}				
	'.des		{'Generic model.'}				
	dcr	***nme***	{'Discretized Component'}				
		des	{'Indicates, if the component must be treated as discrete in size (1, relevant for technologies with one large main element such as stand-alone generators) or can be handled as continuously scalable technology (0, relevant for components with many small elements such as PV systems).'}				
		scp	[1 1]				
		val	[0] → *0 – continuous; 1 – discrete capacity/size cmp*				
		dtp	{'bin'}				
		upp	[1]				
		low	[0]				
	lft[59]	***nme***	{'Lifetime'; 'LT'}				
		des	{'Design lifetime under normalized conditions.'}				
		scp	[1 1]				
		val	[25]				
		dtp	{'int'}				
		unt	{'a'}				
		upp	[Inf]				
		low	[0]				
	cpx(c).[60]		*CAPEX*				

[58] All attributes can be stored in lib, and then overwritten individually in ***sys*** object. (General rule)
[59] *Could* specify several values, distinguishing between different lifetime definitions for economic and technical evaluation ("columns") as well as sub-elements, see ***cpx***(c) ("rows")
[60] Initial investment costs. Several sub-elements (modelling independent) can be specified. Can e.g. also include sub-element "installation".

‘.idx		*# of **cpx** element (first is main)*				
‘.nme		{‘Main component’; ‘ ‘}				
‘.des		{‘Initial investment cost for main component.’}				
‘.lnk		[1 1] → *[**idx@mod idx@rat**] → main mode, main cap*				
‘.ivc[61]	***nme***	{‘Capacity-independent offset’, ‘Capacity linear factor’, ‘Capacity quadratic factor’, ‘Power function factor’, ‘Power function exponent’, ‘Logarithmic function outer factor’, ‘Logarithmic function inner addend’, ‘Logarithmic function inner exponent’, ‘Exponential function outer factor’, ‘Exponential function inner factor’, ‘Exponential function inner exponent’; 'pPl1', ‘pPl2’, ‘pPl3', 'pPw1', ‘pPw2’, 'pLg1', ‘pLg2’, ‘pLg3’, ‘pEx1', ‘pEx2’, ‘pEx3’}				
	des	{'Capacity-independent absolute cost.’, ‘Capacity dependent relative cost.’, ‘Polynomial function: quadratic factor.’, ‘Power function: linear factor.’, ‘Power function: exponent.’, ‘Logarithmic function: outer factor’, ‘Logarithmic function: inner addend’, ‘Logarithmic function: inner exponent’, ‘Exponential function: outer factor’, ‘Exponential function: inner factor’, ‘Exponential function: inner exponent'}				
	scp	[2 1]				
	val	[1000 50 0 0 0 0 0 0 0 0 0]				
	dtp	{‘float’, ‘float’, ‘float’, ‘float’, ‘float’, ‘float’, ‘float’, ‘float’, ‘float’, ‘float’, ‘float’}				
	unt	{‘EUR’, ‘EUR/kW’, ‘–‘, ‘–‘, ‘–‘, ‘–‘, ‘–‘, ‘–‘, ‘–‘, ‘–‘, ‘–‘}				
	upp	[Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf]				
	low	[0 0 0 0 0 0 0 0 0 0 0]				
‘.rpc[62]		*Replacement cost, same structure as investment cost*				
opx(o).[63]		*OPEX, excl. fuel*				
‘.idx		*# of **opx** element (first is main)*				
‘.nme		{‘Main component’; ‘ ‘}				
‘.des		{‘Operation and maintenance cost for main component (except fuel cost.’}				
‘.lnk		[1 1] → *[**idx@mod idx@rat**] → main mode, main cap*				
‘.omc		*Capacity-defined annual OPEX, same structure as **ivc**.*				
‘.omo[64]	***nme***	{'Capacity-independent absolute cost.’, ‘Capacity dependent relative cost.’				

[61] Defining the elements of the several options of initial investment cost functions.
[62] Replacement cost refer to a complete exchange of main equipment, which might be less than the initial investment cost. Other categories could be defined, e.g. repair cost resulting in improvement (extension of lifetime)
[63] OPEX excl. fuel costs.
[64] Operation-defined OPEX excl. fuel (arising only when operated). ***Omc*** and ***omo*** are complementary.

			‘Cost relative to electrical output power (#1)’, ‘Cost relative to thermal heat output power (#2)’, ‘Cost relative to natural gas input power (#1)’, ‘Cost relative to auxiliary energy input power (#2)’; ' ', ' ', ' ', ' ', ' ', ' '}
		des	{‘ ’, ‘ ’, ‘ ’, ‘ ’, ‘ ’, ‘ ’}
		scp	[2 2]
		val	[0.5 0.2 0.02 0 0 0 0]
		dtp	{‘float’, ‘float’, ’ float’, ‘float’, ‘float’, ‘float’}
		unt	{‘EUR/h’, ‘EUR/kWh’, ‘EUR/kWh’, ‘EUR/kWh’, ‘EUR/kWh’, ‘EUR/kWh’}
		upp	[Inf]
		low	[0]
	cap.[65]		*Rated capacity*
	‘.lnk		[1 1] → ***[idx@mod idx@rat]*** → *main mode, main cap*
	’.bnd[66]	***nme***	{‘Minimum capacity’, ‘Maximum capacity’}
		des	{‘Minimum applicable rated capacity, for which the applied technology characteristics are valid’, ‘Maximum applicable rated capacity, for which the applied technology characteristics are valid’}
		scp	[1 1]
		val	[5 100]
		dtp	{‘float’, ‘float’}
		unt	{‘kW’, ‘kW’} → *‘kW’ or ‘kWh’, depending on **cap.lnk***
		upp	[Inf]
		low	[0]
	‘.sze(s).[67]		*Size related to rated capacity*
	‘’.idx		*# of size relation*
	“.typ.		*Modeling type*
	“’.idx		[1] → *1 – area, 2 – volume, 3 - …*
	““.nme		{‘Area’}
	““.des		{‘Floor space required for installation.’}
	“.rel	***nme***	{'Capacity-independent area’, 'Capacity-dependent area’; ' ', ' '}
		des	{'Capacity-independent floor space required for installation.', 'Capacity-dependent floor space required for installation.'}
		scp	[1 1]
		val	[10 2]
		dtp	{‘float’, ‘float’}

[65] Mode-independent, abstract boundary values. Mode-specific capacities are linked with linear factor.
[66] In case of discrete technologies (***dcr***): the first value is the actual capacity; the second could (implicitly) specify the # of units for which the economic figures are valid for (e.g. specific CAPEX decreases with increasing # of units).
[67] = Size: Additional capacity constraints, e.g. required floor space or volume, given in <xxx>/kW_{rat}. In case of discrete components, also possible to be given in <xxx> per ***cmp*** unit. Further quantities can be listed in the same attribute (added in the same row).

	unt	{'m2', 'm2/kW'} *→ 'kW' depending on **cap.lnk**; 'm2' depending on **sze.typ***
	upp	[Inf Inf]
	low	[0 0]
mod(m).		*If several modes are possible, e.g. reversible HPs*
'.idx		*# of mode*
'.nme		{'<Mode 1>'}
'.des		{' '}
'.typ		*Modeling type*
''.idx		[0] *→ cannot run simultaneously to other modes*
''.nme		{'Non-simultaneous'}
''.des		{'Cannot run simultaneously to any other operational mode.'}
'.cap(c)[68]		*Mode-specific rated capacity factor*
''.idx		#
''.lnk[69]		# of port *→ [2 2] → second output port*[70]
''.rel[71]	***nme***	{'Capacity factor'}
	des	{'Linear factor, which the mode-specific rated capacity is linked to the overall component capacity.'}
	scp	[1 1]
	val	[1]
	dtp	{'float'}
	unt	{'–'} *→ depending on quantity types (kW ←→ kWh)*
	upp	[Inf]
	low	[0]
''.<xxx>		*Additional: e.g. definition of short-term peak capacity (relative values, time, ...)*
'.inp(1).		*Input ports (first = main input)*
''.idx		[1] *# of input port*
''.lnk		***dmn*** *type:* [4 1 1] *→ [FU|GS|NG] → natural gas*
''.<xxx>		*Additional constraints, e.g. by structural features, min/max pwr, min/max ene (life) etc.*
''.pwr[72]	***nme***	{'Power, size-independent offset', 'Power, size-dependent offset', 'Power, operation-dependent, thermal heat output', ' ', ' ', ' '}
	des	{'Size-independent absolute power offset', 'Size-dependent relative power offset', 'Power relative to thermal heat output (#2)'}

[68] Link to modelling capacity! Further might be indicated, automatically calculated from ***opn*** constraints (if given by user it would be a constraint in addition to the ones in ***opn***). Only 1 capacity needed, all other dependencies are handled in ***opn***. cpf ionly unequal to 1 if rated capacity links to different mode.
[69] Linked to design element ***dsn*** "type" (***inp***, ***out***, ***soc***) + ***idx***,
[70] In this example, the capacity is given as rated thermal heat
[71] In case of discrete technologies (***dcr***): the first value is the actual capacity; the second could (implicitly) specify the # of units for which the economic figures are valid for (e.g. specific CAPEX decreases with increasing # of units).
[72] For main input (i=1), cf. Table 15

	scp	[1 2]				
	val	[10 0 2.5]				
	dtp	{‘float’, ‘float‘, ‘float’}				
	unt	{‘kW’, ‘–’, ‘–‘}				
	upp	[Inf Inf Inf]				
	low	[0 -Inf 0]				
‘.inp(2).		*Auxiliary electrical power*				
‘’.idx		[2]				
‘’.lnk		***dmn** type:* [[1 2 1] → *[EL|MV|AC] → MV AC electricity*				
‘’.<xxx>		*Additional constraints*				
‘’.pwr[73]	*nme*	{‘Power, size-independent offset’, ‘Power, size-dependent offset’, ‘Power, operation-dependent, thermal heat output’, ‘Power, operation-dependent, natural gas input’, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘}				
	des	{‘Size-independent absolute power offset’, ‘Size-dependent relative power offset’, ‘Power relative to thermal heat output (#2)’, ‘Power relative to natural gas input (#1)’}				
	scp	[1 2]				
	val	[0.2 0.05 0 0]				
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’}				
	unt	{‘kW’, ‘–’, ‘–‘,‘–‘}				
	upp	[Inf Inf Inf Inf]				
	low	[0 -Inf 0 0]				
‘’.grd[74]	*nme*	{‘Down-gradient, size-independent offset’, ‘Down-gradient, size-dependent offset’, ‘Down-gradient, operation-dependent’, ‘Up-gradient, size-independent offset’, ‘Up-gradient, size-dependent offset’, ‘Up-gradient, operation-dependent’; ‘<1>‘, ‘<2> ‘,‘<3>’, ‘<4>‘, ‘<5>‘, ‘<6>‘}				
	des	{‘Absolute power offset for downward power gradient in [kW]’, ‘Downward power gradient relative to rated capacity [–]’, ‘Downward power gradient relative to operational power [–]’, ‘Absolute power offset for upward power gradient in [kW]’, ‘Upward power gradient relative to rated capacity [–]’, ‘Upward power gradient relative to operational power [–]’}				
	scp	[1 2]				
	val	[0 1 0 0 1 0]				
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’, ‘float‘, ‘float’}				
	unt	{‘kW/h’, ‘–/h’, ‘–/h‘, ‘kW/h’, ‘–/h‘, ‘–/h‘}				
	upp	[Inf Inf Inf Inf Inf Inf]				
	low	[0 0 0 0 0 0]				
‘’.ene[75]		*Boundaries for energy over time, t.b.d.*				

[73] For all other inputs (i>1), cf. Table 15

[74] Gradient $\Delta P / \Delta t$ boundaries: Values for max possible gradients related to downward and upward modulation. To be defined for most interesting power. Only one operation-dependent value (linked to entity).

‘.out(1).		*Output ports (first = main output)*				
‘’.idx		[1] *# of output port*				
‘’.lnk		***dmn*** *type:* [1 2 1] → *[EL|MV|AC]* → *MV AC electricity*				
‘’.<xxx>		*Additional constraints*				
‘’.pwr[76]	***nme***	{‘Power, size-independent offset’, ‘Power, size-dependent offset’, ‘Power, operation-dependent, thermal heat output (#2)’, ‘Power, operation-dependent, natural gas input (#1)’, ‘Power, operation-dependent, auxiliary electricity input (#2)’; ‘ ‘, ‘ ‘, ‘ ‘,’ ‘, ‘ ‘}				
	des	{‘Size-independent absolute power offset’, ‘Size-dependent relative power offset’, ‘Power relative to thermal heat output (#2)’, ‘Power relative to natural gas input (#1)’, ‘Power relative to auxiliary electricity input (#2)’}				
	scp	[1 2]				
	val	[0 0 0.4 0 0]				
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’, ‘float’}				
	unt	{‘kW’, ‘–’, ‘–‘, ‘–‘, ‘–‘}				
	upp	[Inf Inf 1 1 1]				
	low	[-Inf -Inf 0 0 0]				
‘.out(2).		*For output port 2* → *thermal heat (waste heat CHP)*				
‘’.idx		[2]				
‘’.lnk		***dmn*** *type:* [2] → *[HE]* → *Thermal heat*				
‘’.<xxx>		*Additional constraints*				
‘’.pwr[77]	***nme***	{‘Min power, size-independent offset’, ‘Min power, size- dependent’, ‘Max power, size-independent offset’, ‘Max power, size-dependent’; ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘}				
	des	{‘Size-independent absolute minimum power boundary offset’, ‘Minimum power boundary, relative to rated capacity’, ‘Size- independent absolute maximum power boundary offset’, ‘Maximum power boundary, relative to rated capacity’}				
	scp	[1 2]				
	val	[0 0.5 0 1]				
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’}				
	unt	{‘kW’, ‘–’, ‘kW‘, ‘–‘}				
	upp	[Inf 1 Inf 1]				
	low	[0 0 0 0]				
‘.out(3).		*For output port 3* → *pwr example to show principle*				
‘’.pwr[78]	***nme***	{‘Power, size-independent offset’, ‘Power, size-dependent offset’, ‘Power, operation-dependent, thermal heat output (#2)’,				

[75] Could be defined in ***dsn***. In ***opn*** deviations could be specified. Likely to be applied only for rated capacity.

[76] For main output (o=1), cf. Table 15

[77] Second output, but this is the modelling capacity → no dependency, but min/max

[78] For all other (hypothetical) outputs (o>1)

		‘Power, operation-dependent, electricity output (#1)’, ‘Power, operation-dependent, natural gas input (#1)’, ‘Power, operation-dependent, auxiliary electricity input (#2)’; ‘ ‘, ‘ ‘, ‘ ‘,’ ‘, ‘ ‘}
	des	{‘Size-independent absolute power offset’, ‘Size-dependent relative power offset’, ‘Power relative to thermal heat output (#2)’, ‘Power relative to electricity output (#1)’, ‘Power relative to natural gas input (#1)’, ‘Power relative to auxiliary electricity input (#2)’}
‘’.grd		*Same definitions as for inputs*
‘’.ene		*Same definitions as for inputs*
‘.soc[79]		*For **cmp** with storage capacity only*
‘’.idx		*# of storage (assumed to be just one possible)*
‘’.lnk		***dmn** type:* [1 2 1] *→ [EL|MV|AC] → MV AC electricity*
‘’.<xxx>		*Additional constraints*
“.ene	***nme***	{‘Min SOC, size-independent offset’, ‘Min SOC, size-dependent’, ‘Max SOC, size-independent offset’, ‘Max SOC, size-dependent’; ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘}
	des	{‘Size-independent absolute offset to minimum SOC boundary’, ‘Minimum SOC boundary, relative to rated capacity’, ‘Size-independent absolute offset to max. SOC boundary’, ‘Maximum SOC boundary, relative to rated capacity’}
	scp	[1 2]
	val	[0 0.5 0 1]
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’}
	unt	{‘kWh’, ‘–’, ‘kWh’, ‘–‘}
	upp	[Inf 1 Inf 1]
	low	[0 0 0 0]
‘’.grd[80]	***nme***	{‘Down-gradient, size-independent offset’, ‘Down-gradient, size-dependent offset’, ‘Down-gradient, operation-dependent’, ‘Up-gradient, size-independent offset’, ‘Up-gradient, size-dependent offset’, ‘Up-gradient, operation-dependent’; ‘<1>‘, ‘<2> ‘,‘<3>’, ‘<4>‘, ‘<5>‘, ‘<6>‘}
	des	{‘Absolute offset for downward SOC gradient in [kWh]’, ‘Downward SOC gradient relative to rated capacity [–]’, ‘Downward SOC gradient relative to operational SOC [–]’, ‘Absolute offset for upward SOC gradient in [kWh]’, ‘Upward SOC gradient relative to rated capacity [–]’, ‘Upward SOC gradient relative to operational SOC [–]’}
	scp	[1 2]
	val	[0 1 0 0 1 0]
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’, ‘float‘, ‘float’}
	unt	{‘kWh/h’, ‘–/h’, ‘–/h‘, ‘kWh/h’, ‘–/h‘, ‘–/h‘}

[79] Almost identical to definition of ***inp\|out*** except “SOC” + “kWh” instead of “power” + “kW”. NOTE! Storages have two processes (*charg\|disch*) + soc. If *disch* differs from *charg*, to be defined in mod(**2**).
[80] The ***soc.grd*** of storage is (quite) equivalent to the ***inp.pwr***. Defining both would (a) not make too much sense and could (b) evoke problems in modelling/solving.

	upp	[Inf Inf Inf Inf Inf Inf]				
	low	[0 0 0 0 0 0]				
‘.tme.		*Definition of stand-still and running time min/max boundaries*				
‘’.run	*nme*	{‘Min runtime, size-independent offset’, ‘Min runtime, size-dependent’, ‘Max runtime, size-independent offset’, ‘Max runtime, size-dependent’, ‘Operation-mode overlapping’; ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘}				
	des	{‘Size-independent absolute minimum runtime.’, ‘Size-dependent relative minimum runtime.’, ‘Size-independent absolute maximum runtime.’, ‘Size-dependent relative maximum runtime.’, ‘Linked to other operation modes.’}				
	scp	[1 2]				
	val	[5 0 0 0]				
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’, ‘int‘}				
	unt	{‘h’, ‘h/kW’, ‘h‘, ‘h/kW’, ‘–‘}				
	upp	[Inf Inf Inf Inf Inf]				
	low	[0 0 0 0 0 0]				
‘’.std	*nme*	{‘Min standstill time, size-independent offset’, ‘Min standstill time, size-dependent’, ‘Max standstill time, size-independent offset’, ‘Max standstill time, size-dependent’, ‘Operation-mode overlapping’; ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘}				
	des	{‘Size-independent absolute minimum standstill time.’, ‘Size-dependent relative minimum standstill time.’, ‘Size-independent absolute maximum standstill time.’, ‘Size-dependent relative maximum standstill time.’, ‘Linked to other operation modes.’}				
	scp	[1 2]				
	val	[2 0 0 0]				
	dtp	{‘float’, ‘float‘, ‘float’, ‘float’, ‘int‘}				
	unt	{‘h’, ‘h/kW’, ‘h‘, ‘h/kW’, ‘–‘}				
	upp	[Inf Inf Inf Inf Inf]				
	low	[0 0 0 0 0 0]				
‘.stu(s).[81]		*Start-up (penalties)*				
‘’.idx		#				
‘’.nme		{‘ ‘}				
‘’.des		{‘ ‘}				
‘’.tme	*nme*	{‘Boundary standstill time’, ‘Start-up time, size-independent offset’, ‘Start-up time, size-dependent’; ‘ ‘, ‘ ‘}				
	des	{‘Maximum allowed duration of standstill without applying the penalty’, ‘Size-independent absolute actual start-up time.’,				

[81] Several kind of penalties can be considered, that might have different time-ranges depending on the duration of stand-still. Either ***pwr*** or ***cst*** or both (***pwr*** and ***cst*** are complementary).

			‘Size-dependent relative actual start-up time.}
		scp	[1 2]
		val	[0.001388 0.000277]
		dtp	{‘float’, ‘float‘}
		unt	{‘h’, ‘h‘}
		upp	[Inf Inf]
		low	[0 0]
	‘’.pwr	***nme***	{‘Start-up power, size-independent offset’, ‘Start-up power, size-dependent’; ‘ ‘, ‘ ‘}
		des	{‘Size-independent absolute actual start-up power.’, ‘Size-dependent relative actual start-up power.}
		scp	[1 2]
		val	[1 0.5]
		dtp	{‘float’, ‘float‘}
		unt	{‘kW’, ‘–‘}
		upp	[Inf Inf]
		low	[0 0]
	‘’.opc[82]	***nme***	{‘Start-up cost, size-independent offset’, ‘Start-up cost, size-dependent’; ‘ ‘, ‘ ‘}
		des	{‘Size-independent absolute actual start-up cost.’, ‘Size-dependent relative actual start-up cost.}
		scp	[2 2]
		val	[5 2]
		dtp	{‘float’, ‘float‘}
		unt	{‘EUR’, ‘EUR/kW‘}
		upp	[Inf Inf]
		low	[0 0]
	‘.spc.		***Cmp*** *(sub-)type specific operation parameters*

In the actual ***cmp*** instance in ***sys***, all results are summarized in ***set***:

	set.		
	‘.cpx.		*CAPEX*
	‘.opx.		*OPEX*
	‘.sze.		*Size*
	‘.mod(m).		*Modes*
	‘’.rat(r)		***8***
	‘’.inp(i)		
	‘’.out(o)		
	‘’.soc		

[82] (Operational) cost, e.g. due to increased maintenance.

8 Loads[83]

This is the user-determined project-specific “library” of effective loads (net energy)[84] to be considered / covered. It includes both non-influenceable fixed loads and flexible loads, providing operational degrees of freedom (DoF)[85]. In the most restricted case, fixed load profiles are time-series created during the modelling process prior to the optimization run.

The basic characterizing elements of loads are average requested **power** P per time-step t (kW = kWh/h), and **cumulated energy** E over a certain time period p (kWh)[86]. To consider coherent load operation processes, e.g. a certain program of domestic appliances / industrial procedure (or energy phases within), also the definition of **shapes/sequences** S is possible. A shape is a pre-specified sequence of $P(t)$ in successive t – thus automatically defining $E(p)$ – that might be further constrained as non-interruptable.

$$E(p) = \sum_{p} P(t) \cdot dt \quad (1)$$

Load owners making flexibility potential available and providing flexibility (to either support network operators or energy suppliers) will be rewarded with **externally offered incentives** (demand rate / working price), but may be faced with additional **internal cost** at the same time. For the extreme case of non-available energy supply under special (emergency) conditions, such as blackouts of supply network or generators, a special **value-of-lost-load** (VoLL) need to be defined[87]. Thus, that there need to be defined several ranges of load-internal additional cost for distinct deviation in coverage of the default load. Additionally, loads can be defined as including (being combined with) an up-stream storage element[88] in order to provide easily commensurable and rateable flexibility[89]. Different kind of effective loads may be defined as follows. Related types of cost are summarized in Table 18.

1. Non-sequential flexibility, cf. Figure 4
    a. If a default load value $P(\mathrm{t})$ is supplied with less (or more) power certain internal cost can be defined, with the value depending on the $P(\mathrm{t})$ deviation range (Figure 4, left),
    b. At the same time, the deviation from a default energy demand $E(p)$[90] can be penalized in the same way (Figure 4, right);
2. Sequential flexibility, cf. Figure 5
    a. Sub-process with fixed shape S ($P(t_X)$, and therewith E) can be shifted in time; internal cost depending t deviation range can be defined (Figure 5, left),
    b. Several sub-processes can be either combined in chronological order (Figure 5, right) or in non-chronological order (batch processing, *not depicted*).
3. Compounded load
    a. An upstream component ***cmp***, e.g. a storage, is linked to a load of type 1 or 2

[83] Alternative: Consumption, Demand, Energy sinks, …
[84] *Nutzenergie* = Actual amount of energy used to cover the need of customer/consumer
[85] Marginal note: Concepts like load shedding / shifting are always related to external perspective, driven by external needs/desires with related offered incentive. If load internal constraints are defined appropriately, load can apply accordingly (fulfil request + benefit from incentive). Does not make sense to squeeze external concepts into actual load definition (but need to be considered in modeling).
[86] One could consider to define other constraints as well like for technologies (***grd***, ***tme***, …)
[87] This can be likewise modelled by an upstream „virtual“ generator attached to the load (with peak power and energy related cost components). That way, the constraints of the actual load (min power) can still be fulfilled, allowing the solver to find a feasible solution.
[88] In need for a definable operational “analogue” state-of-charge (SOC). Depending on PoV, storage not part of effective load, but facility to provide/supply load → grey area with need for distinction to ***cmp***.
[89] Fitting the concept of Thomas Walter.
[90] Value is linked to the default $P(\mathrm{t})$,

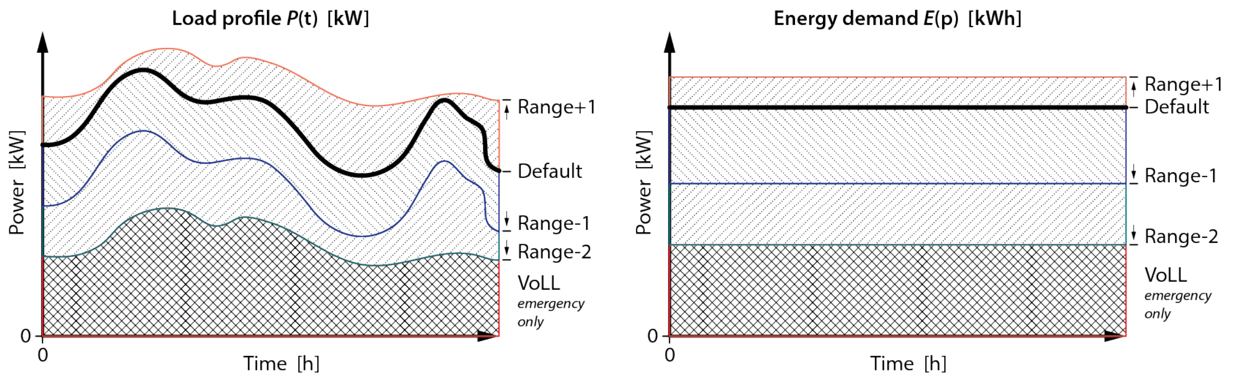

Figure 4 Non-sequential load – operation ranges of power profile *P*(t) (*left*) and energy demand *E*(p) (*right*)

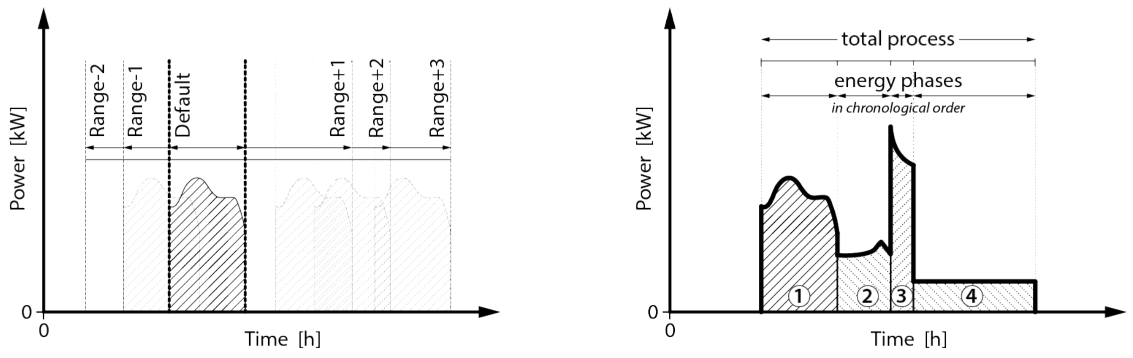

Figure 5 Sequential load – temporal operation ranges (*left*) and sequencing (*right*)

Table 18 Load internal cost due to provision of flexibility

#	Cost group	Cost type	Non-sequential		Sequential
			Power range	Energy range	Time range
	Base fee	Fee per band[91]	x	x	x
		Fee per period	x	x	x
	Work price	E(dt)	x	–	–

Following these guidelines, for loads to be “stored” in the ***loa*** “library” at least the following attributes need to be given:

- Load type (***typ***)[92],
- Duration of load time-series (***drt***) + the assemblage of single load periods within (***asm***),
- These single load periods (***prd***) with
  - Default values of load power (time-series) (***pwr***),
  - Ranges defining internal cost in case of deviation (***rng***) [93].
  - Type-specific info (***spc***) to build the time-series by a separate simulation[94].

Other attributes can be defined/stored in here, if applicable to all instances in all scenarios. Otherwise, in the ***sys*** object[95]:

- Presence in existing system (***plc***), optimization options (***opt***), and optimized system (***set***),
- Linked resources (***rsc***),
- Linked stakeholders (***stk***).

[91] Band = the total of repeating periods of the same type.
[92] See appendix, Tab. A 5.
[93] At least one range should be given, defining VoLL (this might be just one high value = penalty).
[94] E.g. for space heating (thermal building simulation), hot water usage, or electric load profiles.
[95] One can of course “store” all attributes of an element in lib + overwrite them individually in each ***sys*** object. (General rule: specific info → higher priority than general one)

Table 19 Objects load ***loa**(l)* – attributes and grouping of attributes

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*# of **loa** in **loa** "library"*	–			
2	***typ.***		*Modeling type*				
	'.idx		[1 1 1] *→ [EL|NS|RS] → non-sequential residential load*				
	'.nme		{'Electricity', 'Non-sequential Load', 'Residential' ; 'EL', 'NS', 'RS'}				
	'.des		{'An aggregated electricity load profile of a group of 10 residential buildings.'}				
	rsc(r).		*Aligned resources*[96]				
	'.idx		*# of **rsc** @**loa***				
	'.lnk		*# of **rsc** in **rsc** lib*				
	stk(s).		*Aligned stakeholder*[97]				
	'.idx		*# of **stk** @**loa***				
	'.lnk		*# of **stk** in **stk** lib*				
	drt	***nme***	{'From', 'To', 'Duration';' From', 'To', 'Dur'}				
		des	{'Starting time (beginning point), absolute time', 'Ending time (ending point), absolute time', 'Duration in hours'}				
		scp	[1 1]				
		val	{'2017-01-01T00:00:00', '2018-01-01T00:00:00', '8760'}				
		dtp	{'char', 'char', 'float'}				
		unt	{'yyyy-mm-ddTHH:MM:SS', 'yyyy-mm-ddTHH:MM:SS', 'hours'}				
		upp	[Inf Inf Inf]				
		low	[0 0 0]				
	asm[98]	***nme***	{'From', 'Load element'; 'time', 'LE'}				
		des	{'Starting time (beginning point), absolute time', ''}				
		scp	[1 1]				
		val	{'2017-01-01T00:00:00', '1'; … '2017-12-22T00:00:00', '1'} *→ idx #*				
		dtp	{'char', float'}				
		unt	{'yyyy-mm-ddTHH:MM:SS', '–'}				
		upp	[Inf Inf]				
		low	[0 0]				
	prd(p).		*Single load period elements (can be applied repeatedly)*				
	'.idx		[1]				
	'.typ.		*Modeling type*				
	''.idx		[2] *→ 0 – indifferent, 1 – weekday, 2 – weekend, 3 – spc.*				
	''.nme		{'Week-end profile.', 'WE'}				
	''.des		{'Typical weekend profile of this residential block of houses.'}				

[96] Applied in ***loa*** lib if instance already linked to ***rsc*** instance in ***rsc*** lib (e.g. weather data, for calculation of heat load).
[97] Applied in ***loa*** lib if instance already linked to ***stk*** instance in ***stk*** lib (e.g. one specific facility owner).
[98] Total load assembled from single period elements.

‘.drt[99]	***nme***	{‘From’, ’To’, ’Duration’;’ From’, ‘To’, ‘Dur’}
	des	{‘Starting time (beginning point), relative time’, ‘Ending time (ending point), relative time’, ‘Duration in hours’}
	scp	[1 1]
	val	[0 24 48]
	dtp	{‘float’, float‘, ‘float’}
	unt	{‘hour’, ‘hour’, ‘hours’}
	upp	[Inf Inf Inf]
	low	[0 0 0]
‘.pwr[100]	***nme***	{‘Time stamp’, ‘Power’; ‘time’, ‘P’}
	des	{‘Time stamp (beginning of time-step).’, ‘Average power per time-step.’}
	scp	[1 1]
	val	[0 52.34; 0.25 30.7; <…> ; 47.75 41.9][101]
	dtp	{‘float’, float‘}
	unt	{‘hour’, ‘kW’}
	upp	[Inf Inf]
	low	[0 0]
‘.ene		*Energy ← not necessary to be given explicitly*
‘.bsc		*Base cost ← no internal cost related to default*
‘.opc		*Operational cost ← no internal cost related to default*
‘.rng(r).[102]		*Separate ranges in **prd***
‘‘.idx		*# of **rng** in **prd***
‘‘.pwr[103]	***nme***	{‘Time stamp’, ‘Max power’; ‘time’, ‘P’}
	des	{‘Time stamp (beginning of time-step).’, ‘Upper peak power boundary per time-step.’}
‘‘.ene	***nme***	{‘Max energy’; ‘E’}
	des	{‘Upper boundary of cumulated energy.’}
	scp	[1 1]
	val	[600]
	dtp	{‘float‘}
	unt	{‘kWh’}
	upp	[Inf]
	low	[0]
‘.bsc[104]	***nme***	{'Base cost – band', 'Base cost – period'; 'BCb', 'BCp'}
	des	{‘Base cost, if applied in any instance of this potentially repeated period (EUR/band).’, ‘Base cost, if applied in this instance of the period (EUR/period).’}
	scp	[2 1]

[99] Same structure as ***drt*** of total load, start + end are given in plain (relative) hours
[100] Default load curve (power)
[101] As default, all time-steps all modeled with equal size. Deviating time-series will be adapted.
[102] Defining additional ranges with upper boundaries. Last one is VoLL.
[103] If no pwr / ene is given the next higher is taken → for VoLL no extra boundary has to be defined.
[104] Base cost: Load-internal cost caused by deviation from default.

	val	[30 0]				
	dtp	{‘float’, ‘float‘}				
	unt	{‘EUR’, ‘EUR’}				
	upp	[Inf Inf]				
	low	[0 0]				
‘.opc[105]	*nme*	{‘Time stamp’, 'Operational cost'; ‘Time’, 'OC’}				
	des	{‘Time stamp (beginning of time-step)’, ‘Energy-specific cost’}				
	scp	[2 1]				
	val	[0 0.11; 0.25 0.11; <…> ; 47.75 0.11]				
	dtp	{‘float’, float‘}				
	unt	{‘h’, ‘EUR/kWh’} → *related to kWh deviation!*				
	upp	[Inf Inf]				
	low	[0 0]				
‘.spc.		***Loa** (sub-)type specific modeling parameters*				

NOTE! Sequential loads need to be defined slightly different. Definition will follow …

[105] Operational cost: Load-internal cost caused by deviation from default.

9 Stakeholders

This is the user-determined project-specific “library” of participating stakeholders to be considered. The definition includes owners and operators of networks, components, and loads, determining economic system boundaries. Only this distinction allows for interactions / exchange and definition of optimization objectives / benefits.

Table 20 Objects stakeholder ***stk*** – attributes and grouping of attributes → “Library” of stakeholders involved

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		[1]	–			
2	***nme***		{‘Company X-rayYankeeZulu’, ‘XYZ’}				
3	***des***		{‘XYZ is a large ...’}				
2	***typ.***						
	‘.idx		[2] → *(1) Owner, (2) Operator, (3) both*[106]				
	‘.nme		{‘Operator’; ‘OP’}				
	‘.des		{‘Responsible for the operation/performance of an element.’}				
	???		Specific info/data could be linked to stakeholders, e.g. minimum internal rates, …				

10 Settings

What is necessary? No objects needed?

Table 21 Settings ***set*** – attributes and grouping of attributes

#	Group / attribute	field	Content / example	Num ?	Relevant for		
					GUI	MW	SE
1	***idx***		*t.b.d. (classification?)*	–			
2	***typ***		*t.b.d. (classification?)*				
3	***nme***		{‘Network’}				
4	***des***		{‘Specification of network design.’}				

←

QGIS

OPF (lin, act) – like normal PF, but now costs aligned to specific generators

[106] Additional definitions to be added later.

11 Appendix

Tab. A 1 Classification of main domains ***dmn*** → also used for physical resources ***rsc***, ***cmp***, and ***loa***

Domain #[107]				Description	
1				Electric, EL	→ *Secondary energy carrier*
2				Thermal heat, HE[108]	→ *Secondary energy carrier*
3				Thermal cooling, CL	→ *Secondary energy carrier*
4				Fuels, FU	→ *Secondary energy carrier*
5				Primary energy (non-grid, on-site), PE	

Tab. A 2 Classification of (sub-) domains ***dmn***

Type # levels				Description	
	Sub-domain				
	1	2	3		
1				Electric, EL	→ *Secondary energy carrier*
	Potential				
	1			Low voltage, LV	
	2			Medium voltage, MV	
	3			High voltage, HV	
		AC / DC			
		1		Alternate current, AC	
		2		Direct current, DC	
2				Thermal heat, HE	→ *Secondary energy carrier*
	Potential				
	1			Low temperature, LT	
	2			Medium temperature, MT	
	3			High temperature, HT	
		Medium			
		1		Water	
3				Thermal cooling, CL	→ *Secondary energy carrier*
	Potential				
	1			Low temperature, LT	
	2			Medium temperature, MT	
	3			High temperature, HT	
		Medium			
		1		Water	
4				Fuels, FU	→ *Secondary energy carrier*
	Aggregate phase (under norm conditions)				
	1			Gaseous fuels, GF	→ *grid-bound (lines)*
	2			Liquid fuels, LF	→ *grid-bound (lines)*
	3			Solid Fuels, SF	→ *non-grid*
		Specific types			

[107] Main domain number: 1st position of ***typ*** # for ***dmn*** and ***cmp***, 2nd position in ***rsc*** (1st is general resource type: (1) phs = energy, (2) ecn = …)
[108] NOTE! Heat and Cooling could be merged into *one* domain ("Thermal", "TH"). The current division is motivated by seemingly different application rather then actual networks. In fact, even application is common in wide sections (space conditioning) if one neglect hot water (res) and process heat (ind).

	1	1		Natural gas
		2		Biogas[109]
		3		Hydrogen
	2	1		Fuel oil / Diesel
	3	1		Wood pellets
			Pressure (Alternative 1)	*→ for grid-bound fluids only*
			1	Low pressure, LP
			2	Medium pressure, MP
			3	High pressure, HP
			Composition (Alternative 2)	
			1	Low caloric value, CV
			2	Medium caloric value, MV
			3	High caloric value, HV
5				Primary energy (non-grid, on-site), PE[110]
	Scope			
	1			Solar, SL
	2			Wind, WD
	3			Temperature (ambiance + ground), TM
	4			Humidity, HM
	5			Hydro, HD

[109] Compared to natural gas: different composition (less CH_4 share), also different source/impact (renewable)
[110] PE only form _virtual_ domains (virtual buses) to which the input ports of ***cmp*** connect to (no man-made infrastructure, it is just there) → thus only relevant for model generation, not for user; just included for a better understanding / clear structuring of model (info can be copied from ***rsc*** allocation).

Tab. A 3 Classification of resources ***rsc*** = exchange (inputs mainly) with system external environment

Type # levels				Description	
	Sub-levels				
	1	2	3		
1				Physical (energy carriers)	
	1			Electric, EL[111]	→ *Energy carrier provision from external*
		1		Low voltage, LV	
		2		Medium voltage, MV	
		3		High voltage, HV	
			1	Alternate current, AC	
			2	Direct current, DC	
	2			Thermal heat, HE	→ *Energy carrier provision from external*
		1		Low temperature, LT	
		2		Medium temperature, MT	
		3		High temperature, HT	
			1	Water	
	3			Thermal cooling, CL	→ *Energy carrier provision from external*
		1		Low temperature, LT	
		2		Medium temperature, MT	
		3		High temperature, HT	
			1	Water	
	4			Fuels, FU	→ *Energy carrier provision from external*
		1		Gaseous fuels, GF	→ *grid-bound (lines)*
		2		Liquid fuels, LF	→ *grid-bound (lines)*
		3		Solid Fuels, SF	→ *non-grid*
		1	1	Natural gas	
			2	Biogas	
			3	Hydrogen	
		2	1	Fuel oil / Diesel	
		3	1	Wood pellets	
	5			Primary energy (non-grid),	→ *On-site PE*[112]
		1		Solar, SL	→ *weather data*
		2		Wind, WD	→ *weather data*
		3		Temperature (ambiance + ground), TM	→ *weather data (only ambiance)*
		4		Humidity, HM	→ *weather data*
		5		Hydro, HD	
		1	1	Extraterrestrial Direct Normal Radiation {Wh/m2}	
			2	Direct Normal Radiation {Wh/m2}	
			3	Global Horizontal Irradiance {W/m2}, GH	
			4	Direct Horizontal Irradiance {W/m2}, DH	
			5	Diffuse Horizontal Irradiance {W/m2}, FH	

[111] Assumed wholesale-market (external connection); the following lower sub-domain positions determine the *ntw*-bus @node to which connected to. If these are not specified, any will do.
[112] For PE resources, the definitions are about physical conditions and restrictions (e.g. what time-series/weather data are available), in general no or very minor economic information (cost)

			6	Global Horizontal Illuminance {lux},	
			7	Direct Normal Illuminance {lux}	
			8	Total Sky Cover {–}	
			9	Opaque Sky Cover {–}	
			10	Albedo {-}, AL	
		2	1	Wind Speed {m/s}, WS	
			2	Wind Direction {°}, WD	
		3	1	Dry Bulb Temperature {°C},	→ *from weather data*
			2	Dew Point Temperature {°C},	→ *from weather data*
			3	Ground Temperature @depth	
			4	Ground Water Temperature	
		4	1	Relative Humidity {–}	
			2	Precipitable Water {mm}	
			3	Liquid Precipitation Depth {mm}	
			4	Liquid Precipitation Quantity {hr}	
		5	1	Surface stream speed {m/s}	→ *for run-of-the river hydroelectric*
			2	Surface stream direction {°}	→ *for run-of-the river hydroelectric*
			3	*Missing: Tidal, …*	
2				Economic ()	
	1			Income	
	2			Financing	
	1	1		Investment (CAP__)	
		2		Operation (OP__)	
	1	1	1	Grant	
			2	Premium	
	1	2	1	Intern	
			2	Extern	

Tab. A 4 Classification of components ***cmp*** (technologies)[113]

Type # levels	Sub-levels 1	2	3	#5	#6	Description		
1						Electric, EL		
2						Thermal heat, HE		
3						Thermal cooling, CL		
4						Fuels, FU *(e.g. hydrogen production)*		*not yet modeled*
	1					Dispatchable generation/conversion (intra + inter main ***dmn***),	DP[114]	
	2					Non-dispatchable generation/conversion,	ND[115]	
	3					Storing,	ST	
	4					Inter-nodal (inter-bus, intra-network) connections (line, pipe)	LN	
	5					Final facilities (supplying effective loads),	FF[116]	*spc model t.b.d.*
	6					Assembled / compound component,	AS[117]	*spc model t.b.d.*
		Driving principle, source-related[118]						
	1-3	1				Electrically driven,	__|__|EL	
	1-3	2				Heat (engine) driven,	__|__|HE	
	1-3	3				Mechanically driven,	__|__|ME	
	1-3	4				Chemically driven,	__|__|CH	
	1-3	5				Radiation-driven,	__|__|RD	
				EL: #5: LV/MV/HV, #6: AC/DC, #7: CHP				
1				1		LV		
1				2		MV		
1				3		HV		
1					1	(to) AC		
1					2	(to) DC		
1	**1/2**					1 HR (heat recovery → CHP)		
1	**1**	1	1		1	Converter AC-AC,	EL|DP|EL|AC|__|AC	*spc model t.b.d.*
1	1	1	1		2	Converter AC-DC,	EL|DP|EL|AC|__|DC	*spc model t.b.d.*
1	1	1	2		1	Converter DC-AC[119],	EL|DP|EL|DC|__|AC	*spc model t.b.d.*
1	1	1	2		2	Converter DC-DC,	EL|DP|EL|DC|__|DC	*spc model t.b.d.*
1	1	1	3		1	Frequency changer,	EL|DP|EL|FQ|__|AC	*spc model t.b.d.*
1	1	2	1			Combustion engines,	EL|DP|HE|CE|__|AC	
1	1	2	2			Turbo machines,	EL|DP|HE|TM|__|AC	*spc model t.b.d.*

[113] Highest level: main output. The entire classification serves the goal to structure ***cmp*** types mainly from a technological PoV (→ modeling + easy definition/integration of new types). However, it cannot avoid being random to some extent and most probably do not fit all kind of audiences.
[114] Such as transformers / converters in EL ***dmn*** or heat exchangers in HE ***dmn***.
[115] "Non-dispatchable" (= not freely dispatchable) refers to a volatile source (Renewables). The ***cmp*** output depends on the fluctuating availability of the source, but may be dispatched to a certain extent.
[116] Such as heating system convectors.
[117] Any combination of these sub-types, also possible to link loads. NOTE! Also the other sub-types may be (are) compound systems in reality, but are treated as one in the system model.
[118] Specifically for generation (1-2) and storage (3). For other components it can match the main ***dmn*** #
[119] NOTE! Every DC-AC converter (*Wechselrichter*) can be operated as AC-DC converter (*Gleichrichter*), but not vice versa.

1	1	3	1		1	Motor-Generator (*Umformer*),	EL|DP|ME|AC|__|AC	*spc model t.b.d.*
1	1	3	2			Hydro (constant source),	EL|DP|ME|HD|__|AC	*spc model t.b.d.*
1	1	4	1			Fuel cell (galvanic cell),	EL|DP|CH||FC|__|DC	
1	**2**	3	1			Wind turbine,	EL|ND|ME|WT|__|AC	
1	2	3	2			Hydro (fluctuant),	EL|ND|ME|HD|__|AC	
1	2	5	1			PV systems,	EL|ND|RD|PV|__|DC	
1	**3**	3	1			Fly wheels,	EL|ST|ME|FW|__|AC	
1	3	4	1			Lead Acid battery,	EL|ST|CH|LA|__|DC	
1	3	4	2			Li-Ion battery,	EL|ST|CH|LI|__|DC	
1	**4**	1	1			Line type XX,	EL|LN|EL|XX|LV|AC	
				HE+CL: #5: LT, MT, HT #6: diverse				
				1-3		LT, MT, HT		
					1	For generation: specific type, e.g. air source HP, AR		
					1	For storage: water, WA		
2	**1**	1	1			Electric heater,	HE|DP|EL|EH	
2	1	1	2	1	1	Heat pump, air source	HE|DP|EL|HP|LT|AR	
2	1	2	1			Heating exchanger	HE|DP|HE|HX	
2	1	2	2			Heating boiler,	HE|DP|HE|HB	
2	**2**	5	1			Solar thermal system,	HE|DP|RD|ST	
2	**3**	3	1		1	Buffer storage (water tank)	HE|ST|ME|BF|__|WA	
2	**4**	2	1					
3	**1**	1	1			Compression chiller,	CL|DP|EL|CC	
3	1	2	1			Absorption chiller,	CL|DP|HE|AC	
3	**3**	3	1		1	Buffer storage (water tank)	CL|ST|ME|BF|__|WA	

Tab. A 5 Classification of effective loads ***loa*** (end-user demand)

Type # levels	Sub-levels 1	2	3	Description	
1				Electric, EL	
2				Thermal heat, HE	
3				Thermal cooling, CL	
4				Fuels, FU *(e.g. hydrogen demand)*	*not yet modeled*
	1			Non-sequential load,	NS[120]
	2			Sequential load,	ND[121]
	3			???	
		Scope			
		1		Residential	
		2		Commercial	
		3		Industrial	
		4		???	
1	1	1		Residential electric load	EL|NS|RS
1	1	2		Commercial electric load	EL|NS|RS
1	1	3		Industrial electric load	EL|NS|RS
2	1	1	1	Residential building: space heating	HE|NS|RS|SP
2	1	1	2	Residential building: hot water use	HE|NS|RS|HW
3	1	1	1	Residential building: space cooling	CL|NS|RS|SP

Tab. A 6 Classification of stakeholders ***stk***

Type # levels	Sub-levels 1	2	3	Description
1				Owners
2				Operators

[120] Aggregated loads, which can serve flexibility within certain power and energy bands, cf. Figure 4.
[121] Sequence: A definitive shape that can be shifted as a whole such as a washing machine program.